Eclipse Process Framework (EPF) Composer

Installation, Introduction, Tutorial and Manual

Table of Contents

1. EPF Composer Download and Installation

"The EPF Composer is a stand-alone application, using the Eclipse Rich Client Platform (RCP). Therefore it installs independently from other Eclipse programmes.

The EPF Composer requires an installed Java Runtime Environment (JRE). For the current version (1.5.0.4), the JRE version 1.5 must be installed prior to the EPF Composer. You can download and install the JRE from http://www.java.com.

1.1. Download

Go to http://www.eclipse.org/epf/ and click on "**Downloads"**. A new webpage comes up (http://www.eclipse.org/epf/downloads/downloads.php) and you will see:

Figure 1. EPF Downloads

Under EPF Composer, click on "**Download**" which brings up yet another web page: (http://www.eclipse.org/epf/downloads/tool/tool_downloads.php).

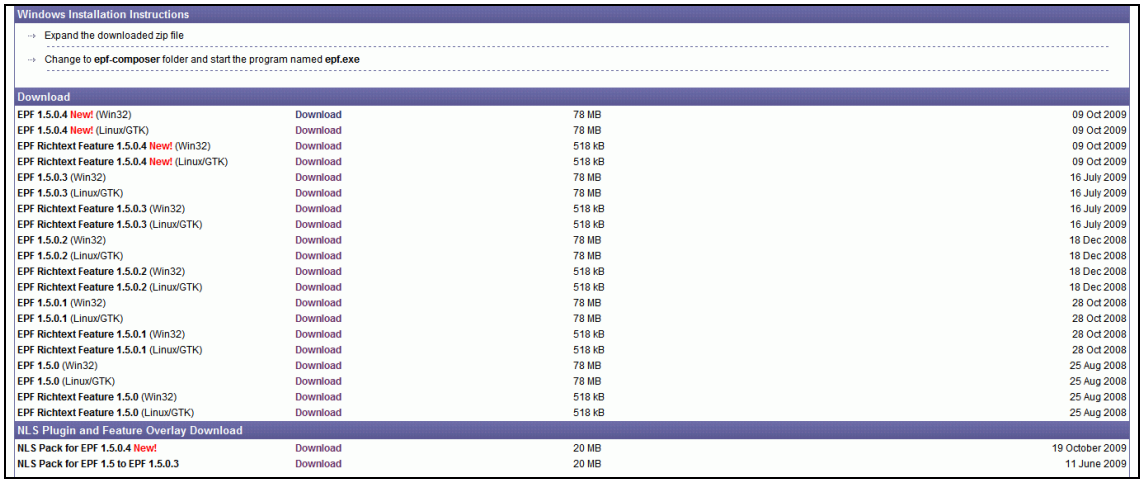
Figure 2. EPF Downloads

Click on "**Download"** and yet another web page comes up:

http://www.eclipse.org/epf/downloads/tool/epf1.5.0_downloads.php

Scroll down the page until you see the section Download.

Figure 3. EPF Downloads



Windows Installation Instructions

- Expand the downloaded zip file
- Change to **epf-composer** folder and start the program named **epf.exe**

Download

EPF 1.5.0.4 New! (Win32)	Download	78 MB	09 Oct 2009
EPF 1.5.0.4 New! (Linux/GTK)	Download	78 MB	09 Oct 2009
EPF Richtext Feature 1.5.0.4 New! (Win32)	Download	518 kB	09 Oct 2009
EPF Richtext Feature 1.5.0.4 New! (Linux/GTK)	Download	518 kB	09 Oct 2009
EPF 1.5.0.3 (Win32)	Download	78 MB	16 July 2009
EPF 1.5.0.3 (Linux/GTK)	Download	78 MB	16 July 2009
EPF Richtext Feature 1.5.0.3 (Win32)	Download	518 kB	16 July 2009
EPF Richtext Feature 1.5.0.3 (Linux/GTK)	Download	518 kB	16 July 2009
EPF 1.5.0.2 (Win32)	Download	78 MB	18 Dec 2008
EPF 1.5.0.2 (Linux/GTK)	Download	78 MB	18 Dec 2008
EPF Richtext Feature 1.5.0.2 (Win32)	Download	518 kB	18 Dec 2008
EPF Richtext Feature 1.5.0.2 (Linux/GTK)	Download	518 kB	18 Dec 2008
EPF 1.5.0.1 (Win32)	Download	78 MB	28 Oct 2008
EPF 1.5.0.1 (Linux/GTK)	Download	78 MB	28 Oct 2008
EPF Richtext Feature 1.5.0.1 (Win32)	Download	518 kB	28 Oct 2008
EPF Richtext Feature 1.5.0.1 (Linux/GTK)	Download	518 kB	28 Oct 2008
EPF 1.5.0 (Win32)	Download	78 MB	25 Aug 2008
EPF 1.5.0 (Linux/GTK)	Download	78 MB	25 Aug 2008
EPF Richtext Feature 1.5.0 (Win32)	Download	518 kB	25 Aug 2008
EPF Richtext Feature 1.5.0 (Linux/GTK)	Download	518 kB	25 Aug 2008

NLS Plugin and Feature Overlay Download

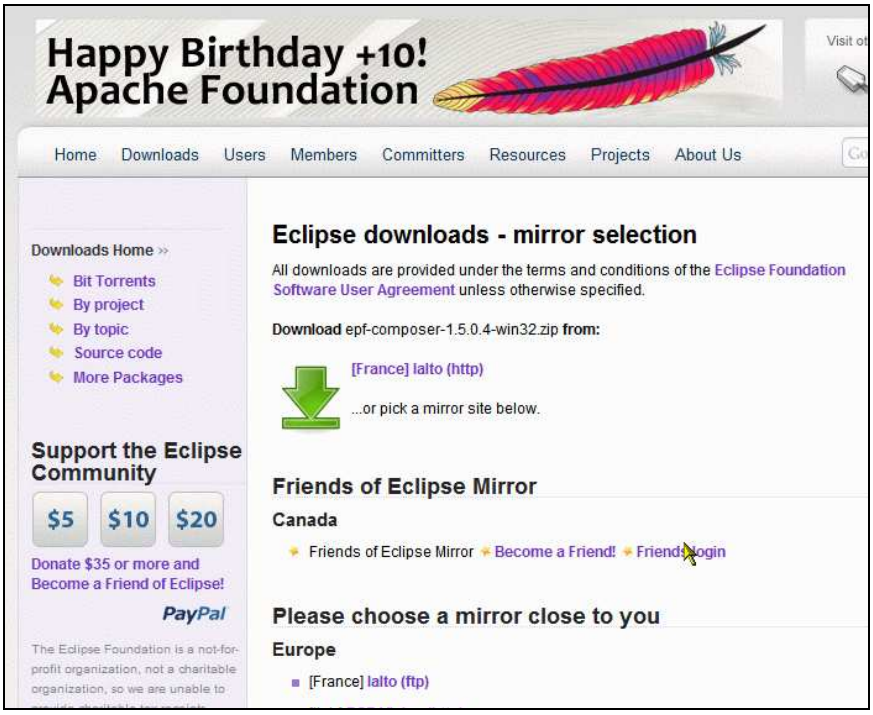
NLS Pack for EPF 1.5.0.4 New!	Download	20 MB	19 October 2009
NLS Pack for EPF 1.5 to EPF 1.5.0.3	Download	20 MB	11 June 2009

The NLS Plug-in is required if you want to use the EPF Composer and the published documentation in another language than English. The NLS Pack is installed by just merging the NLS directory with the existing EPF Composer directory.

> *Comment: We have to say if and when the EPF Rich Text Feature and/or NLS Plug-in and Feature Overlay Download must be download and installed and how to install them.*

Click on "**Download**" to the right of "EPF 1.5.04 **New**". That brings up yet another webpage. The page will depend upon the version that you have selected.

Figure 4. EPF Downloads



This time, you should click on the **green arrow (**not on "Download").

You are then asked to save the file: *epf-composer-1.5.0.4-win32.zip*.

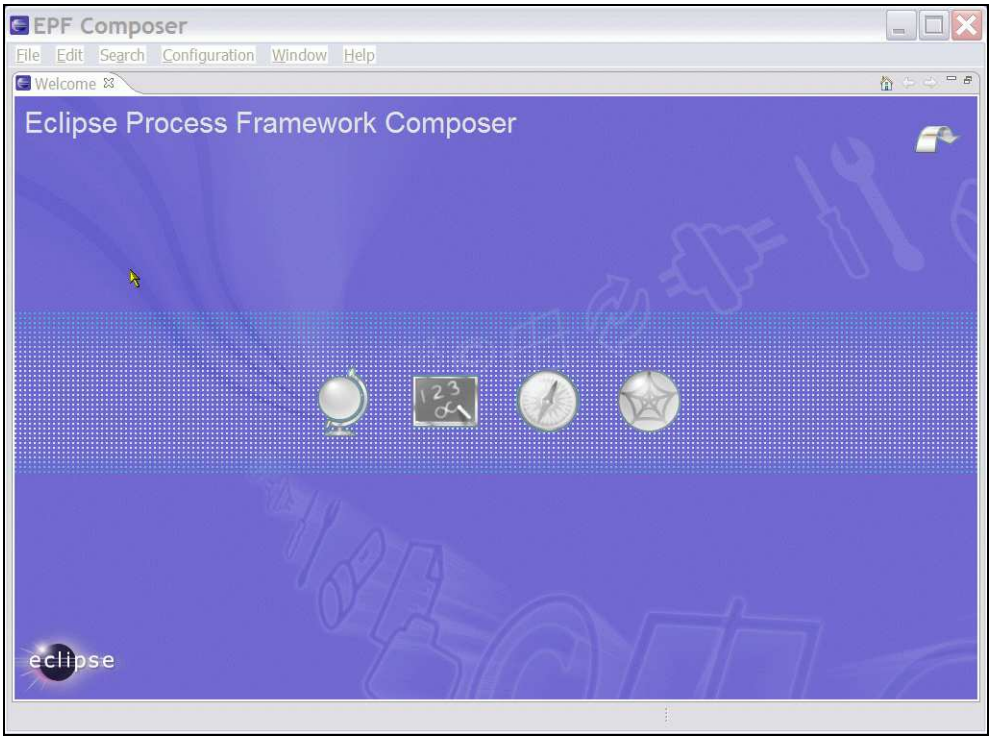
1.2. Installation

Once the download has finished, you have to extract the zip file. The result is a folder with the name "*epf-composer*" and you can either extract it directly to a destination folder or move it to a destination folder after extraction. The folder can be put under *c:\program files\eclipse*, *c:\epf-composer* or any other destination folder you fancy.

Then go to the destination folder, in our example the "*epf-composer*" in the root directory of the D: partition, and start the EPF composer programme by double-clicking on the **epf.exe** file (for future use, it is recommended to create a short-cut on the desktop).

The Eclipse Process Framework Composer "Welcome" screen appears.

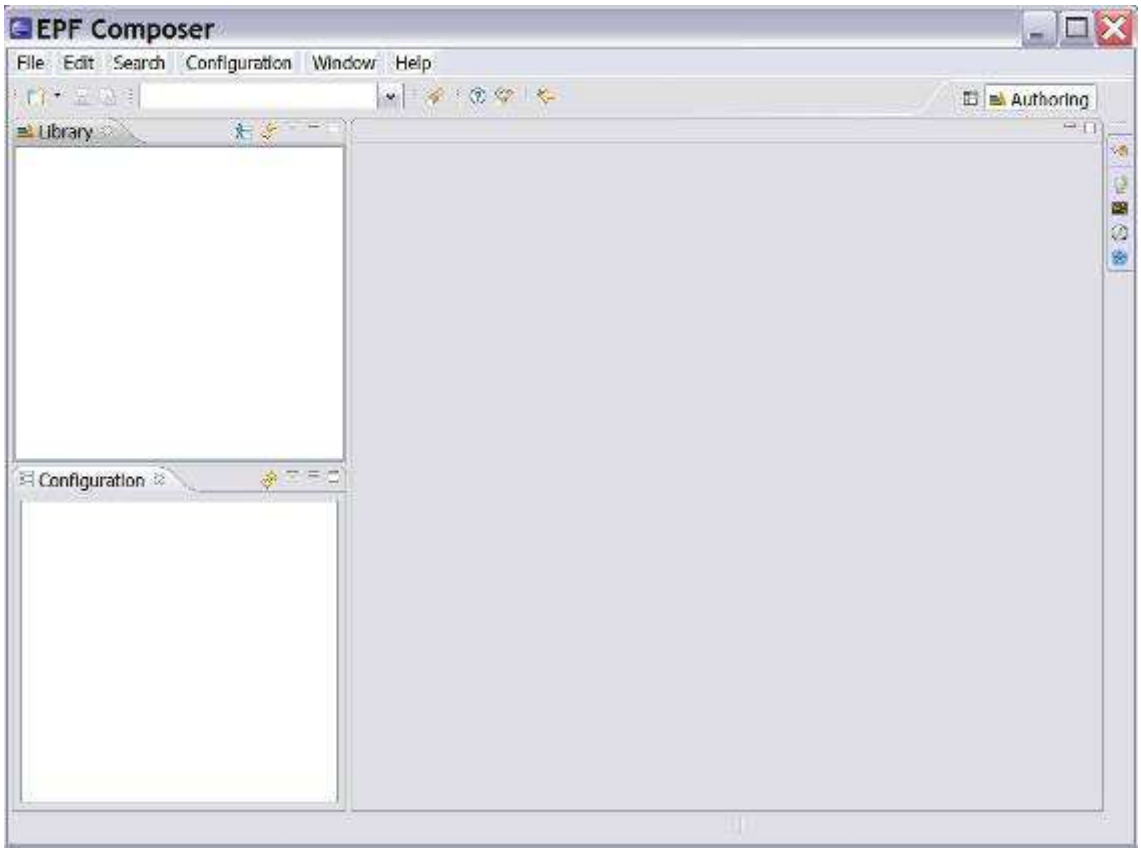
Figure 5. EPF Welcome Screen



Clicking on the arrow in the top right corner brings you directly to the Workbench. If you want to go back to the Welcome screen, go the menu item "Help" and select "Welcome".

The workbench screen is empty:

Figure 6. EPF Workbench Screen



After these steps, the application is ready for use. In most cases you will however download content created by others in order to reuse the content. This is explained in the next chapter.

1.3. Configuration

After installation, you should also pay attention to default folders used by EPF Composer. Select **Windows** → **Preferences**. In the preference dialogue window expand **Method** and click on **Authoring** and on **Publishing / Browsing**. After installation, they will point to folders:

- *C:\Documents and Settings\User\EPF\Method Libraries*;
- *C:\Documents and Settings\User\EPF\Publish*;

Note: Replace "User" with your computer login user name.

You can create similar folders elsewhere, for example in the epf-composer folder and make the application point to these folders instead.

In addition, the application creates the following folders:

- *C:\Documents and Settings\User\EPF\layout*
- *C:\Documents and Settings\User\EPF\workspace.150*

These folders contain files that keep track of application changes performed by the user. You can change the location of these files by creating a folder elsewhere, for example under the *epf_composer* folder and by changing the **epf.ini** file. The second line in the epf.ini file has the text*: @user.home/EPF/workspace.150*. Change the part before the mention of *workspace.150* to the path/folder you have created, for example to: *D:\epf-composer\Startup\workspace.150*.

By now, the application and your files are all to be found under the *epf-composer* folder. That makes the application portable and easy to backup.

2. Method Content Download and Installation

2.1. Introduction

You have multiple options to work with the EPF to create your own process documentation: (1) Create your content from scratch. (2) Use an existing content library and modify its content. (3) Use an existing content library to add specific content to your own documentation. Which one to choose depends on what amount of material you already have available, and how much of content you expect to be reusable from the provided libraries.

Already created method content can be imported, either by importing method libraries, method plug-ins or XML files.

All method contents are stored in a method library and importing already created content requires either an existing method library or the creation of a new library. Importing a method library creates a new method library, while importing a method plug-in or an XML file requires an existing method library or the creation of a new method library before importing.

Existing method content import sources:

- Library Configuration
- Method Plug-ins
- XML

By installing already created method elements and by changing and extending the content, the authors can create process descriptions that are appropriate for their environment, without having to start from scratch.

2.2. EPF Download Site

A library is a container for method plug-ins and method configuration definitions. All method elements are stored in a library. To download existing content from the EPF site, go back to the first download page as in previous Figure 1, here figure 7: “EPF Downloads”.

Figure 7. EPF Downloads

2.2.1. EPF Wiki

There is no need to download the EPF Wiki, at least not at this stage. You may however consider installing the EPF Wiki later.

An important success factor for living process descriptions is the possibility for the users to give feedback on the content. Even without the EPF Wiki, after you have published your process description, users can provide feedback using the appropriate link in the upper right corner of the browser window. This creates, for instance, an e-mail with a link to the currently displayed page, and the user can simply add his proposal for improvement. This feedback then may be collected, decided upon, and used to create a new release.

However, in some cases this is not sufficient to achieve user acceptance. The EPF Wiki is another mean to collect user feedback, by allowing the user to add the feedback directly to the provided content.

For the installation and usage of the EPF Wiki, please refer to the project home on http://www.eclipse.org/epf/downloads/epfwiki/downloads.php.

And

http://wiki.eclipse.org/EPF_Wiki_Installation_Guide

The installation of EPF Wiki will not be part of this document.

2.2.2. EPF Practices

The practices library is included as part of the EPF 1.5 Most of the open source content derives from OpenUP 1.0, refactored to support independent practices.

On the first download page (Figure 7): Click on "**Downloads**" under EPF Practices. A new webpage appears.

Figure 8. EPF Practices Downloads

Download the two zip files. Extracting the zip files results in two folders: "*epf_practices*" and "*epf_practices_published*".

The *epf_practices* library is an example of already created content in modifiable "**source format**" published as EPF method plug-ins, which can be edited once they have been loaded in the EPF Composer and which can subsequently be published.

The already provided published version is a static html website (set of web pages) that can be installed on any computer, even those that do not run the EPF Composer. Therefore, if you only want to use the out-of-the-box content "as is", you can simply unzip the *published*.zip, and run index.htm in your favourite browser and you will be able to navigate the already published content.

We recommend that you move the two folders under the "*epf-composer*" folder.

To install the epf practice libraries, start the EPF composer program by double-clicking on **epf.exe** again. The go to menu **File** → *Open* → **Method Library**. Click **Browse** and navigate to the "*epf_practices*" folder placed under the "*epf-composer*" folder and click OK. The file that the programme is looking for is named: "*library.xmi*" which it finds in the "*epf_practices*" folder.

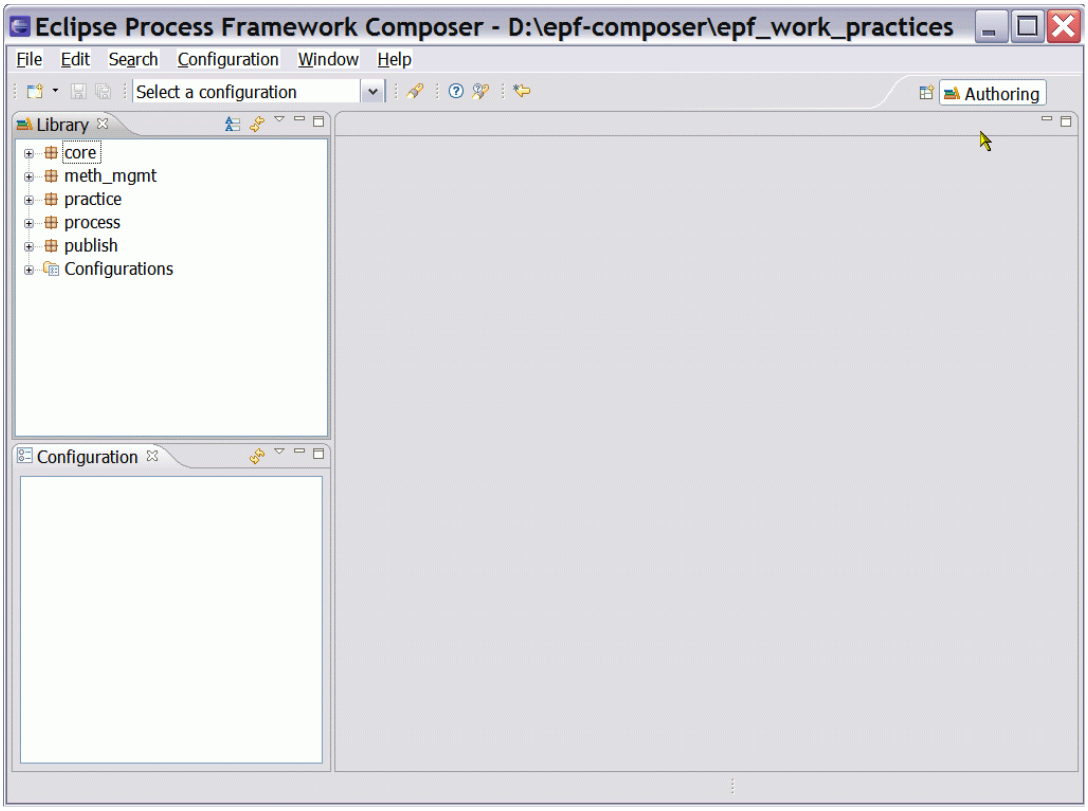
Then the "**Copy Library**" dialogue window pops up and tells you: "The library you are opening is a default library supplied with the composer."

Figure 9. Copy Library

What this message attempts to say, is that you have the choice of either using the files in the "***epf_work_practices***" folder or creating a separate folder (using the operating system's file utility) and then letting the epf-composer copy the existing files over to the new folder (for example "***epf_work_practices***"). In the latter case, you will have two sets of files, one working set that you can modify and another with the original pristine and unchanged files. In the second case you will always have an unchanged reference copy.

You can for example create a new folder using the operating system's File Explorer (in the case of MS Windows), with for example a folder name like "***epf_work_practices***". Afterwards, having switched back to the EFP application, you click on Browse and navigate so that it points to the new "***epf_work_practices***" folder. You then click **OK**, which brings you back to the pop-up "Copy Library" window and you click on **Copy** and wait a few seconds. The workbench screen has now changed. The title bar shows the path and the name of the working folder.

Figure 10. EPF Workbench Screen



2.2.3. OpenUP

Library

The OpenUP Library is included in the EPF Practices and does not need to be installed separately.

Then why do we complicate things by listing it on the webpage?

Published OpenUP web pages

The UpenUP web pages are however available for download. Extract the files from the "openup_published-1.5.0.4-20091008.zip" file and place them in the "*epf-composer*" folder. Double-click on **index.htm**, which will open the Internet browser and you will be able to navigate the already published content

2.2.4. Other Libraries

For the other libraries follow the same steps. On the EPF Downloads page, see Figure 1, the following libraries are available for downloading.

- Scrum
- XP
- MAM (Method Authoring Method) for Eclipse Practices Library (EPL)

2.3. Other Download Sites

Already created method content can be downloaded from different sites.

Do we have any examples of this?

2.3.1. TOGAF

There is a TOGAF 9 Method plug-in, which can be found at:

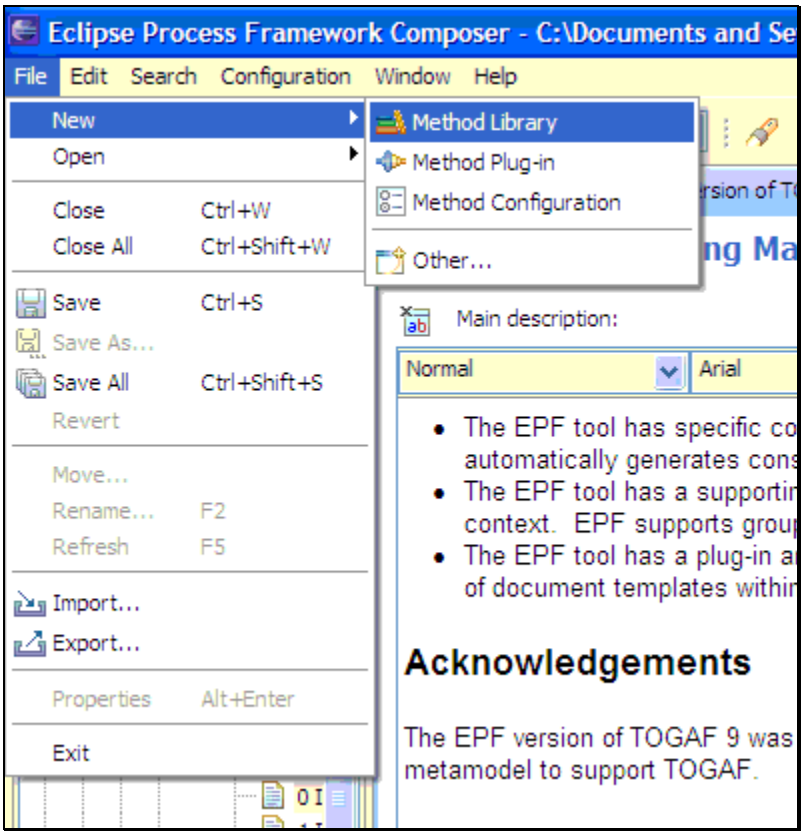
http://www.opengroup.org/architecture/togaf/epf_intro.html

Installation of a plug-in is different from the installation of a library.

Installation:

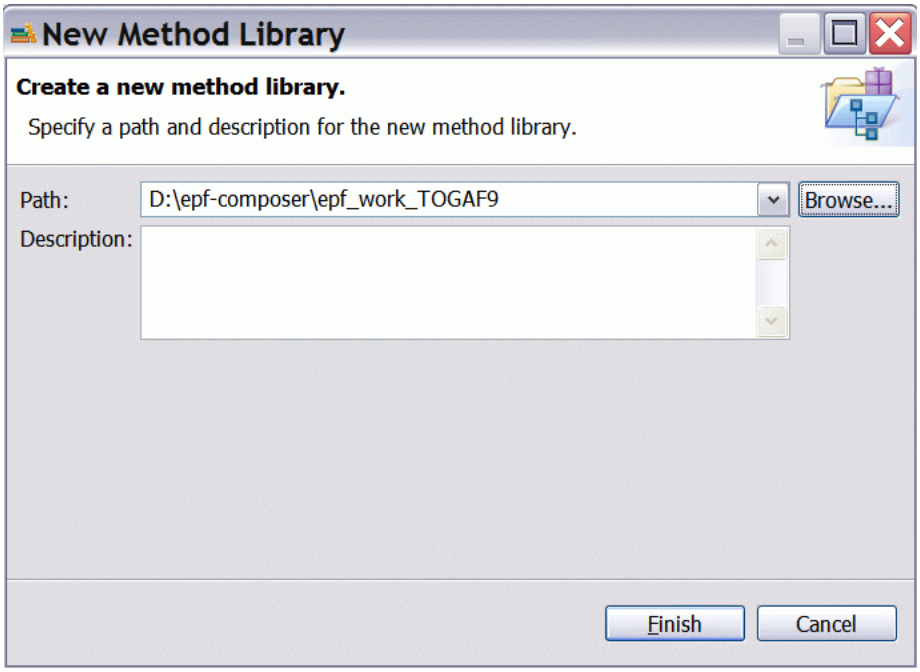
Once you have downloaded the TOGAF_9_Final_EPF_Export.zip file, the next step is to extract the zip file. The result is a folder and you can change its name. We called the folder "*TOGAF_Export*" for example. We suggest you place the folder under the "**epf-composer**" folder.

Figure 11. Create new Method Library



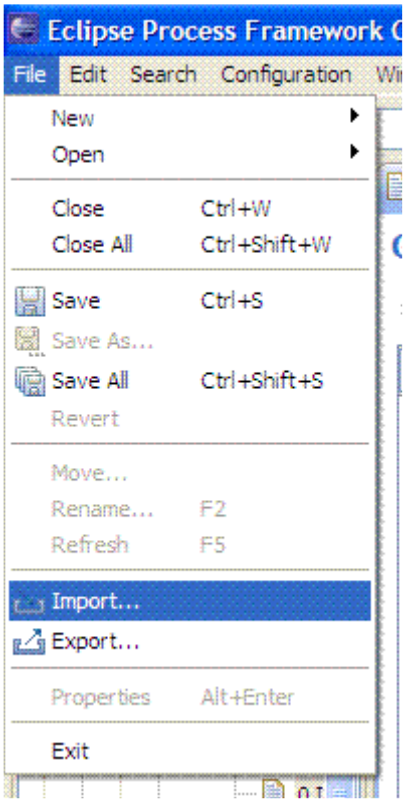
Again, you will create a working directory for the method; in our case we created a folder called "*epf_work_TOGAF9*". Then open EPF and create a new Method Library.

Figure 12. Create new Library Folder and point to the path.



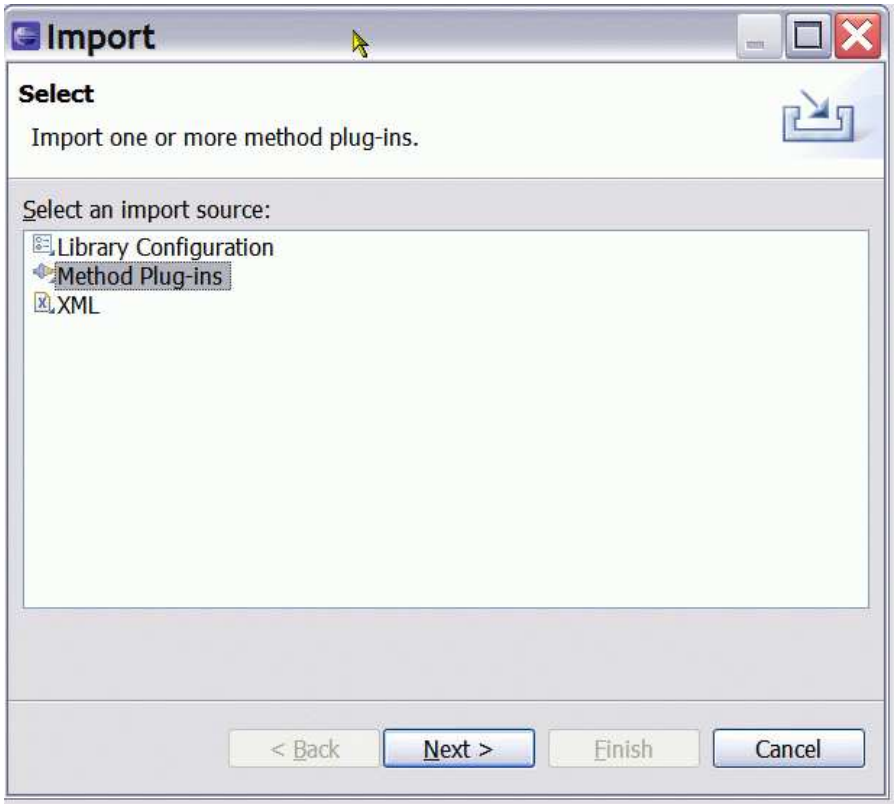
Click **Finish** and the Method Library is then created. You can now import the TOGAF9 Method Plug-in to this library.

Figure 13. Import Method Plug-in



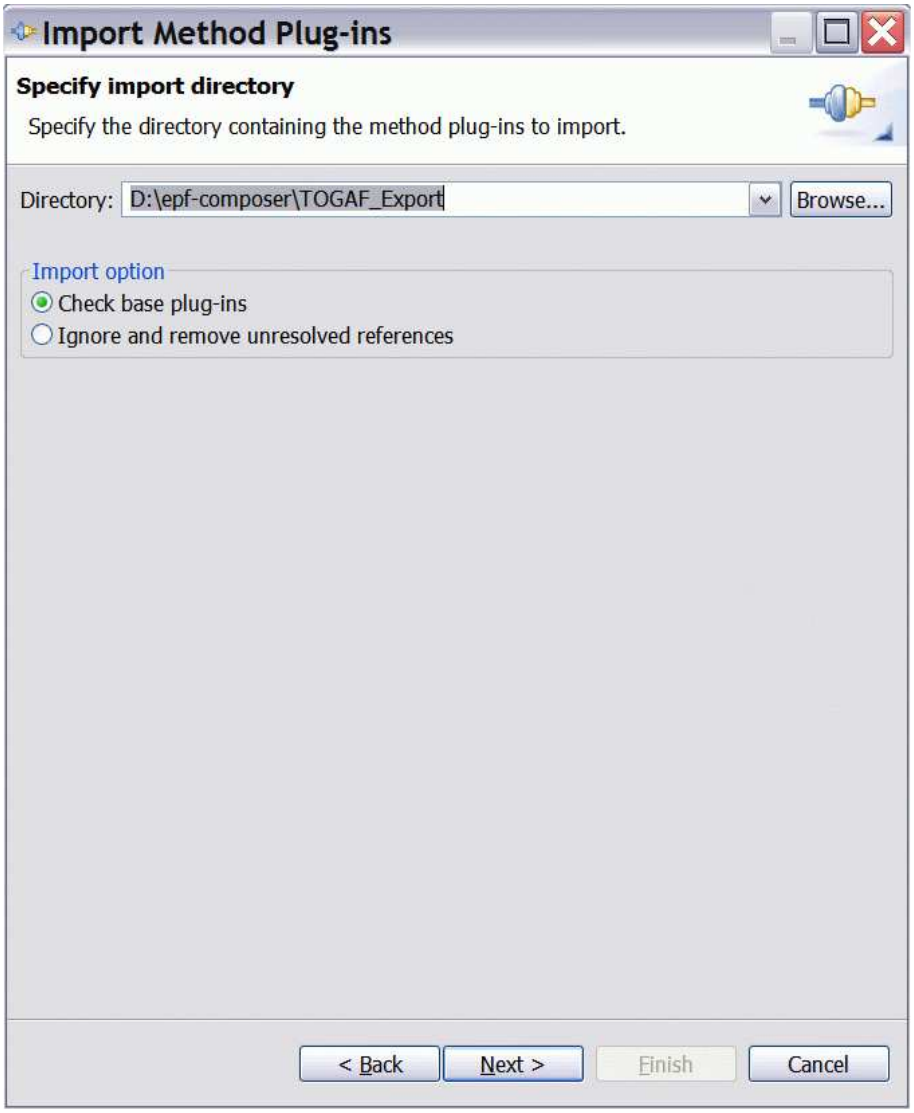
The result is the following window:

Figure 14. Import Source Type



You will have to navigate to the Method Plug-in folder.

Figure 15. Import Directory

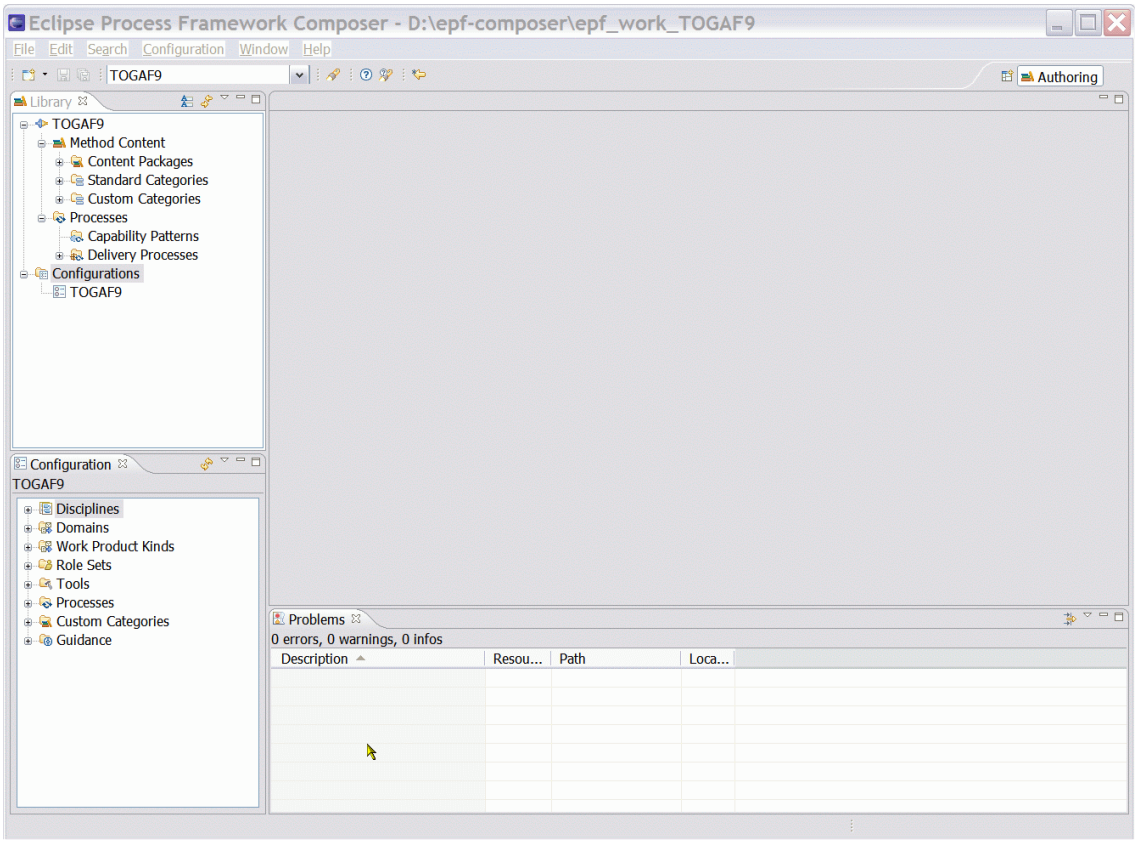


Having clicked on **Next** → then another screen appears:

Figure 16. Method plug-ins

You select TOGAF9 and click on Finish. You are in business. You have now an Enterprise Architecture library in your work folder "*epf_work_TOGAF9*".

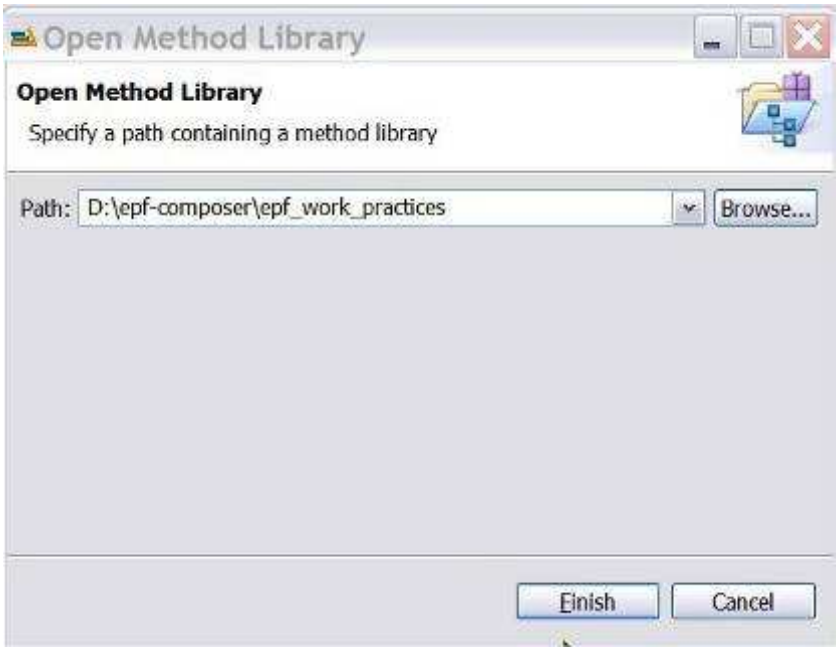
Figure 17. Import Directory



2.4. Switching Method Libraries

Start EPF and click on **File** → **Open** → **Method Library**. The Open Method Library dialogue box opens. Browse to select the folder with the library (or with the editable copy of the original library) files. In this case we have selected "*D:\epf-composer\epf_work_practices*" folder.

Figure 18. Open Method Library



Then, somewhat surprisingly, the Copy Library dialogue box pops up again:

Figure 19. Copy Library

Again, this message tells you, although not very clearly, that you have the choice between using the files in the "*epf_work_practices*" folder or creating a separate folder and copying the files over. In the latter case, you would have two copies. In our case, since we clicked on '**Copy**' when we installed the files the first time, the files in the "*epf_work_practices*" folder are already a copy of the files in the "*epf_practices*" folder. Since we do not need an additional copy of the copy of the original files, we therefore click on the **Skip** button to load the already existing method library in the default working folder "*epf_work_practices*".

There should be an option of turning this nagging feature/bug off. Most of the time, the user simply wants to open a method library and going through the hassle of this form should not be necessary. It would seem that simply copying a folder or files, using the operating system, offers the same functionality, would the user need it. Since most users will be more familiar with windows than with EPF Composer, the need for this annoying dialog box is not very clear.

3. Introduction

You will find material that can help you getting started on:

http://www.eclipse.org/epf/general/getting_started.php

Papers

- Who will benefit from EPF
- Introduction to OpenUP
- EPF Composer Overview - Part 1
- EPF Composer Overview - Part 2

Tutorials

- An Introduction to EPF
- Customisation scenarios with EPF Composer and OpenUP

Recorded Presentations and Demos

- General Overview
- Method Content Authoring
- What is new to EPF Composer 1.2 (Raindance Web site)
- EPF Wiki Demo Site: feel free to use the new demo resource and to share it with others

Presentations and publications

- EPF Composer 1.2 - New and Noteworthy
- EPF Short Tutorial at EclipseCon 2007
- OMG Process Modelling Special Interest Group (PM-SIG)
- What is EPF
- EPF: An Open Source Process Initiative
- Open Unified Process Distilled
- Increasing Development Knowledge with EPF Composer
- A Development Library At Your Fingertips
- Building embedded software with EPF
- OpenUP - The Best of Two Worlds

3.1. EPF Composer Overview

Welcome to the Eclipse Process Framework (EPF) Composer. The EPF Composer is a is a free, open-source tool platform for enterprise architects, programme managers, process engineers, project leads and project managers to implement, deploy, and maintain processes for organisations or individual projects.

Typically, **two key problems need to be** addressed to deploy new processes successfully.

1. First, teams need to be educated on the methods applicable to the roles for which they are responsible.
2. Second, teams need to understand how to apply these methods throughout a change management and development lifecycle. That is, they need to define or select a process and they need a clear understanding of how the different tasks relate to each other.

To that end, EPF Composer serves **two main purposes**:

- **First**, to provide a ***knowledge base of intellectual capital*** that you can browse, manage and deploy.

  EPF Composer is designed to be a content management system and publishing application that provides a common management structure and look and feel for all content, rather than being just a document management system in which you would store and access documents all in their own shapes and formats.

  All content managed in EPF Composer can be published as a Web site with method guidance and processes that can be deployed to Web servers for distributed usage by multiple teams.

  The content can include externally and internally developed content such as whitepapers, guidelines, templates, principles, best practices, internal procedures and regulations, training material, and any other general descriptions of the methods.

- **Second**, to provide ***process-engineering capabilities*** by supporting architects, process engineers, programme and project managers in selecting, tailoring, and rapidly assembling processes for their concrete projects.

  EPF Composer provides catalogues of pre-defined processes for typical project situations that can be adapted to individual needs. It also provides process building blocks, called capability patterns that represent best development practices for specific disciplines, technologies, or management styles. These building blocks form a toolkit for quick assembly of processes based on project-specific needs. EPF Composer also allows you to set up your own organisation-specific capability pattern libraries. Finally, the processes created with EPF Composer can be published and deployed as Web sites.

EPF Composer provides **key capabilities**:

- Creating processes with breakdown structure editors and workflow diagrams through use of multi-presentation process editors, different process views and synchronisation of all views with process changes.
- Supporting reusable dynamically linked process patterns of best practices for rapid process assembly via drag-and-drop.
- Select, combine, tailor, and rapidly assemble process configurations from method content for an organisation's development projects
- A common management structure, look and feel for all of an organisation's method content
- Authoring methods and processes with rich content such as text, images and multimedia, while remaining independent of the process architecture

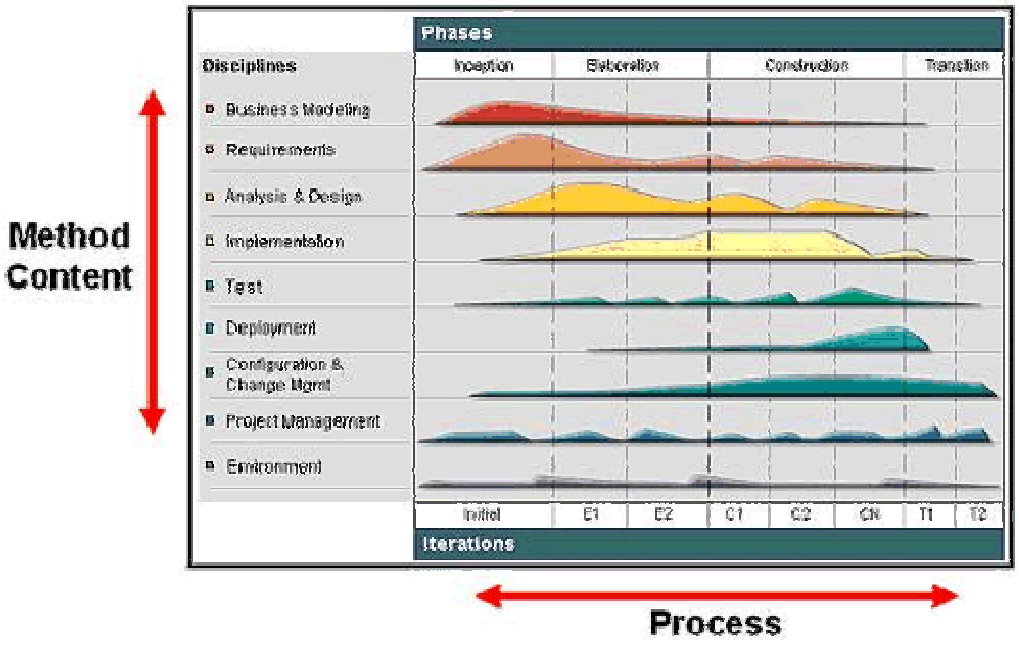
- Managing method content using simple form-based user interfaces and intuitive rich text editors for creating illustrative content descriptions.
- Supporting multiple views of method content through textual and graphical representation and editors allow use of styles, images, tables, hyperlinks, and direct html editing.
- Visually communicating a clear understanding of how the different tasks within a method relate to each other
- Providing just-in-time generation of publication previews in dedicated browsing perspectives that allows a consistent look-and-feel to published processes distributed through the organisation and rapid configuration switching
- Publishing process configurations using a number of platform-independent output formats, such as HTML and PDF
- Lowering the cost of reuse of method content across an organisation and providing rich extensibility capabilities.

Key terminology and concepts

To work effectively work the EPF Composer, you need to understand a few concepts that are used to organise the content. The pages Method Content Authoring Overview and Process Authoring Overview contain more detail and concrete examples of how to work with the tool. This page provides you with a general overview of these concepts.

The most fundamental principle in EPF Composer is the separation of reusable core method content from its application in processes. This directly relates back to the two purposes of EPF Composer described in the first section. Almost all of EPF Composer's concepts are categorised along this separation. **Method Content** describes what is to be produced; the necessary skills required and the step-by-step explanations describing how specific development goals are achieved. These method content descriptions are independent of a development lifecycle. **Processes** describe the development lifecycle. Processes take the method content elements and relate them into semi-ordered sequences that are customised to specific types of projects.

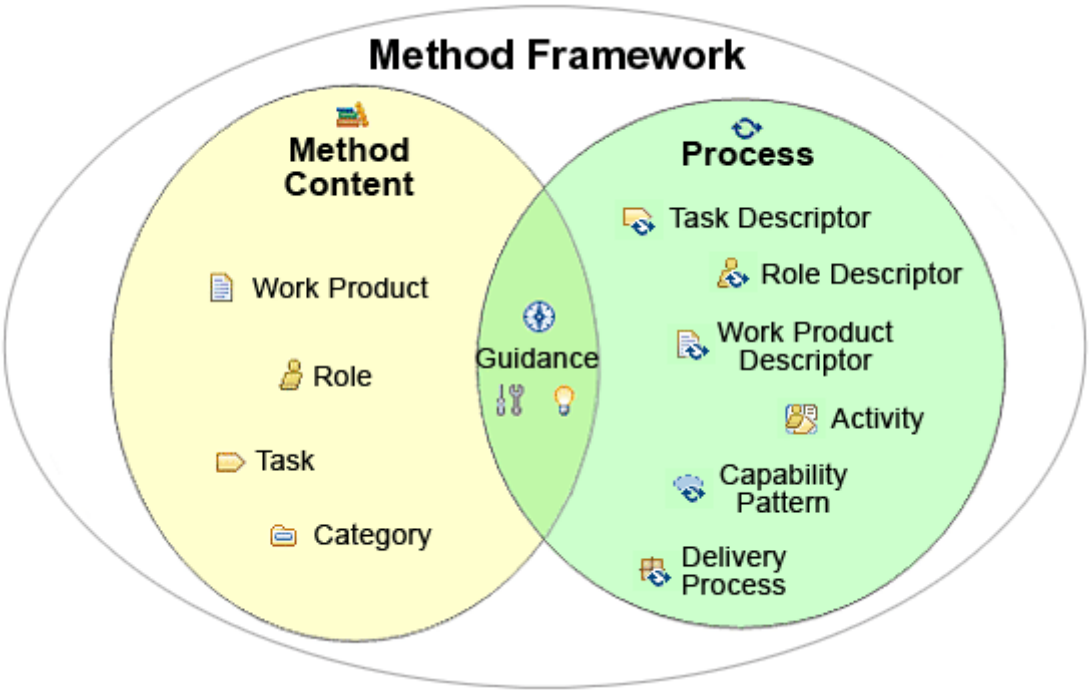
Figure 20. Unified Process



The figure above shows and example of how this separation is depicted in the methodology called Unified Process. The **Method Content**, describing how development work is being performed, is categorised by disciplines along the y-axis of the diagram. The work described in a **Process** is seen along the x-axis representing the timeline. This is the lifecycle of a development project. It expresses when work will be performed.

The graph in the illustration represents an estimated workload for each discipline. As you see, for example, one never stops working on requirements in Unified Process, but there are certainly peak times in which most of the requirements elicitation and description work is performed. There are also times at which a downward trend needs to be observed where fewer and fewer requirements changes have to be processed to end the project. This avoids what is referred to as feature creep in which requirements work remains constant or even increases. Hence, a lifecycle (process) expresses the variances of work performed in the various disciplines (method content).

Figure 21. EPF Method Framework



The picture above provides a summary of the key elements used in the EPF Composer their relationships with processes or method content. As you see, **method content** is primarily expressed using work products, roles, tasks, and guidance. Categories are required for publishing the process methodologies as a Web site. Guidance, such as checklists, examples, or roadmaps, can also be defined to provide exemplary walkthroughs of a process.

On the right-hand side of the diagram, you see the elements used to represent **processes** in EPF Composer. The main process element is the **activity** that can be nested to define breakdown structures as well as related to each other to define a flow of work. Activities also contain descriptors that reference method content. Activities are used to define processes of which EPF Composer support two main kinds: delivery processes and capability patterns.

Delivery processes represent a complete and integrated process template for performing one specific type of project. They describe a *complete end-to-end project lifecycle* and they are used as a reference for running projects with similar characteristics.

Capability patterns are processes that express and communicate *process knowledge for a key area of interest* such as a discipline or a best practice. They are also used as ***building blocks*** *to assemble delivery processes or larger capability patterns*. This ensures optimal reuse and application of their key best practices in process authoring activities in EPF Composer.

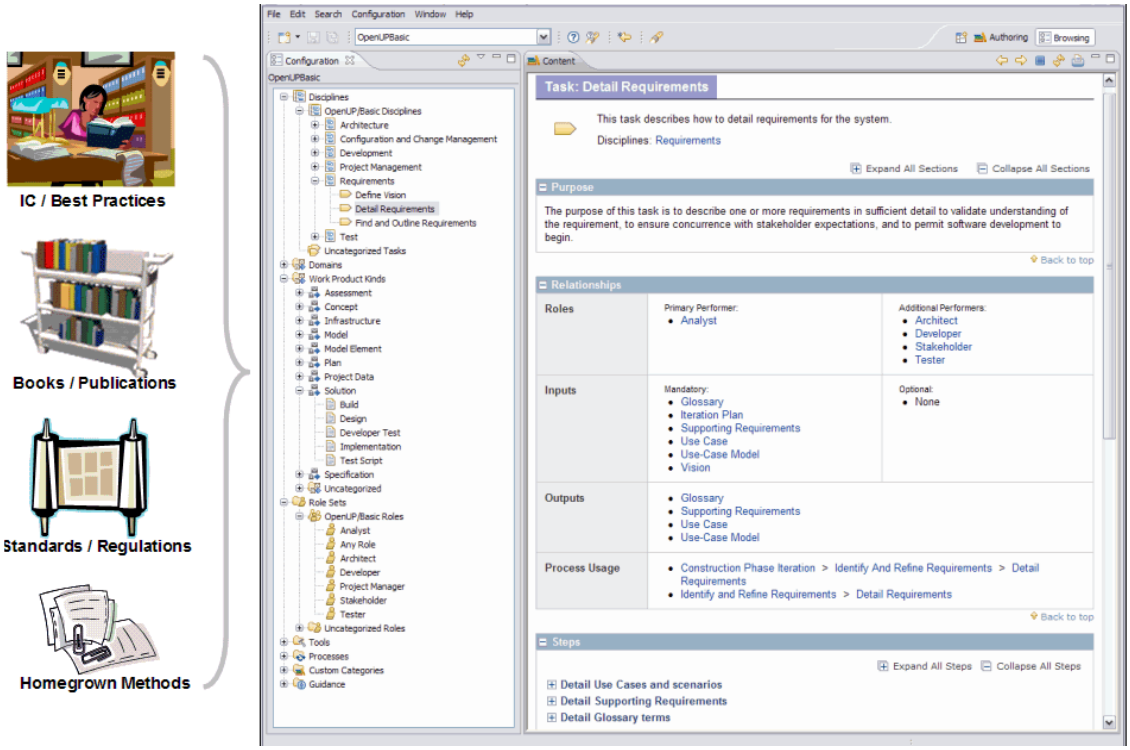
Related Topics

Method Content Authoring Overview
Process Authoring Overview

3.2. Method Content Authoring Overview

Method content describes **roles**, the **tasks** that they perform, the **work products** that are used and produced by those tasks, and supporting **guidance**.

Figure 22. EPF Composer – Method Content Representation



The figure above illustrates how method content is represented in EPF Composer. Many development methods are described in publications such as books, articles, training material, standards and regulations, and other forms of documentation. These sources usually document **methods** by providing step-by-step explanations for a particular way of achieving a specific development goal under general circumstances. Some examples are: transforming a requirements document into an analysis model; defining an architectural mechanism based on functional and non-functional requirements; creating a project plan for a development iteration; defining a quality assurance plan for functional requirements; redesigning a business organisation based on a new strategic direction, and so on.

EPF Composer takes content as described above, and structures it in a specific schema of roles, work products, tasks, and guidance. This schema supports the organisation of large amounts of descriptions for development methods and processes covering design and engineering disciplines such as enterprise architecture, business transformation, software development, mechanical engineering and so on.

The diagram above shows how such method content elements are organised in tree browsers on the left. These tree browsers provide different indexes of the available elements for rapid access. The screen capture shows on the right an example of a task presentation. This task presentation defines the task in terms of steps that need to be performed to achieve the task's purpose. You can see that the task has various relationships, such as relationships to performing roles as well as work products that serve as inputs and outputs to the task.

In addition to roles, tasks, and work products, EPF Composer supports the addition of guidance elements. **Guidance elements** are supplementary free-form documentation such as white papers, concept descriptions, guidelines, templates, examples, and so on.

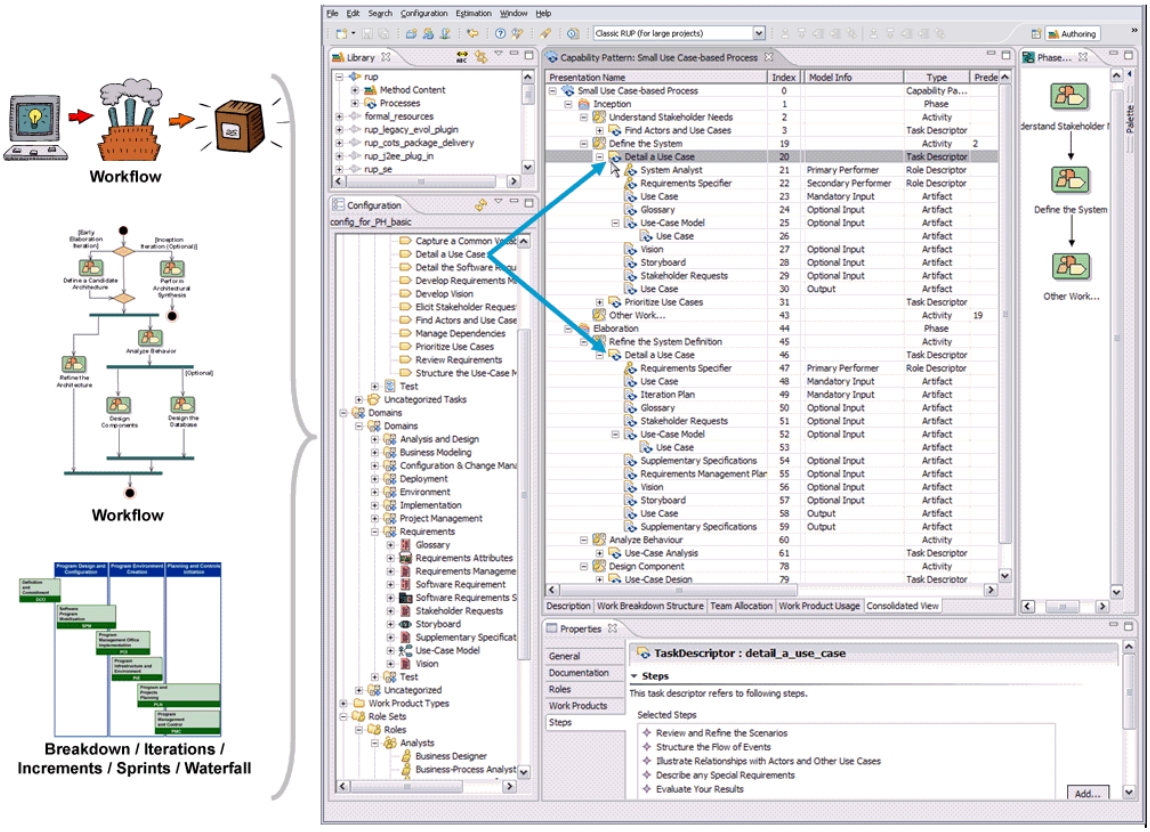
EPF Composer provides various form-based editors to create new method content elements. You can document your task, roles, work products, and guidance elements using intuitive rich-text editors that allow you to copy and paste text from other sources such as web pages or documents. You can also use simple dialogs to establish relationships between content elements.

EPF Composer organises content in **content packages** that allow you to manage your content in configurable units. EPF Composer also allows you to **categorise** your content based on a set of predefined categories (for example, categorise your tasks into development disciplines, or your work products into domains) or to create your own categorisation schemes for your content with your own user-defined categories that allow you to index content in any way you want.

3.3. Process Authoring Overview

A process defines sequences of tasks performed by roles and the work products produced over time.

Figure 23. EPF Composer – Process Authoring Representation



The figure above shows that processes are typically expressed as **workflows** or **breakdown structures**. Defining a strict sequence as in a waterfall model is as much a process as is defining *semi-ordered sequences in iterations of parallel work*. They just represent different development approaches. Hence, for defining a process, one can take method content and combine it into structures that specify how the work shall be organised over time, to meet the needs of a particular type of development project (such as business transformation versus software for a online system supporting the business transformation).

EPF Composer supports processes based on different development approaches across many different lifecycle models, including waterfall, incremental, and iterative life cycles. EPF Composer also supports different presentations for process, such as work-breakdown structure or workflow presentations. You can also define processes in EPF Composer that use a minimal set of method content to define processes for agile, self-organising teams.

The screen capture above shows an example of a process presented as a breakdown structure of nested activities as well as a workflow or activity diagram for one particular activity, the inception phase. It also indicates with the two blue arrows that the particular method content task "Detail a Use Case" has been applied in the process twice; firstly in the inception phase under the activity "Define the System," and secondly, in the elaboration phase in the activity "Refine the system definition". You see below each of these task applications, referred to as a task descriptors, lists of the performing roles as well as the input and output work products. If you look closely, you see that these lists are different for each of these two task descriptors, expressing differences in performing the "Detail a Use Case" method throughout the lifecycle.

You see different roles involved and changes in the list of inputs to be considered and outputs to be produced or updated. These changes were defined by the author that created this process to express the exact focus of the task performance for each occurrence. In addition to updating the roles, input and output work products for a task descriptor, you can also provide additional textual descriptions as well as define the exact steps of the task that should and should not be performed for this particular occurrence of the task.

EPF Composer provides you with a process editor that supports different breakdown structure views as well as graphical process presentations. As a process author, you typically start by creating an activity breakdown, dividing and breaking your process down into phases, iterations, and high-level activities. Instead of creating your activities in the breakdown structure editor, you can alternatively work in a graphical activity diagram editor that allows you to create a graphical workflow for your activities.

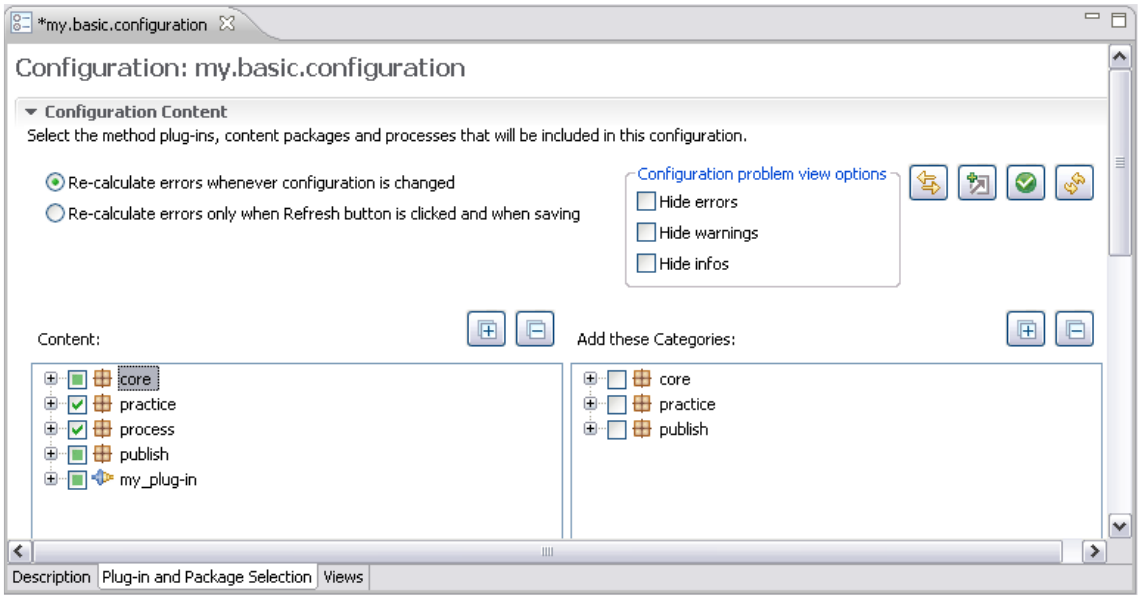
To assign method content to your process, you then have the choice of working in different process views (work breakdown structure, work product usage, or team allocation view). Each view supports a different approach for creating a process. You can define the work to be done, define the results to be produced, or define responsibilities for your roles. If requested, the editor updates the other process views semi-automatically using wizards that prompt you for decisions on selecting method content elements.

3.4. Method Configurations Overview

EPF Composer offers a library containing a great deal of reusable content. Its includes the OpenUp method framework and various plug-ins extending OpenUp with domain-specific additions such as development for concrete technologies such as J2EE or different development circumstances such as adopting a commercial off-the-shelf system (COTS). No organisation or project requires all of this documentation all at once, but would work with a selection of specific subsets.

EPF Composer manages for that purpose so-called **method configurations**, which allow you to specify working sets of content and processes for a specific context, such as a specific variant of the OpenUp framework that you want to publish and deploy for a given software project or as a foundation for a development organisation. All content and processes are organised in **method plug-ins** and ***method content packages***. A method configuration is simply a selection of the method plug-ins and packages.

Figure 24. EPF Composer – Configuration Editor



You create and specify a configuration using the configuration editor depicted in the figure above. You could start creating your own method configuration by copying one of the configurations included with EPF Composer and modify it to fit your specific needs. You can add or remove whole method plug-ins as well as select each plug-in by checking or un-checking packages.

You can use the resulting configuration as your working set for your EPF Composer work. The actual content of the configuration, i.e. the included method content and process elements are always accessible in the configuration view. Use the combo box in the toolbar to select the currently used method configuration.

Publishing overview

Method configurations are the basis for publishing method content and processes. A published configuration is an HTML web site that presents all the method content and processes of the method configuration in a navigable and searchable way. It uses the relationships established during method content and process authoring to generate hyper-links between elements as well as provides tree browsers based on the configuration view and user-defined categorisations of the content.

For publishing, simply create and select a configuration. The publication wizard will do the rest for you and only publish content that is part of the method configuration. It will also automatically adopt content to the configuration such as removing references of method content elements to elements outside of the configuration or removing activities from your processes that only contain work defined outside of the configuration set. Hence, publishing will only include the content that you really need. You can always preview a published configuration using the browsing perspective.

4. Tutorials

Contents

The tutorials can also found in EPF. In the Workbench, click on the **Help** menu in the main tool bar and then click on **Help Contents**. Expand Eclipse Process Framework (EPF) Composer and expand **Tutorials**.

4.1. Explore the EPF Composer Workbench

This tutorial contains a brief summary of key concepts followed by five exercises. Exercises in this tutorial are based on the Practices library available with EPF Composer.

Learning objectives

Upon completion of this tutorial you should be able to do the following:

- Use the buttons and menus that you will need for routine operations
- Use the two main perspectives to see different views of library content
- Drill down into a method library to see how library content is categorised
- Preview the resulting pages

Time required

The estimated time to complete this tutorial is about 45 minutes.

Lessons in this module

- Concepts

  If you are a new user of EPF Composer, then this tutorial is an appropriate starting point. You will explore the basic user-interface features and experiment with some simple browsing actions. This is the first in a series of tutorials about EPF Composer.

- Basic Navigation

  The goal of this exercise is to switch between perspectives and see the features they provide.

- Browse Method Content

  The goal of this exercise is to examine method content in a library using the Browsing perspective.

- Browse Process Content

  The goal of this exercise is to explore process content in a library using the Browsing perspective.

- Browse while Authoring

  The goal of this exercise is to browse some library content while in the Authoring perspective.

- Search

  The goal of this exercise is to use the search function to locate method content.

4.1.1. Concepts

If you are a new user of EPF Composer, then this tutorial is an appropriate starting point. You will explore the basic user-interface features and experiment with some simple browsing actions. This is the first in a series of tutorials about EPF Composer.

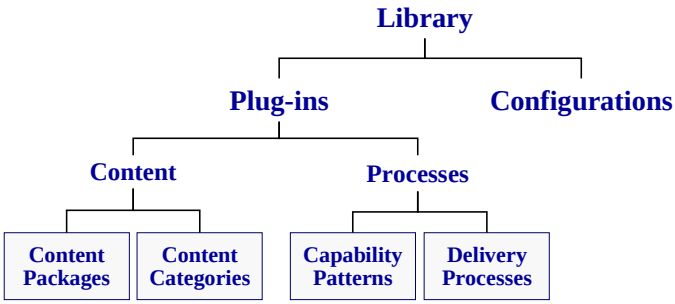
Method Library

The Method Library contains **Method Plug-ins** and **Method Configurations**.

Method Plug-ins

A method library is analogous to a warehouse full of parts that are used to assemble various products. Elements in a method library are organised by their intended function in the same way that similar parts are organised on a shelf in the warehouse.

All method content is organised in Method Plug-ins. The Method Plug-ins contain method Content and Processes.



Method Content consists of method Content Packages and Standard & Custom Categories. Method Content Packages contain four types of elements: Tasks, Roles, Work Products and Guidance. There are five Standard Categories: Disciplines, Domains, Work Product Kinds, Role Sets and Tools. The two types of **Processes** are Capability Patterns and Delivery Processes. Processes organise method content elements into semi-ordered sequences customised for specific types of projects.

Figure 25. Method Library Content

A method library is a repository of method elements used to generate process guidance based on a selected methodology. Process guidance is typically delivered in the form of a published Web site but can also be imported into other compatible applications.

Method Configurations

A method configuration is the manifest of method plug-ins used to generate a specific instance of process guidance. The term *configuration* is also used to refer to the process guidance that is generated by the manifest. Method configurations are built from subsets of elements in the method library. To follow our analogy, method configurations represent the various products, such as cars, that can be assembled from parts in the warehouse. While most cars require unique parts that are used by a specific model, but the warehouse has large numbers of parts that can be used in more than one model.

Library and Configuration Views

The **Library View** shows all of the parts in the warehouse, including those that are used by all products and those that are used by specific products. There might be parts that are not used in any of the products, such as obsolete or experimental parts. The **Configuration View** only shows the parts that are needed and used in a specific product, analogous to the specific list of parts that are used in a particular model of car.

After you select a configuration in the main menu, the Configuration View shows only the library elements that are used in that configuration. Content in the Configuration View is always organised by using the same set of folders, regardless of which configuration is selected.

The Library View shows the complete list of all Method Plug-Ins, together with Method Configurations. This simple and flat list of all Method Content Packages can be long

and unwieldy and the library view offers a hierarchical view of Method Plug-ins grouped together in logical packages, or sets, of Method Plug-ins. Using dots in the name of the plug-ins creates the logical groupings of method plug-ins displayed in hierarchical presentations.

Perspectives

Perspectives are similar to modules, but they are used for different high-level operations. The two main perspectives that are commonly used in routine operations are the Authoring and Browsing perspectives.

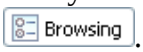
- The **Authoring Perspective** is designed for content editing and configuration operations. You can create and maintain method content by using the editors in this perspective that are optimised for particular content types. As you create and modify content, you will use this perspective most of the time. There is a preview option in this perspective so that you can see the resulting page but without resolving all the contributing elements that make up that page.
- The **Browsing Perspective** provides a more complete preview of the generated configuration, but you cannot modify it. You might want to use this perspective to verify new or edited configurations before publishing them. The Browsing perspective displays all contributing components, but does not show the tree Browser that will be automatically built and included in the configuration when it is published.

4.1.2. Basic Navigation

The goal of this exercise is to switch between perspectives and see the features they provide.

To navigate in the authoring and browsing perspectives:

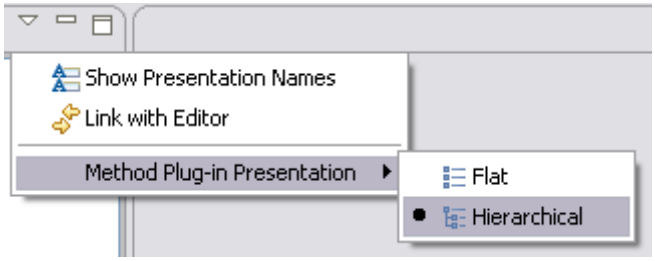
1. Open the *epf-practices library* by clicking **File** → **Open** → **Method Library**, then browse to select the library folder and click **Finish**.
2. The current perspective is displayed in the upper right-hand corner of the main toolbar. The two perspectives you will use most often are **Authoring** 

and **Browsing** 

. Click on the Open Perspective icon  and note the other perspectives available through this menu.
3. You can also change perspectives by clicking on the » symbol to the right of the perspective indicator. Try using this control to switch back and forth between perspectives.
4. Switch to the **Authoring** perspective if you're not already in it. In the left-hand side of the main window you should see two panels with tree-navigators, one called **Library** and the other called **Configuration**. If the Library panel isn't visible you should use the **Window** menu in the main toolbar and select **Show View**, and then select **Library**. This control lets you show or hide the view panels.
5. Explore the **Library** view. Note the Library view is only available while you are using the Authoring perspective. It is not accessible while you are in the Browsing perspective. The Library view shows you all method content in the current library. The highest-level library content unit is a **Method Plug-in** 

when you are using the **Flat** presentation. When you are using the **Hierarchical** presentation, the highest-level content unit is the logical grouping or set of Method Plug-ins, called **logical package** . The logical packages are sets of plug-ins created simply by using dots in their names. The structure of the names and dots of the plug-ins creates the logical groupings of the method plug-ins that are displayed in hierarchical presentations.

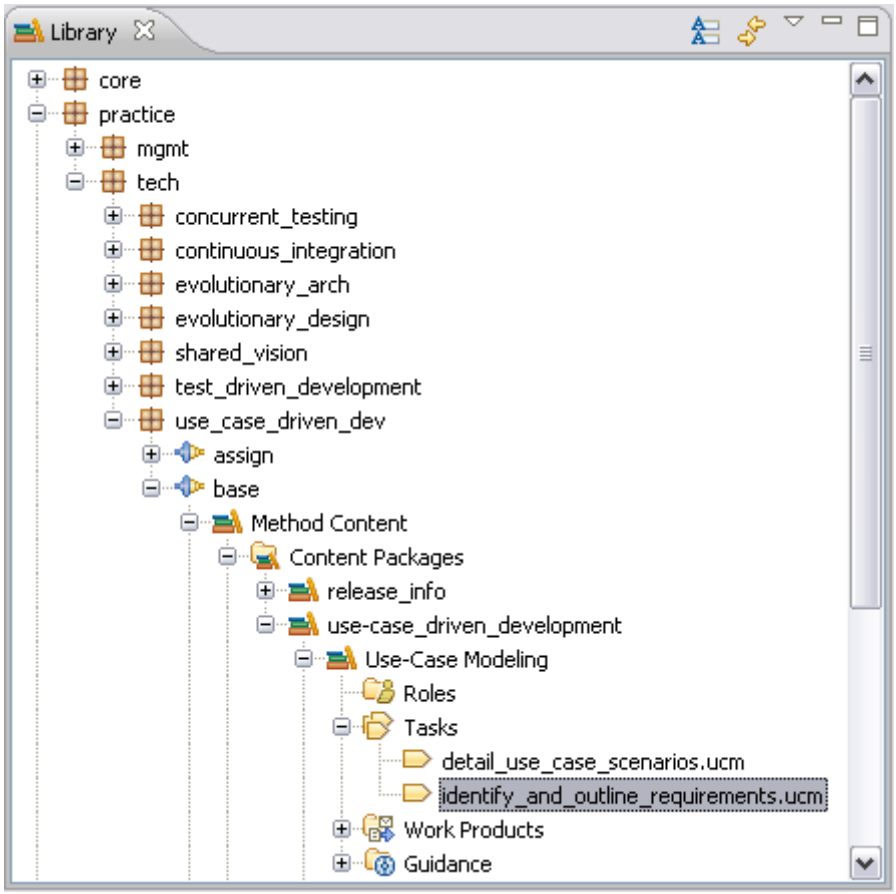
To switch between Flat and Hierarchical presentations, click the down arrow ▾ in the Library tool bar and select **Method Plug-in Presentation**, then choose **Flat** or **Hierarchical**.



Click on some of the + symbols and explore content in the plug-ins. Note the + symbols become - symbols. Expanded tree nodes can be collapsed by clicking on the - symbols.

6. Drill down into a practice's base plug-in as follows: *expand practice* → *tech* → *use_case_driven_dev* → *base* → **Method Content** → **Content Packages** → *use-case_driven_development* → *Use-Case Modelling* → **Tasks**. The Library view panel should look like this[1]:

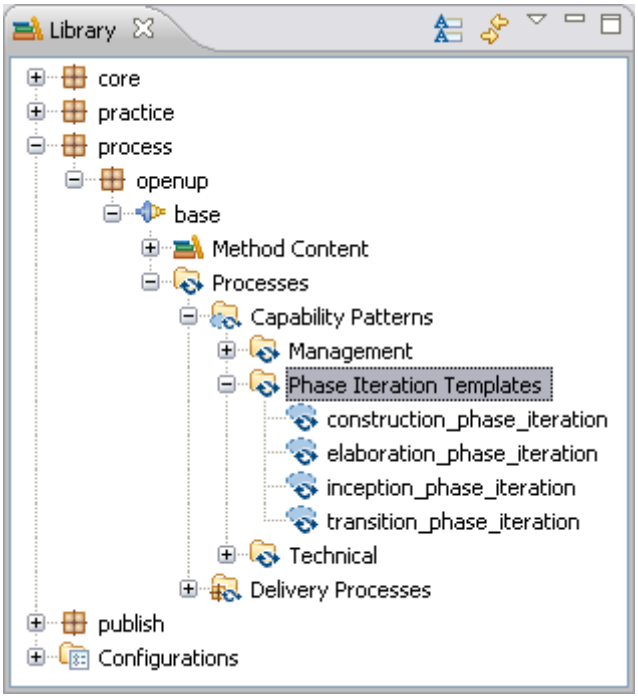
[1] The view and the content may change somewhat according to revisions.



Note that individual **tasks** are represented by this ***icon***: 

7. Double-click on one of the tasks listed there. The **task editor** is displayed on the right of the screen. There are separate editors for the different types of content that you can create in a library. Each editor has a series of tabs and a number of form elements on each tab. You can find out more about these in the Create Method Content tutorial.
8. Click the **Preview** tab. A preview of an HTML page is displayed. This is what the selected element will look like in a published Web site.
9. Use the configuration selection menu to select the ***publish.openup*** configuration.

10. Switch to the **Browsing** perspective. The Configuration view is now displayed. ***The Configuration view always has the same structure but the content changes based on the configuration currently selected***.
11. Expand the **Disciplines** folder until you get to tasks and click on a task. The HTML preview of the selected item is shown on the right of the screen.
12. Switch configurations. The current configuration is displayed in the selection box below the main menu bar. Select a configuration from the drop down list. You will see the Configuration view refresh when you do this.
13. Switch back to the Authoring perspective. Expand the *process* → *openup* → *base* plug-in, then the Processes folder, then the Capability Patterns folder, then *Phase Iteration Templates*.



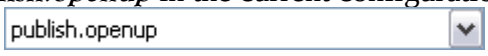
Capability patterns are represented by this icon:

Note that not all plug-ins have capability patterns. Double-click on one of the capability patterns. The panel on the right side of the window shows the capability pattern editor and the properties view. You can find out more about these in the Working with Processes tutorial.

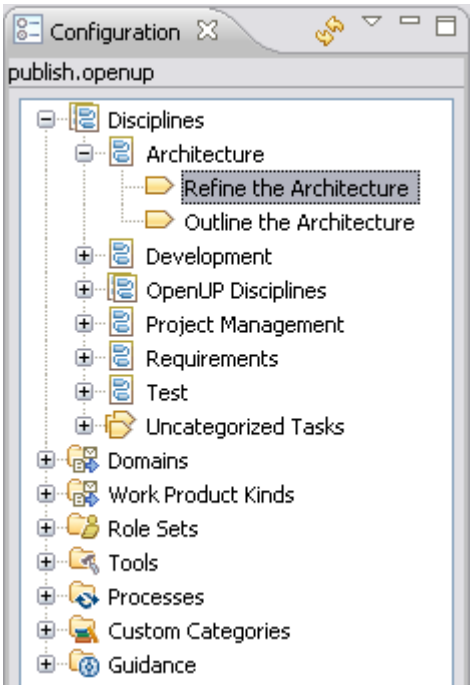
4.1.3. Browse Method Content

The goal of this exercise is to examine method content in a library using the Browsing perspective.

To browse method content in the browsing perspective:

1. Select *publish.openup* in the current configuration drop-down list in the main menu bar. 

2. Select the **Browsing** perspective. The **Configuration** view is displayed in the left-hand panel.
3. To view a task, expand the tree nodes in the Configuration panel by clicking on the + symbols. Expand **Disciplines** → Architecture. The **Configuration** view panel should look like this:



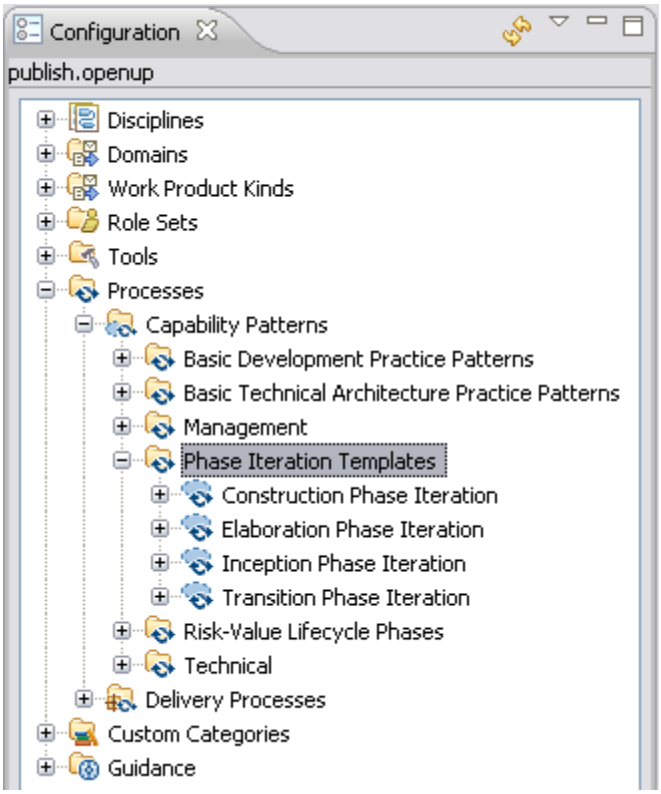
4. Click the Refine the Architecture task. A preview of the published Web page for the selected element is displayed in the content panel.
5. On the published Web page, click the link under **Primary Performer** in the **Relationships** section. This will show the preview page for the Architect.
6. You can click on any links in the preview and use the back icon in the preview toolbar to return to previous pages.
7. Use this technique to view other elements in the Configuration view. Explore elements under Domains, Role Sets, etc.

4.1.4. Browse Process Content

The goal of this exercise is to explore process content in a library using the Browsing perspective.

To browse process content in the browsing perspective:

1. Switch to the **Browsing** perspective.
2. To view a **Capability Pattern**, in the **Configuration**, expand the tree node labelled **Processes**, then **Capability Patterns**, then Phase Iteration Templates. The Configuration view panel should look like this:



3. Click Construction Phase Iteration. A preview of the published page for the selected capability pattern is displayed in the right panel.
4. A capability pattern contains a large amount of information and is displayed over four tabs:
   - Description
   - Work Breakdown Structure
   - Team Allocation
   - Work Product Usage

   Explore the information on each tab.

   Note: When you click on a task in the Work Breakdown Structure tab you preview a Task Descriptor. **A task descriptor is a task within a process**. ***The task descriptor page has information about the task as it is used at a specific point in a process***. The task descriptor has a link to the task on which it is based.
5. Clicking on a role or work product in a task descriptor takes you to a **Role Descriptor** or a **Work Product Descriptor**. These provide information about the role or work product at the same point in the process and provide links back to the core method element on which they are based.
6. To view a Delivery Process, in the Configuration view, expand the tree node for Processes, then Delivery Processes, then OpenUP Lifecycle. The display layout for a delivery process is similar to a capability pattern.

4.1.5. Browse While Authoring

The goal of this exercise is to browse some library content while in the **Authoring** perspective.

To browse in the authoring perspective:

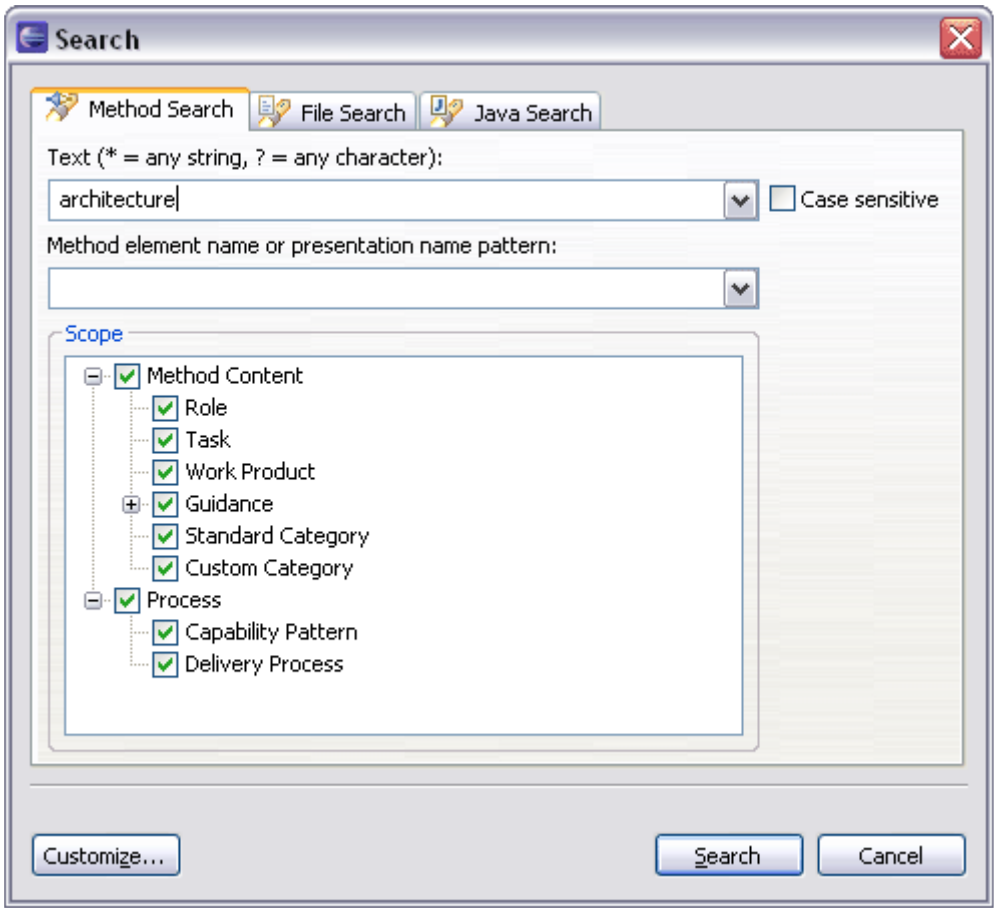
1. Switch to the **Authoring** perspective.
2. In the **Library** panel, expand *core* → *default* → *role_def* → *base* → **Method Content** → **Content Packages** → *basic_roles* → Roles and double-click the *architect* role. Detailed information about this is displayed in the right-hand panel.
3. Use the **Preview** tab at the bottom of the role window to view the page describing that role and its relationships to other method elements. Use the other tabs to see the information presented when authoring the method content and relationships for a role.

4.1.6. Search

The goal of this exercise is to use the search function to locate method content.

To search content:

1. Switch to the **Authoring** perspective if it is not already selected.
2. Begin the search by doing either of the following:
   - Click the Search icon in the toolbar.
   - Click the Search menu in the main tool bar and select **Search**.
3. Click the **Method Search** tab. Enter "*architecture*" in the text box and click **Search**.



It will take a few seconds to find all occurrences. Results will be displayed in a new panel labelled Search.

4. Expand the search results and double-click an element. The editor for the selected item is opened. If you have **Link with Editor** selected in the **Library** view, the display will update to show you where the selected item is located in the library. If the Link with Editor is not selected, you can right-click an element in the Search view and select **Show In → Library**.
5. Open the Search window again and repeat the same search but this time reduce the scope to look for **Task** elements only. Clear all the other content types in the scope panel. This time the search should be quicker and result in fewer hits.
6. Open the Search window again and repeat the same search but this time use a wildcard like "*arch*"as a file name pattern. Examine the results set.
7. Experiment with other method searches.

4.2. Create Method Content

This tutorial contains a brief summary of key concepts followed by ten exercises.

In this tutorial you will create or modify content elements that will be used in subsequent tutorials. For this reason we suggest that you complete all steps and use the exact names and text strings as described.

Learning objectives

Upon completion of this tutorial you should be able to do the following:

- Create a new method plug-in
- Create a new content package
- Create a new work product
- Create a new role
- Create a new task
- Work with steps
- Create basic guidance
- Add a method plug-in to a method configuration

Time required

The estimated time to complete this tutorial is about 90 minutes.

Prerequisites

You should make a back up copy of your library before continuing. In the following exercises we will make minor changes to the library content that you may not want to preserve.

Lessons in this module

- Concepts

This page describes the basic concepts that are required to complete this tutorial.

- Create a Method Plug-in

  The goal of this exercise is to create a new method plug-in.

- Create a Content Package

  The goal of this exercise is to create a method content package for our new plug-in.

- Create a Work Product

  The goal of this exercise is to create a new work product.

- Create a Role

  The goal of this exercise is to create a new role in my_plug-in.

- Create a Task

  The goal of this exercise is to create a new task and then relate it to the other elements that we created in previous exercises.

- Work with Steps

  The goal of this exercise is to use the Step Editor, which is part of the Task Editor.

- Create Guidance Elements

  The goal of this exercise is to create a guidance element.

- Apply Guidance

  The goal of this exercise is to perform some simple management tasks, specifically adding and removing guidance.

- Create a Standard Category

  The goal of this exercise is to create a new standard category.

- Add a Method Plug-in to a Configuration

  The goal of this exercise is to include the method plug-in created earlier to a copy of an existing configuration.

4.2.1. Concepts

This page describes the basic concepts that are required to complete this tutorial.

Method content provides step-by-step explanations describing how specific development goals are achieved independent of the placement of these steps within a development lifecycle. Method content is always separated from its application in processes.

There are four main types of method content elements:

- **Task**: how to perform the work;
- **Role**: who performs the work;
- **Work Product**: what is produced, such as artefact, deliverable or outcome;
- **Guidance**, such as: checklist, concept, example, guideline, estimation consideration, practice, report, reusable asset, roadmap, supporting material, template, term definition, tool mentor or white paper

These "basic elements" are the building blocks from which processes are composed. A Process Engineer typically authors these elements, categorises them, and defines the relationships between them. Processes are then constructed from these method elements by organising them into semi-ordered sequences, (work breakdowns or workflows) that are customised to specific types of projects.

Plug-ins: All content is organised in method plug-ins. A method plug-in is a container for method content packages that contains both method and process content. A method plug-in can be standalone or can reference other plug-ins. When you create a method plug-in, you can **reference** other plug-ins and reuse their content by modifying or extending the content or adding your own content. Referencing other extensible method plug-ins is one of the most powerful features provided by EPF Composer. This mechanism allows extensive customisation of already generated content without directly changing the original base content.[2]

You should always create new content in a method plug-in that you produce. This separates your content from the original content and allows you to update your own library with new releases of base content without affecting the content that you have created in your own plug-ins.

Standard categories provide a means to categorise core method content in line with best practices for creating structured methods. To encourage good method structure, there are standard categories for grouping tasks into **disciplines**, work products into **domains**, roles into **role sets**, and tool-mentors into **tools**. Unlike custom categories, standard categories, by definition, are linked to a specific type of method content.

Method and process elements use two names: **Name** and **Presentation name**. It is good practice to use file names that are all lowercase with no spaces and no special characters. The **Name** will be used as the name of the file that stores the element. We do this so that the name is valid on all operating systems and for integration with version control software. The name is shown in the Library view.

The presentation name is the name that is shown on published pages and in the configuration view. This name can contain any uppercase characters, spaces, and special symbols such as the trademark symbol.

In the Library view you can choose to display names or presentation names. You can switch the name display using the Show Presentation Names button  in the library view toolbar.

4.2.2. Create a Method Plug-in

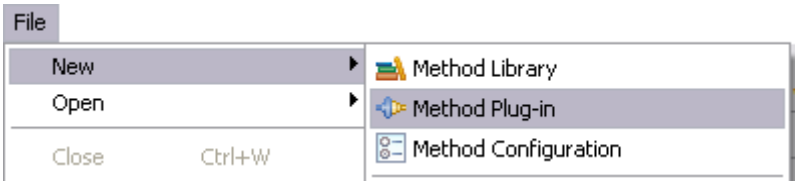
The goal of this exercise is to create a new method plug-in.

To create a method plug-in:

1. Make sure that you are in the **Authoring** perspective. 

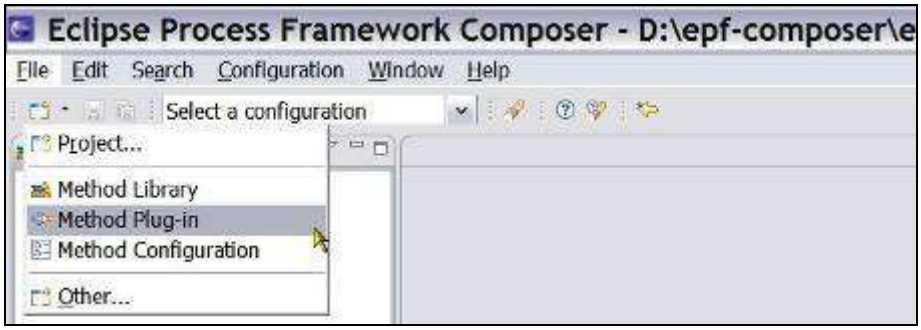
2. There are three ways to create a new method plug-in. Use one of the following:
   a. Click **File** → **New** → **Method Plug-in**.

[2] The base method content included with EPF Composer is protected from direct modification. Read-only method plug-ins are dimmed in the Library view indicating that they are locked.



b. Right-click in the Library view and select **New Method Plug-in**.

c. Click the Down Arrow of the '**New**' icon in the toolbar and select **Method Plug-in** from the drop-down list.

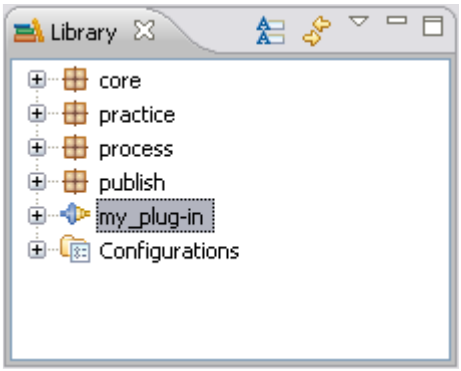


3. In the New Method Plug-in wizard, provide a name for the new method plug-in. In this instance, we will call it "*my_plug-in*".

   Note: We will refer to this method plug-in by name in many of the following exercises, so be sure to enter it exactly as shown.

4. Enter text into the Brief description field. It is always a good idea to add brief descriptions to new method elements as you create them.

5. In the **Referenced Plug-ins** panel, select *publish.openup.base* from the list and click **Finish**. This indicates that your plug-in will be an **extension** to the OpenUp plug-in. Unified Process provides a repository of base content that you can supplement with your own content. The new method plug-in is now in the Library view panel on the left side of your screen. It is not dimmed indicating that it is not locked and that you can edit it.



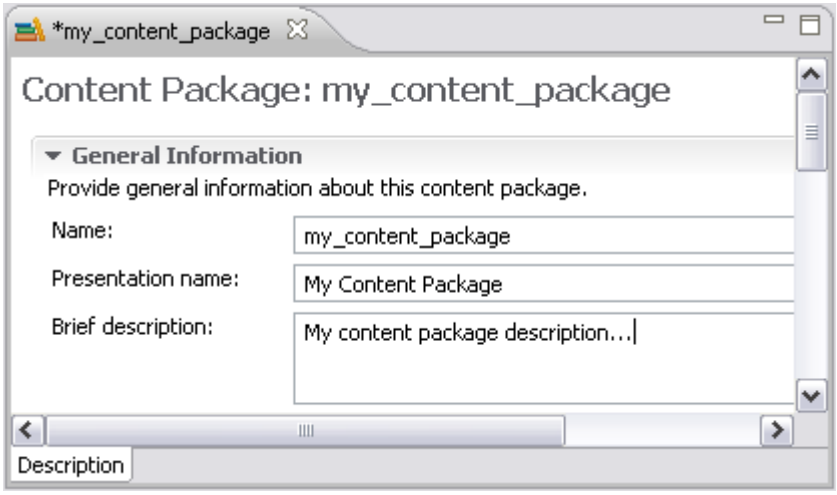
6. To open the plug-in editor, double-click *my_plug-in* (actually once you click Finish on the wizard, the editor opens automatically). About halfway down the page you will see the **Lock Plug-in** checkbox. This is unchecked for your new plug-in.

4.2.3. Create a Content Package

The goal of this exercise is to create a method content package for our new plug-in.

To create a content package:

1. In the **Library** view panel, expand the tree node for *my_plug-in* by clicking the + symbol, and then expand the node for **Method Content**.
2. Right-click **Content Packages** and select **New → Content Package** to create a new content package. The content package editor opens.
3. **Name** the package "*my_content_package*" with a **Presentation name** "*My Content Package*" and enter some text in the **Brief description** field.



Important: You cannot create a new content package or any other element in a **locked plug-in**.

4. Save your new content package. There are four ways to save a package or an element:
   - Close the editor and confirm you want to save
   - Click the disk icon in the toolbar
   - Use the shortcut "***ctrl+s***"
   - Click **File → Save**

   Remember: When you create a new element or modify an existing element, a ***** symbol is displayed in the tab next to the name of the element, indicating that the element needs to be saved.

   Tip: EPF Composer automatically creates nodes for Tasks, Roles, Work Products and Guidance under the new content package.

4.2.4. Create a Work Product

The goal of this exercise is to create a new work product. In this exercise you will create a type of work product called an artefact[3]. You can create any content but some

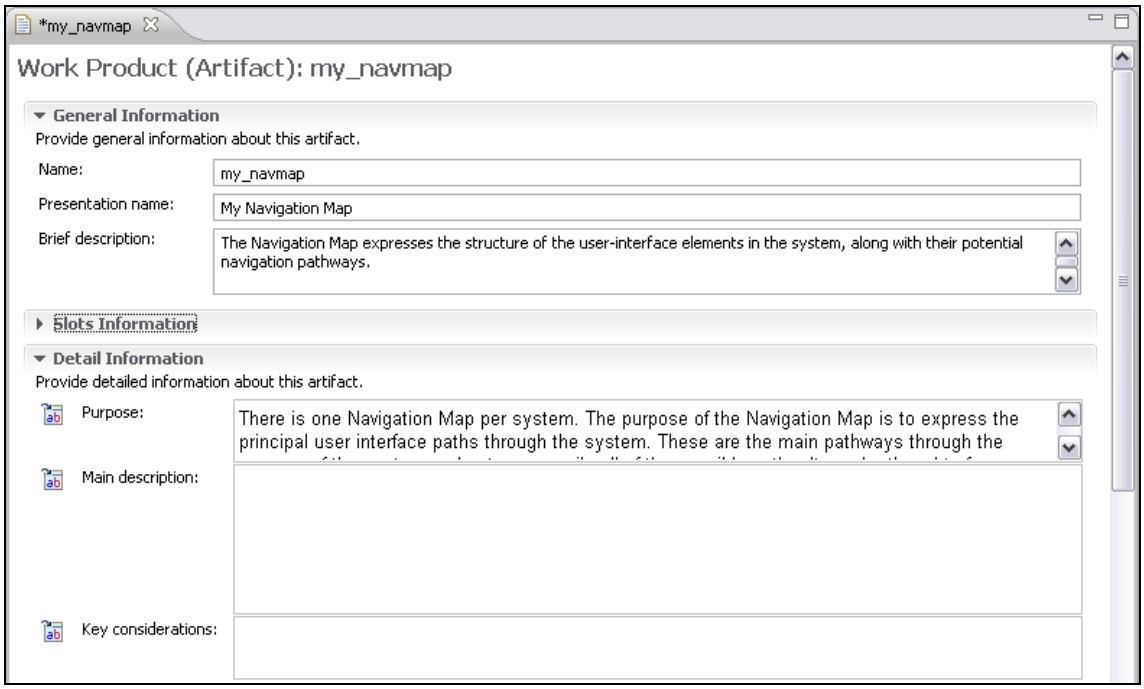
[3] There are three types of work products: artefacts, outcomes and deliverables. An **artefact** is a tangible work product that is consumed, produced, or modified by one or more tasks. Artefacts may be composed of other artefacts. An **outcome** is an intangible work product that may be a result or state. It may also be used to describe work products that are not formally defined. A **deliverable** is a collection of work products, usually artefacts, used to define typical or recommended content in the form of work products packaged for delivery.

examples are provided if you want to use them. These examples are referred to in other tutorials.

In all cases, new method elements are created by right-clicking on the destination folder for the new element and selecting **New**. For example, if you wanted to create a new role in a content package, you would right-click the Roles folder in the tree navigator and select **New** → **Role**. In the current case, you will create a Work Product.

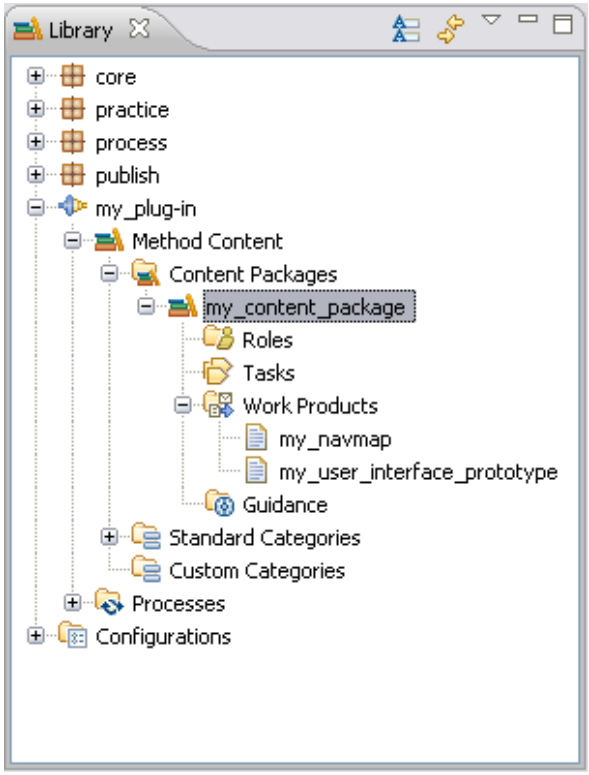
To create a work product under your content package:

1. Right-click **Work Products** and select **New**, then select a work product type such as Artefact.
2. The work product editor is displayed. Enter the following information:
   a. In the **Name** field, enter "*my_navmap*".
   b. In the **Presentation name** field, enter "*My Navigation Map*".
   c. Enter a brief description about this work product in the Brief Description field.
   d. Under the Detail Information tab, enter a purpose for this work product in the Purpose field.
3. The new work product should look similar to this:



4. Click the **Preview** tab to view the new work product.
5. Save the new artefact.
6. Now create another artefact. Repeat the above steps using the following information:
   - **Artefact Name**: "*my_user_interface_prototype*"
   - **Presentation name**: "*My UI Prototype*"
   - **Brief description**: "*A user-interface prototype is an example of the user interface that is built to explore or validate the user-interface design*".
   - **Purpose**: "*To provide a realistic experience of the look and feel of a user interface*".

You should now have two new work products in *my_content_package*:



4.2.5. Create a Role

The goal of this exercise is to create a new role in *my_plug-in*.

To create a role:

1. Under *my_content_package* right-click **Roles** and select **New** → **Role**.
2. The role editor is displayed. Enter the following information:
    3. In the **Name** field, enter "*my_user_interface_designer*".
    4. In the **Presentation name** field, enter "*My UI Designer*".
    5. Enter a description in the **Brief description** field.
6. At the bottom of the window, click the **Work Products** tab.
7. You can now specify work products for which this role is responsible. Click **Add**. This opens a selection list from which you can select work products to add to your role. Make this new role responsible for the two work products that we just created, *my_navmap* and *my_user_interface_prototype*.
8. After selecting the work products, click **OK**. The work products that you selected are in the **Responsible for** panel.

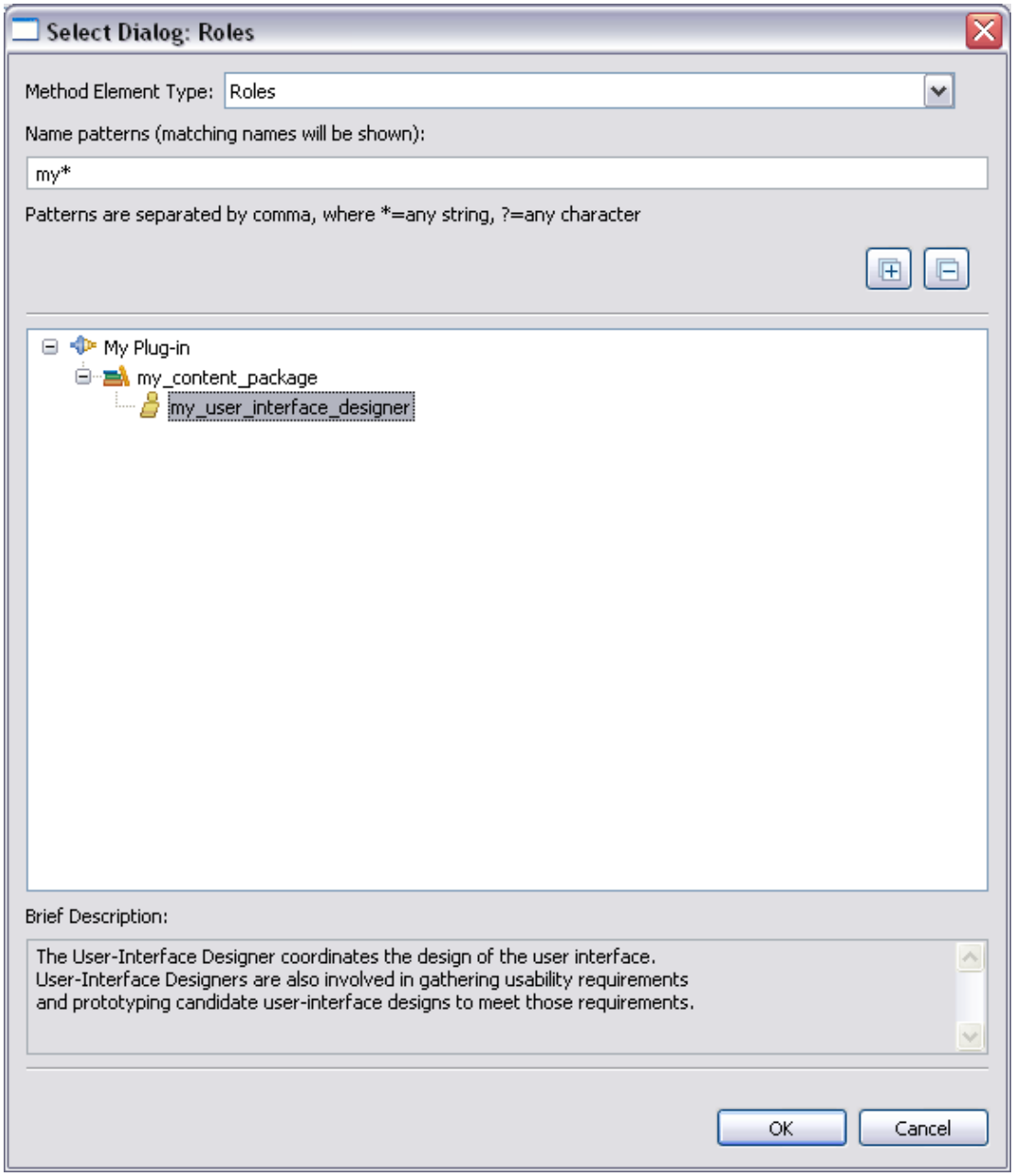
   Note: Selected elements in an Add/Remove section display the element name and the path to that element, the plug-in name, and the package name.
9. Click the **Preview** tab to view the newly created role.
10. Save your work.

4.2.6. Create a Task

The goal of this exercise is to create a new task and then relate it to the other elements that we created in previous exercises.

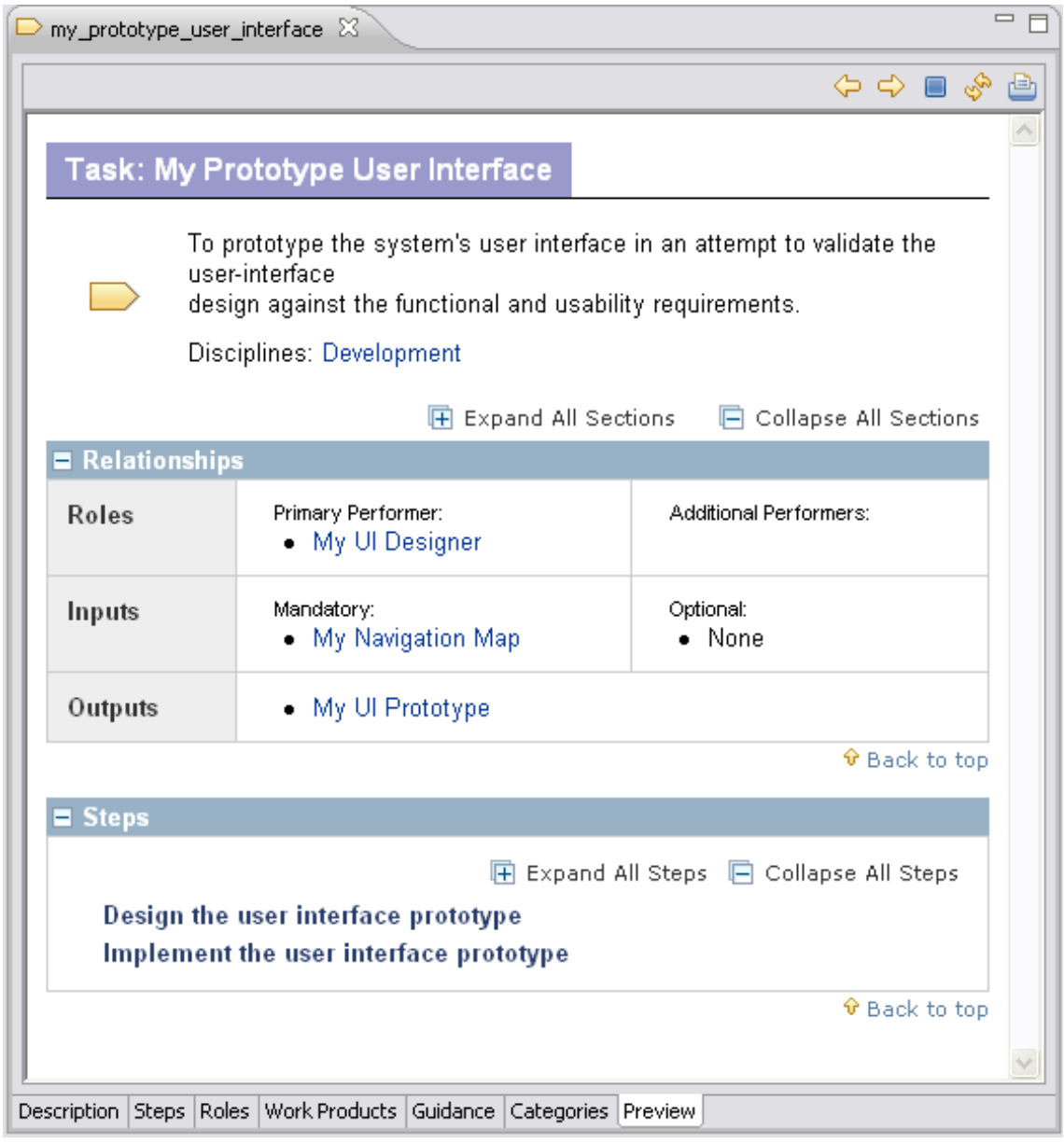
To create a task:

1. In the **Library** view under *my_content_package* right-click **Tasks** and select **New** → **Task**.
2. The **Description** tab is selected in the task editor. Use the following attributes to create the new task:
   - **Name**: "*my_design_user_interface*"
   - **Presentation name**: "*My Design User Interface*"
   - **Brief description**: "*To produce a design of the user interface that can be used to validate the new layout*".
3. Click the **Steps** tab and click **Add**. A new step named New Step is created. Change the name of this step to "*Describe the characteristics of related users*".
4. Click **Add** again and add a second step named "*Identify the primary user interface elements*". Note that you could add additional details about these steps in the Description field.
5. Click the **Roles** tab and in the **Primary performers** field click **Add**. The Select Dialog: Roles window opens. From this list select *my_user_interface_designer* as the Primary Performer. The complete list of roles is long but you can shorten it by entering "*my**" in the Name patterns field. Click **OK**.



6. Click the **Work Products** tab and in the **Mandatory inputs** section, click **Add**. The Select Dialog: Work Products window opens. In the plug-in: ***core.tech.common.extend_sup expand technical_work_products***, select ***use_case_model*** and click **OK**.
7. You should still be on the Work Products panel. In the **Outputs** section, click **Add**. Enter "***my****" in the name pattern field. Select ***my_navmap*** and click **OK**.
8. Click the **Guidance** tab and click **Add**. The Select Dialog: Guidance window opens. Under Visual Modelling Guidance, select **using_visual_modeling** and click OK.
9. Click the **Categories** tab and click **Add** to add a discipline. Select ***development_discipline*** and click **OK**.
10. Click the **Preview** tab to view the newly created task, and then save your work by closing the task editor.

11. Create another task using the following information:
    - **Name:** "*my_prototype_user_interface*"
    - **Presentation name:** "*My Prototype User Interface*"
    - **Brief Description:** "*To prototype the system's user interface in an attempt to validate the user-interface design against the functional and usability requirements*".
    - **Steps:** "*Design the user interface prototype*"
    - **Steps:** "*Implement the user interface prototype*"
    - **Roles (Primary Performer):** "*my_user_interface_designer*"
    - **Work Products (Mandatory Input):** "*my_navmap*"
    - **Work Products (Outputs):** "*my_user_interface_prototype*"
    - **Categories (Discipline):** "*development_discipline*"
12. Click the **Preview** tab to view the newly created task:



13. Save your work by closing the task editor panel.

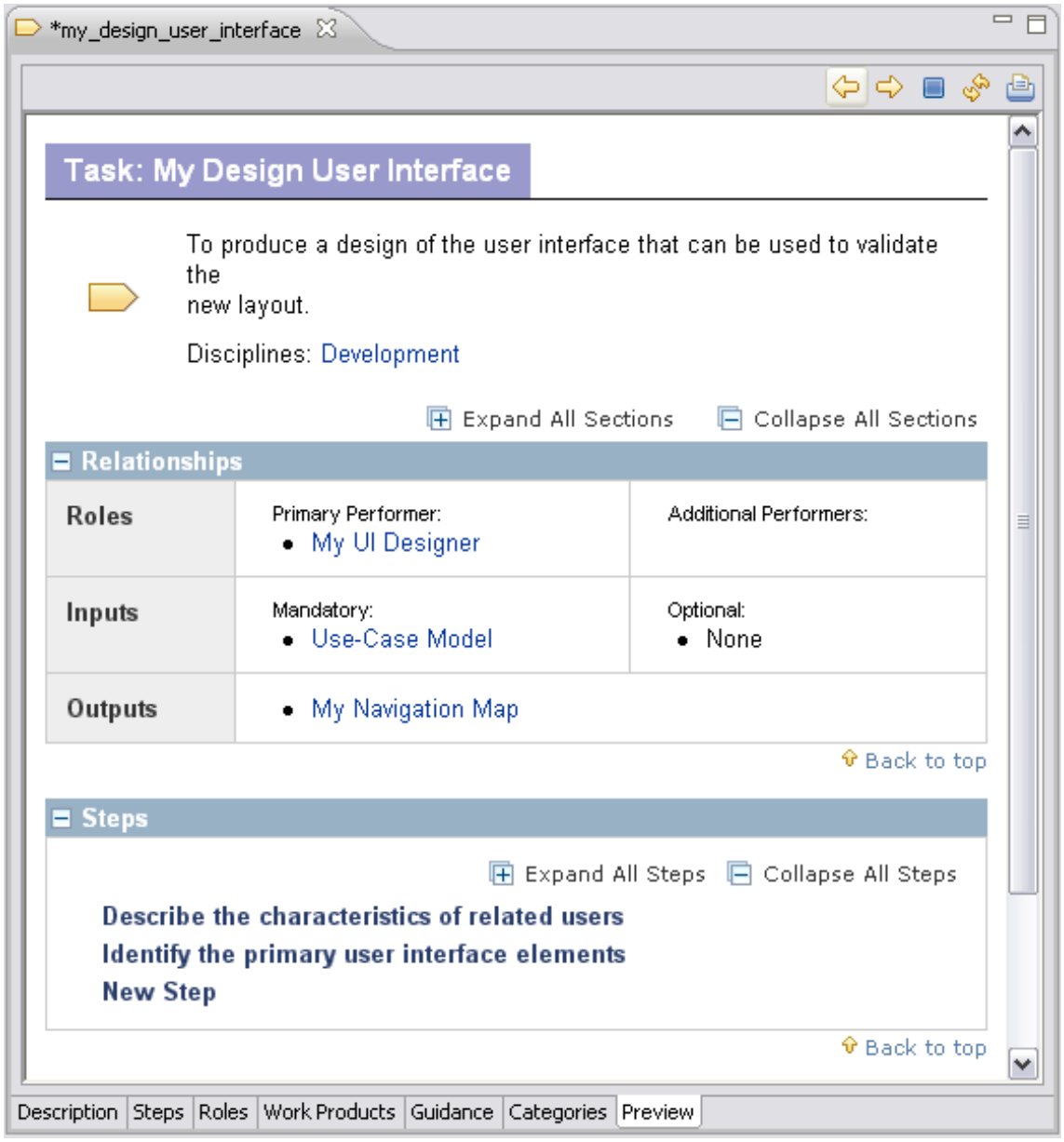
4.2.7. Work with Steps

The goal of this exercise is to use the Step Editor, which is part of the Task Editor.

When you create a task, you can define a series of steps that detail how to perform the task. Using the Step Editor, you can add new steps, change the sequence of steps, and delete steps.

To add steps in a task:

1. Double-click *my_design_user_interface*
2. Click the **Steps** tab.
3. Create a new step:
   a. Click **Add**.
   b. Enter a name for the new step in the **Steps** box.
   c. Enter a **Description**; you can use the rich text editor for this.
   d. See the results on the **Preview** tab



4. Remove a step:
   a. Return to the **Steps** tab.
   b. Select the step that you want to remove.
   c. Click **Delete**.
   d. See the results on the **Preview** tab.
5. Move a step up the list:
   a. Return to the **Steps** tab.
   b. Select the step that you want to move up.
   c. Click **Up**.
   d. See the results on the **Preview** tab.
6. Move a step down the list:
   a. Return to the **Steps** tab.
   b. Select the step that you want to move down.

c. Click **Down**.

d. See the results on the **Preview** tab.

4.2.8. Create Guidance Elements

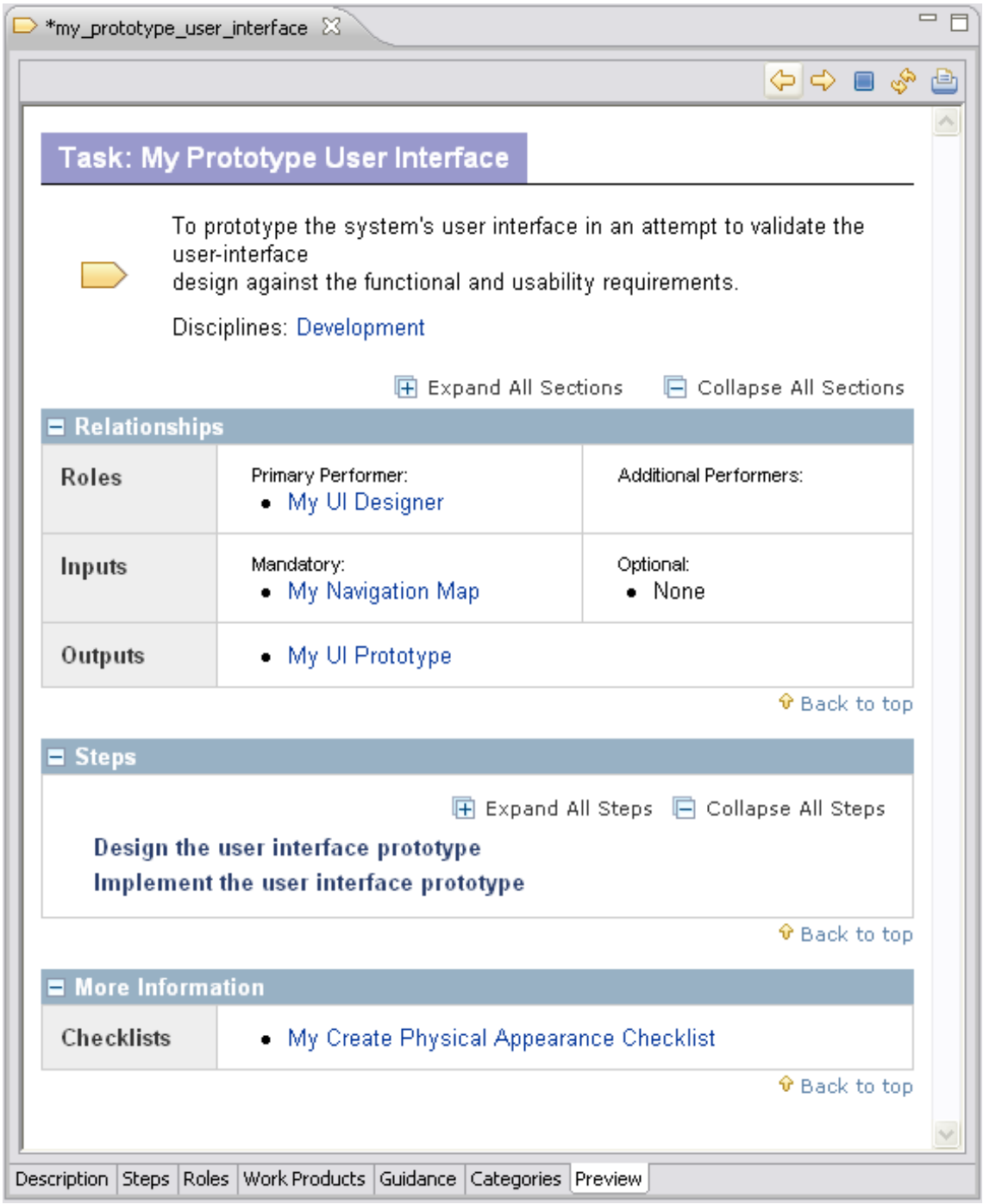
The goal of this exercise is to create a guidance element. Guidance elements are supplementary free-form documentation. There are fourteen guidance element types: Checklist, Concept, Example, Guideline, Estimation Consideration, Practice, Report, Reusable Asset, Roadmap, Supporting Material, Template, Term Definition, Tool Mentor and White Paper.

For this task we will create a representative guidance element, in this case, a checklist. After you have done this, you can try authoring additional guidance types.

In a checklist, check items are created in the same way as steps in a task are created, by using the *Check Items* tab in the checklist editor.

To create guidance elements:

1. In the Library view under *my_content_package*, right-click **Guidance** and select **New → Checklist**.
2. Name the guidance "*my_create_physical_appearance_checklist*" and enter the presentation name "*My Create Physical Appearance Checklist*".
3. Use the **Check Items** tab to add the following check items:
   - "*Create branding specifications*"
   - "*Define physical specifications*"
4. Close the guidance editor and save the new checklist.
5. Go back to the task you created in the previous exercise and add the new checklist to *my_prototype_user_interface*. Recall that you do this by clicking the **Guidance** tab and then selecting from a list of possible guidance elements. Shorten the list by searching for "*my**".
6. Use the **Preview** tab to generate a view of the page as it will look in a published site. *My Create Physical Appearance Checklist* should be displayed at the bottom of the page in a section labelled **More Information**.



4.2.9. Apply Guidance

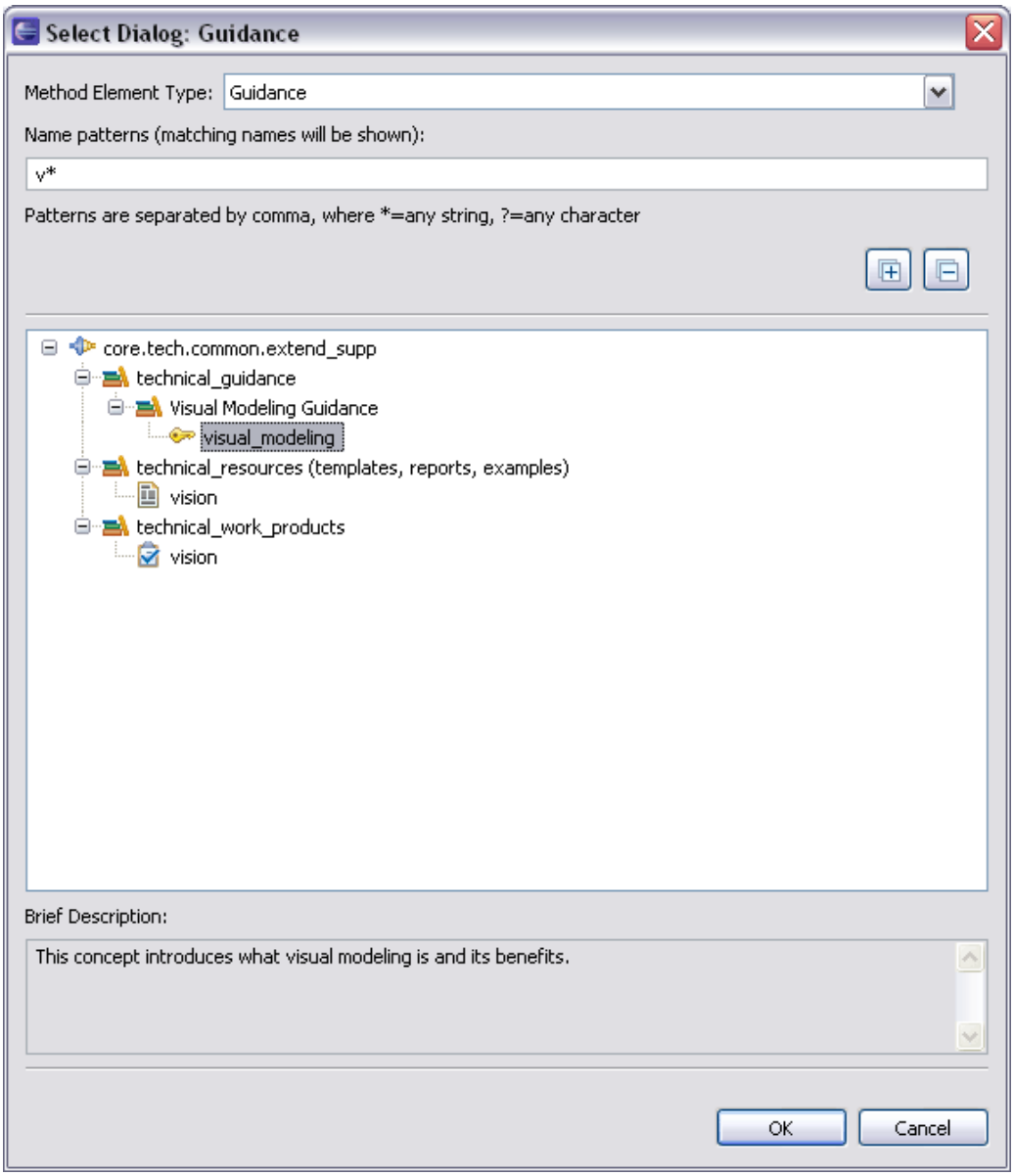
The goal of this exercise is to perform some simple management tasks, specifically adding and removing guidance.

Guidance can be attached to any core method content or process element. Guidance elements can be attached to other guidance elements.

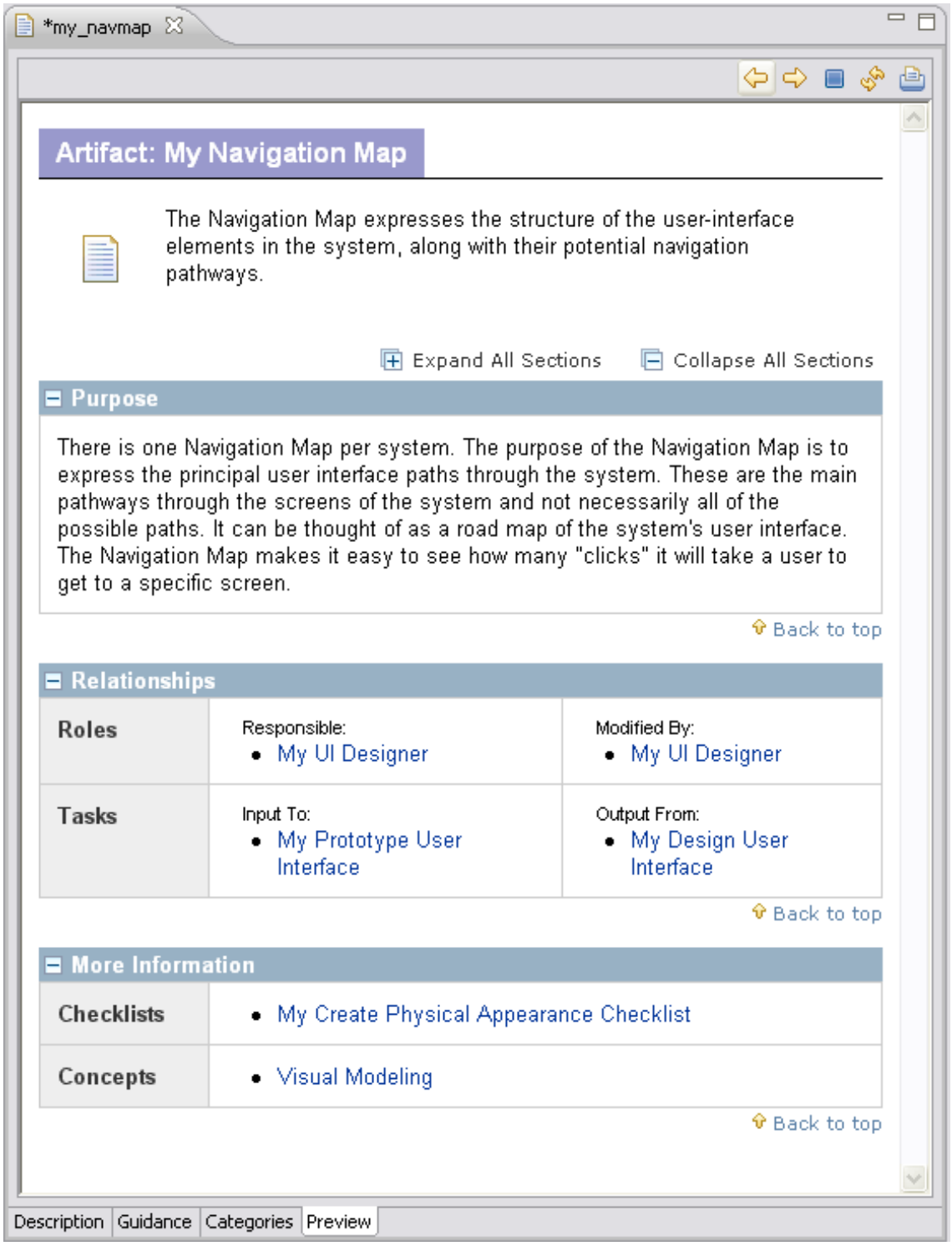
To add and remove guidance elements:

1. Locate the ***my_navmap*** artefact that you created earlier in this tutorial.
2. Double-click the icon for ***my_navmap***. The artefact editor panel opens.

3. Click the **Guidance** tab.
4. Click **Add**. The Select Dialog: Guidance window opens.
5. Add the guidance named *visual_modeling* from the *core.tech.common.extend_supp* plug-in. To shorten the list, enter "*v**" in the **Name patterns** box to filter the items that are displayed. Note that you can use the **Collapse All** and **Expand All** buttons to change the way the tree is displayed. Click **OK** after you make a selection.



6. Use the same process to add *my_create_physical_appearance_checklist* as an additional guidance item.
7. Click the **Preview** tab to view your work.



8. Go back to the Guidance editor and remove the concept *Visual Modelling*.
9. Click the **Preview** tab to view your work.

4.2.10. Create a Standard Method Category

The goal of this exercise is to create a new standard method category.

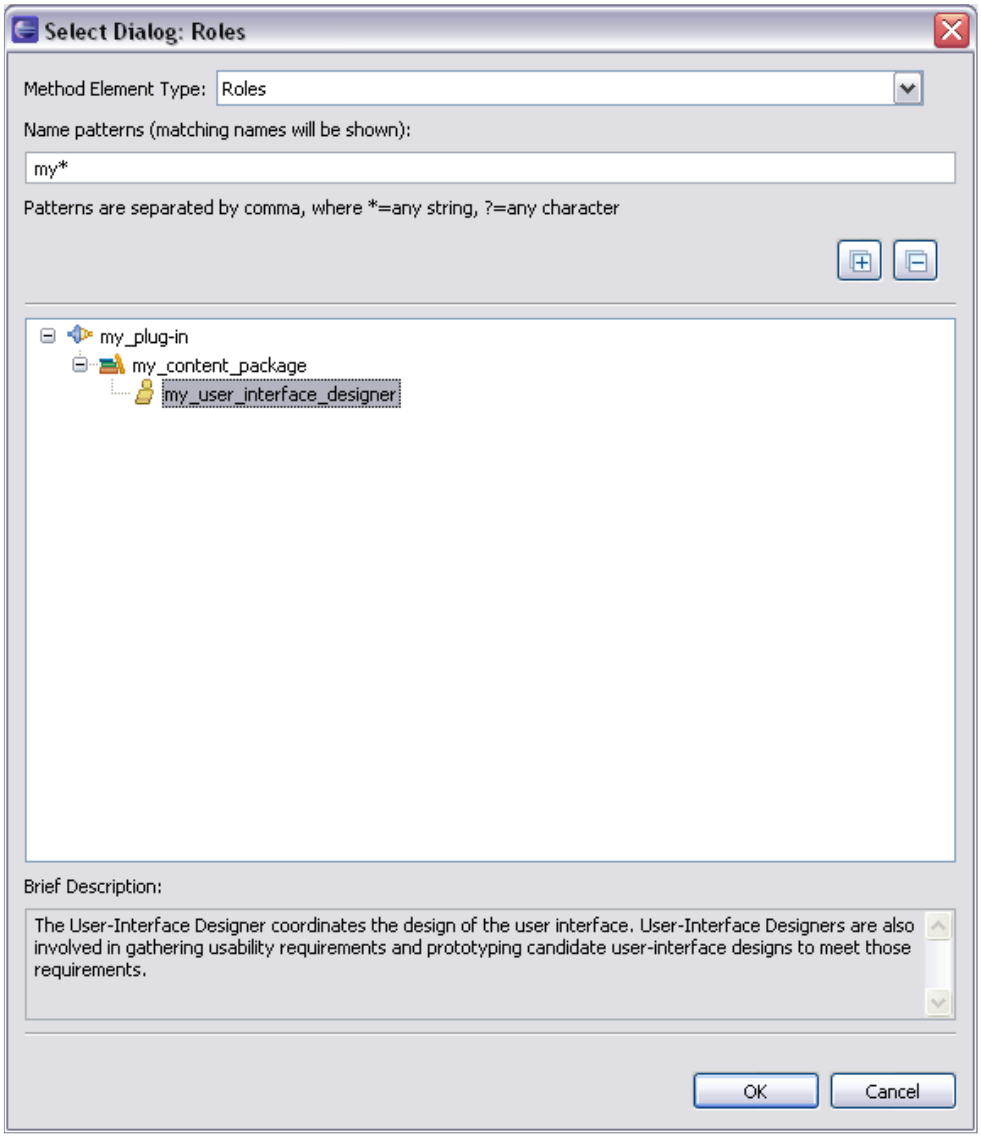
Standard method categories tend to be relatively stable. When you are creating a method plug-in that extends an existing method, you should try to use the standard

categories before creating new ones. You will therefore relatively rarely create new standard method categories.

Standard method categories are predefined and provide means to categorise method content in line with best practices for creating structured methods. Standard categories are linked to specific types of method content: **disciplines** are assigned to tasks, **role sets** groups roles, **domains** and **work products kinds** are for work products and **tool mentors** are assigned to tools. Each standard method category can only contain its specified type of method content; disciplines can only contain tasks for example.

To create a standard category:

1. Under **Standard Categories** in *my_plug-in* → **Method Content**, locate the **Role Sets** folder.
2. Right-click **Role Sets** and select **New** → **Role Set**.
3. Create a new Role Set named "*my_design*" with a presentation name of "*My Design*".
4. Add the role *my_user_interface_designer* to the *my_design* Role Set. There are two ways that you can do this:
   a. In the Standard Category editor for *my_design*, open the **Roles** tab, and add the role to the role set.



b. In the Role editor for *my_user_interface_designer*, open the **Categories** tab and add the **my_design** category as a selected role set.

4.2.11. Add a Method Plug-in to a Configuration

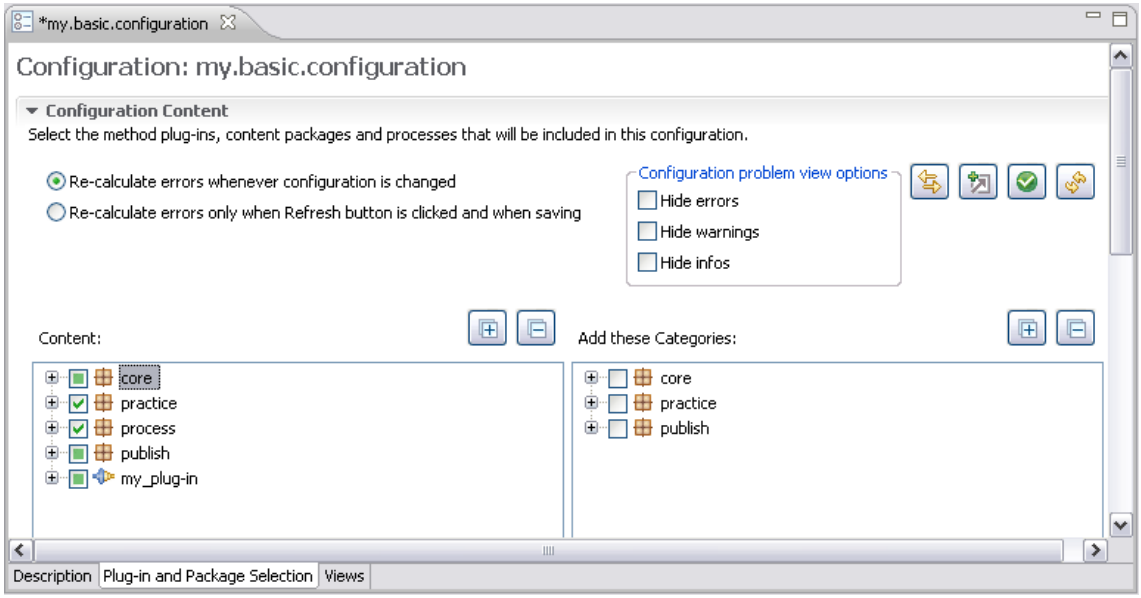
The goal of this exercise is to include the method plug-in created earlier to a copy of an existing configuration.

Before your new elements can be seen in the configuration view, you need to add your new method plug-in to a configuration. We will add the new method plug-in to a copy of the *publish.openup* configuration and see the results in the Configuration view.

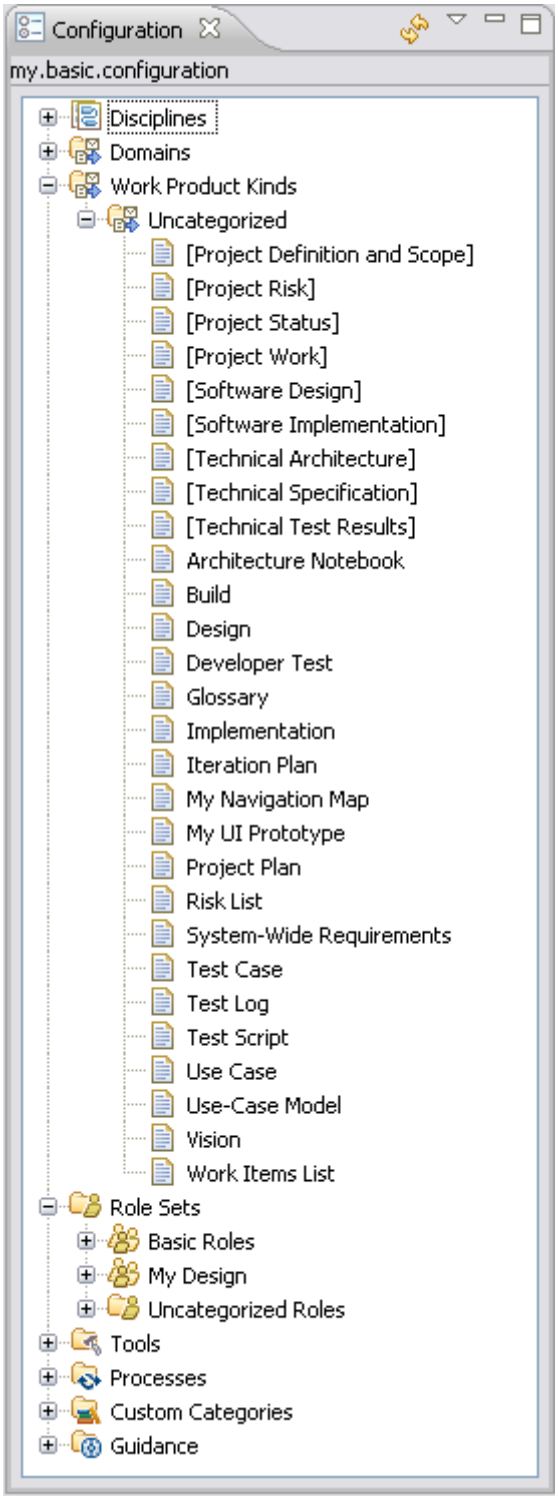
To add a plug-in to a configuration:

1. Use the Open Perspective button to switch to the **Browsing** perspective.
2. Try to find your new content in the configuration view panel. For example, look for **My Design** role set. You should not see it.
3. Switch to the **Authoring** perspective and then expand the **Configurations** node in the **Library view** panel.

4. Make a copy of the *publish.openup* configuration by right-clicking the configuration and selecting **Copy**. Right-click the **Configurations** folder and click **Paste**. Name the new configuration "*my.basic.configuration*".
5. Double-click *my.basic.configuration* to open the configuration.
6. Click the **Plug-in and Package Selection** tab.
7. Add *my_plug-in* to the configuration by checking the box next to it in the **Content** section.



8. Close the editor and save the changes.
9. Change the current configuration to *my.basic.configuration* in the configuration selection box in the main menu bar.
10. Switch to the **Browsing** perspective and look for your content in the **Configuration** view. You should see *My Design* role set. You should also see your new work products in the **Uncategorised** folder under **Work Product Kinds**.



4.3. Reuse Method Content

This tutorial contains a summary of background information followed by eight guided exercises.

Learning objectives

Upon completion of this tutorial you should be able to do the following:

- Gain an overview of method content variability and how it promotes reuse of method content.
- Customise method content in an existing method plug-in by using the contributes variability mechanism.
- Customise method content by using the extends variability mechanism.
- Customise method content by using the replaces variability mechanism.
- Customise method content by using the extends and replaces variability mechanism.

Time required

The estimated time to complete this tutorial is about 2 hours.

Prerequisites

Create method content tutorial.

Lessons in this module

- Concepts

  This page provides background information required to complete the exercises in this tutorial.

- Contribute to a Role

  The goal of this exercise is to extend a role by adding a contribution.

- Contribute to a Work Product

  The goal of this exercise is to extend a work product by adding a contribution.

- Contribute to a Task

  The goal of this exercise is to extend a task by adding a contribution.

- Extend a Role

  The goal of this exercise is to extend base method content associated with a role by using Extends variability.

- Extend a Work Product

  The goal of this exercise is to customise a base method work product using Extends variability.

- Extend a Task

  The goal of this exercise is to extend a base method content associated with a task by using Extends variability.

- Replace a Role

  The goal of this exercise is to extend base method content associated with a role using Replaces variability.

- Extend and Replace a Role

  The goal of this exercise is to extend base method content associated with a role using Extends and Replaces variability.

4.3.1. Concepts

This page provides background information required to complete the exercises in this tutorial.

In this tutorial we will use a powerful mechanism for reusing and customising existing content called **method content variability**. The original content will not be affected by our operations; however, we will build upon it as a base for defining our new content. Then we will demonstrate that modifications to the base content are automatically inherited by variable content in dependent plug-ins. When the resulting configuration is published as a Web site or exported, all of our customisation will be resolved into a single resource.

There are four types of method content variability and in this tutorial we will use all of them to customise existing content. The following are the four ways for using method content variability:

- **Contributes**
- **Extends**
- **Replaces**
- **Extends and Replaces**

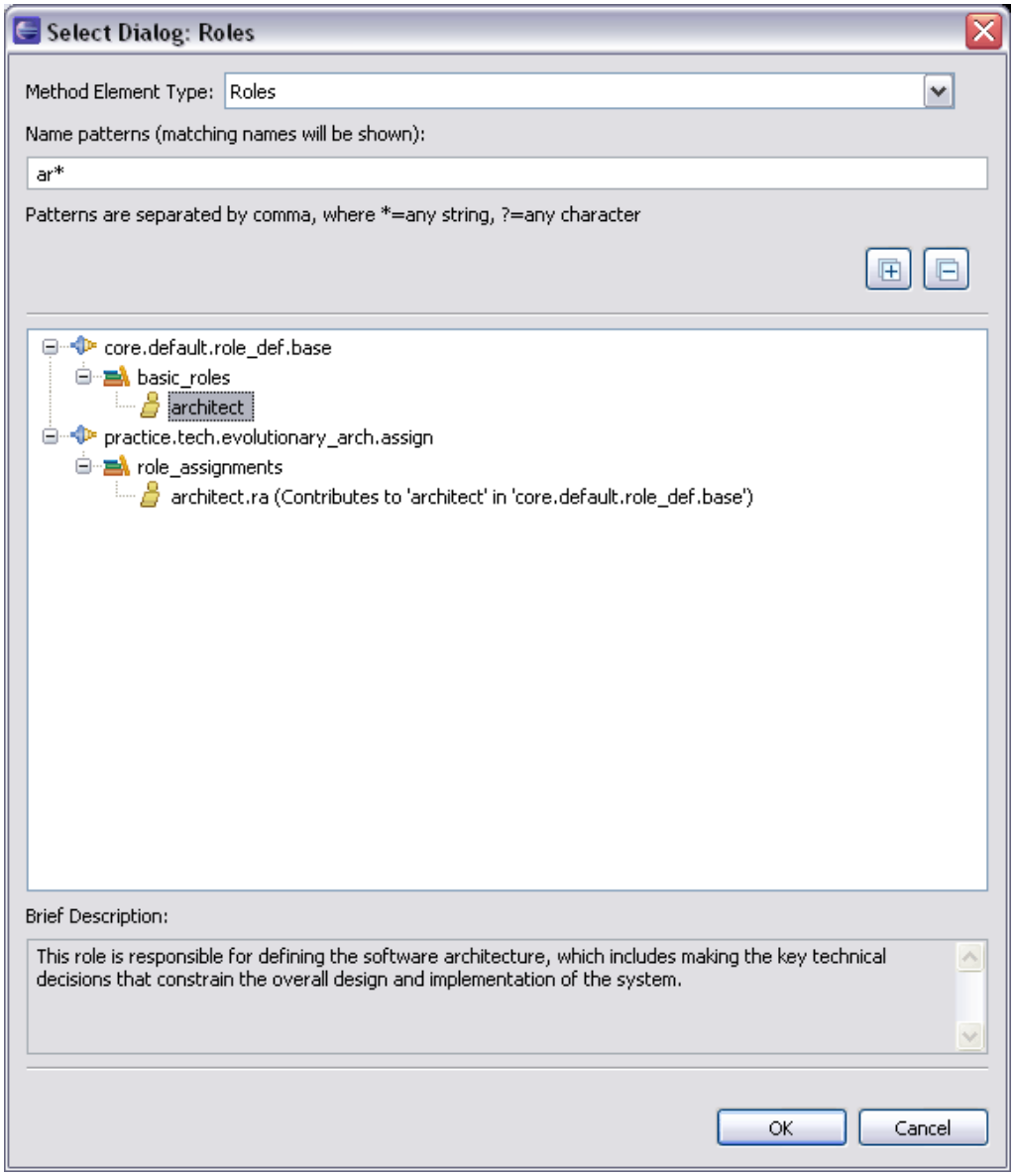
4.3.2. Contribute to a Role

The goal of this exercise is to extend a role by adding a contribution.

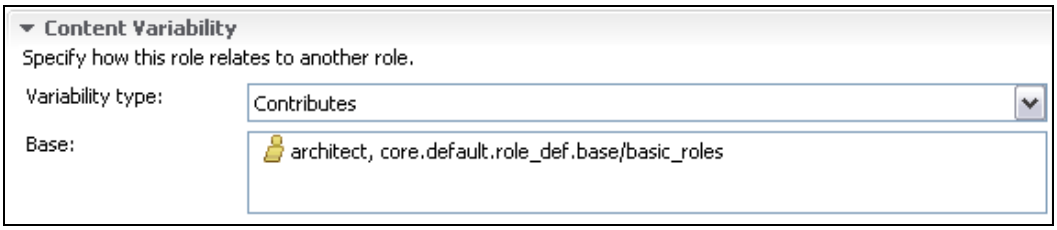
To contribute to a role:

1. Switch to the **Authoring** perspective if you are not already using it.
2. Expand *my_plug-in* until you see the **Content Packages** node. Right-click this node to create a new content package called "*Contribution Test*". Save your changes by closing the editor window after you have named the new content package.
3. Expand the newly created content package and right-click the **Roles** folder and select **New** → **Role**.
4. Use the following attributes for this new role:
   - **Name**: "*my_contributing_architect*"
   - **Presentation name**: "*My Contributing Architect*"

   Do not close the editor window yet because we have not finished defining this role.
5. Scroll down the window to the **Content Variability** section, use the Variability type menu and select **Contributes**.
6. Click **Select** to the right of **Base** in the Content Variability section, the Select Dialog: Roles window opens. Choose the base role *architect*. You will find this in the *core.default.role_def.base* plug-in.



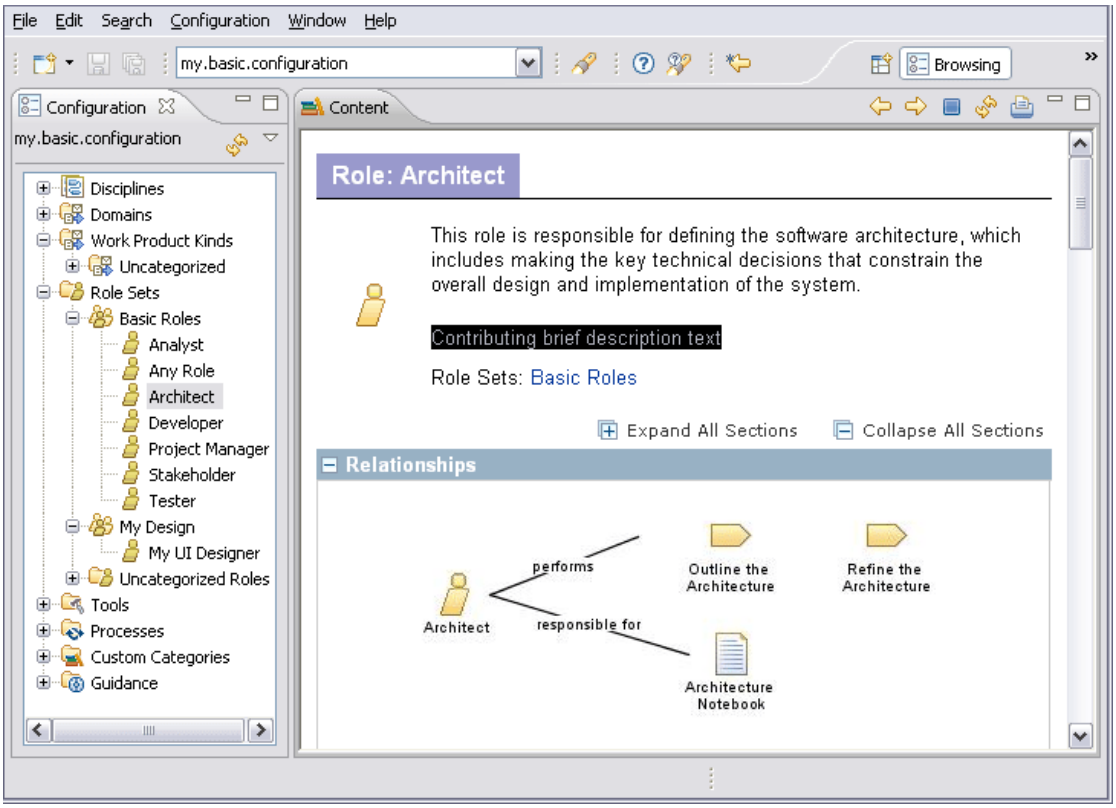
Click **OK** to complete the selection. You can now continue using the role editor. The **Content Variability** section should look like this:



7. Add some sample text to the Brief description and Main description fields.
   - **Brief description:** "*Contributing brief description text*"
   - **Main description:** "*Contributing main description text*"

   We will look for these text strings later in this exercise. Save your changes by closing the role editor.

8. In the Library view under Configuration, double-click *my.basic.configuration* to open it.
9. Click the **Plug-in and Package Selection** tab. Make sure that *base* under *core* → *default* → *role_def* package and the method plug-in that we created earlier (*my_plug-in*) are selected. Save the new configuration by closing the editor window.
10. Use the configuration selection pull-down menu to select *my.basic.configuration*.
11. Switch to the **Browsing** perspective and then open the *Architect* role in the **Configuration** panel. It is in the **Role Sets** → *Basic Roles* folder. You should see the text that you included in the contributing role appended to the existing brief description of this role at the top of the page. This is how contributing variability works: it appends text to the base element.



Note: The Configuration view resolves variability relationships between related plug-ins in a configuration and shows the results, as they will be published in a published Web site.

4.3.3. Contribute to a Work Product

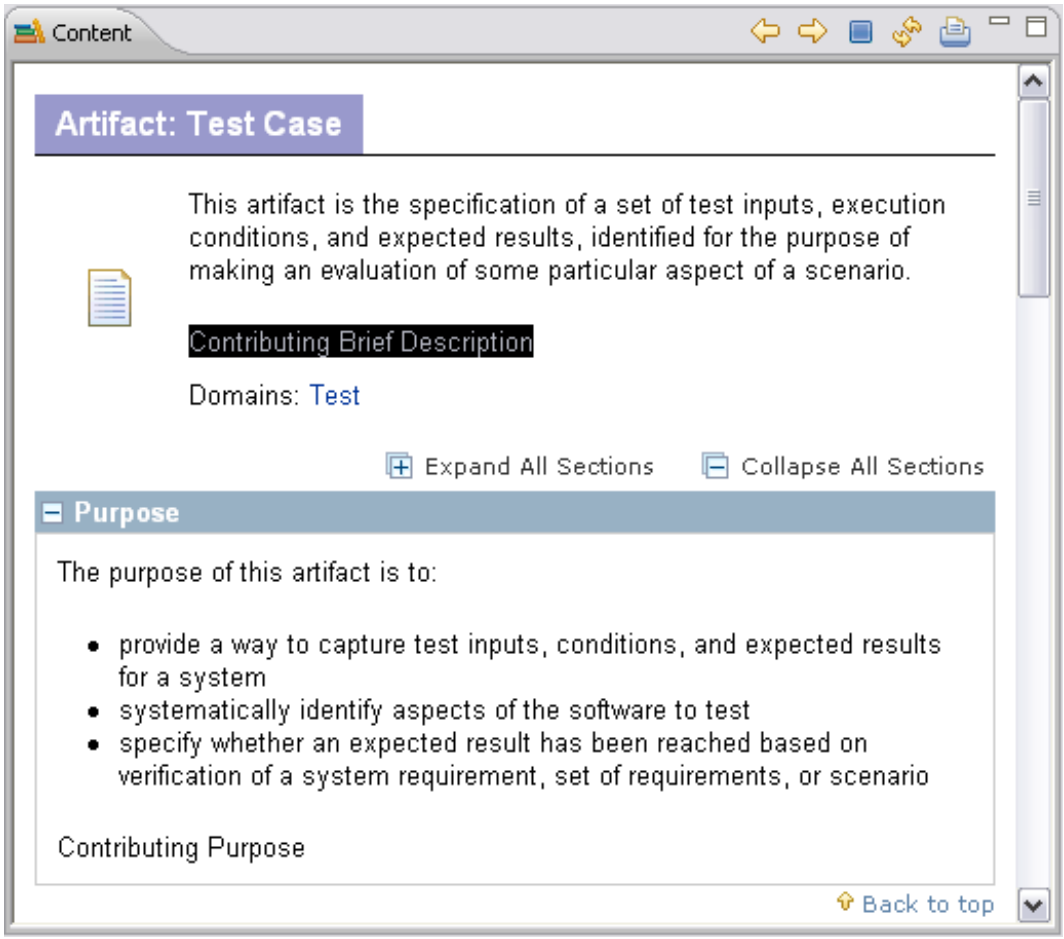
The goal of this exercise is to extend a work product by adding a contribution.

To contribute to a work product:

1. Switch to the **Authoring** perspective if you are not already using it.
2. In *my_plug-in* under the *contribution_test* folder, right-click **Work Products** and select **New** → **Artefact**.
3. Give the new artefact the following attributes:
   - **Name**: "*my_contributing_test_case*"

- **Presentation Name**: "*My Contributing Test Case*"
- **Brief description**: "*Contributing Brief Description*"
- **Purpose**: "*Contributing Purpose*"
- **Main description**: "*Contributing Main Description*"

4. Use the **Variability Type** menu to select **Contributes**, then click **Select** to the right of **Base** in this section. The Select Dialog: Artefacts window opens.
5. Select *test_case* and click **OK** to close this window. You can find this under *core.tech.common.exten_supp* → *technical_work_products*.
6. Save your changes by closing the editor.
7. Switch back to the **Browsing** perspective. Make sure that you are using *my.basic.configuration*.
8. Use the **Configuration** view to **open Work Product Kinds** → **Uncategorised**, then locate *Test Case*.
9. Click the *Test Case* icon to invoke the editor. The description now has text that you included in the contributing artefact. You will see it appended to the appropriate fields in the customised test case.



This is another example of using **contributes** variability to customise content in a dependent plug-in by making changes to the base plug-in.

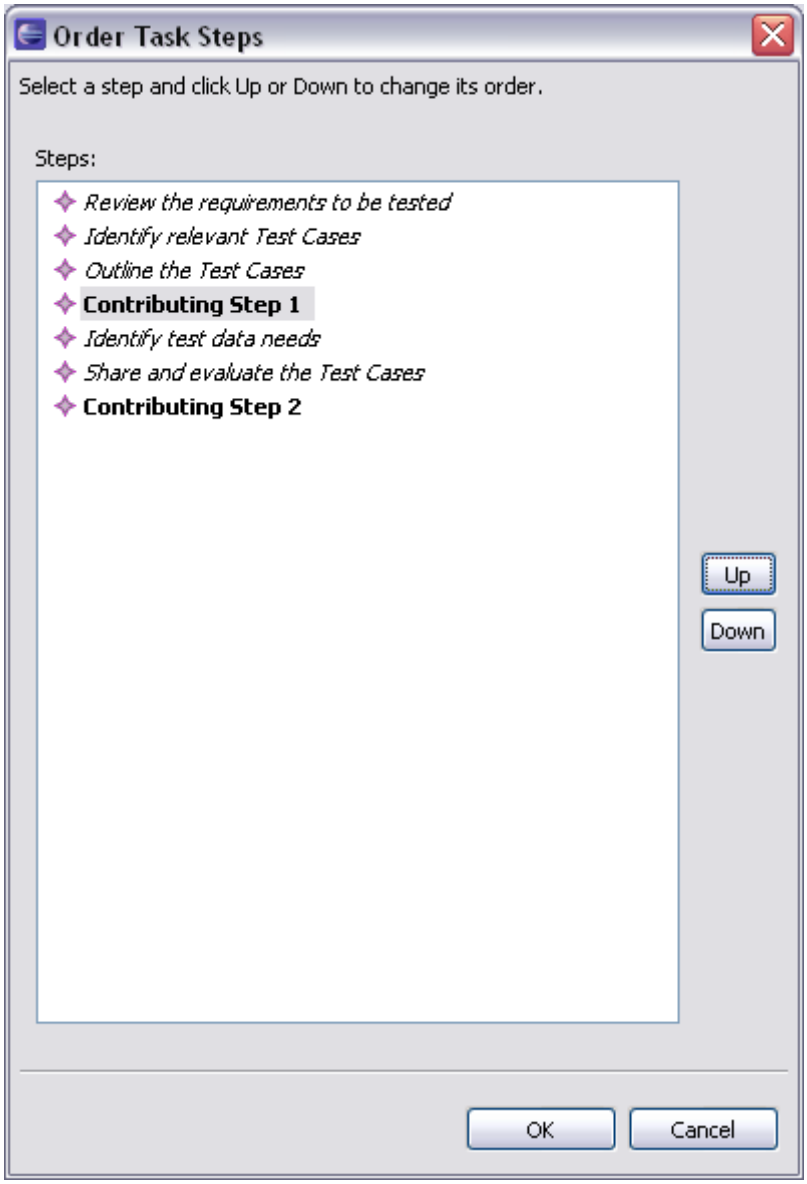
4.3.4. Contribute to a Task

The goal of this exercise is to extend a task by adding a contribution.

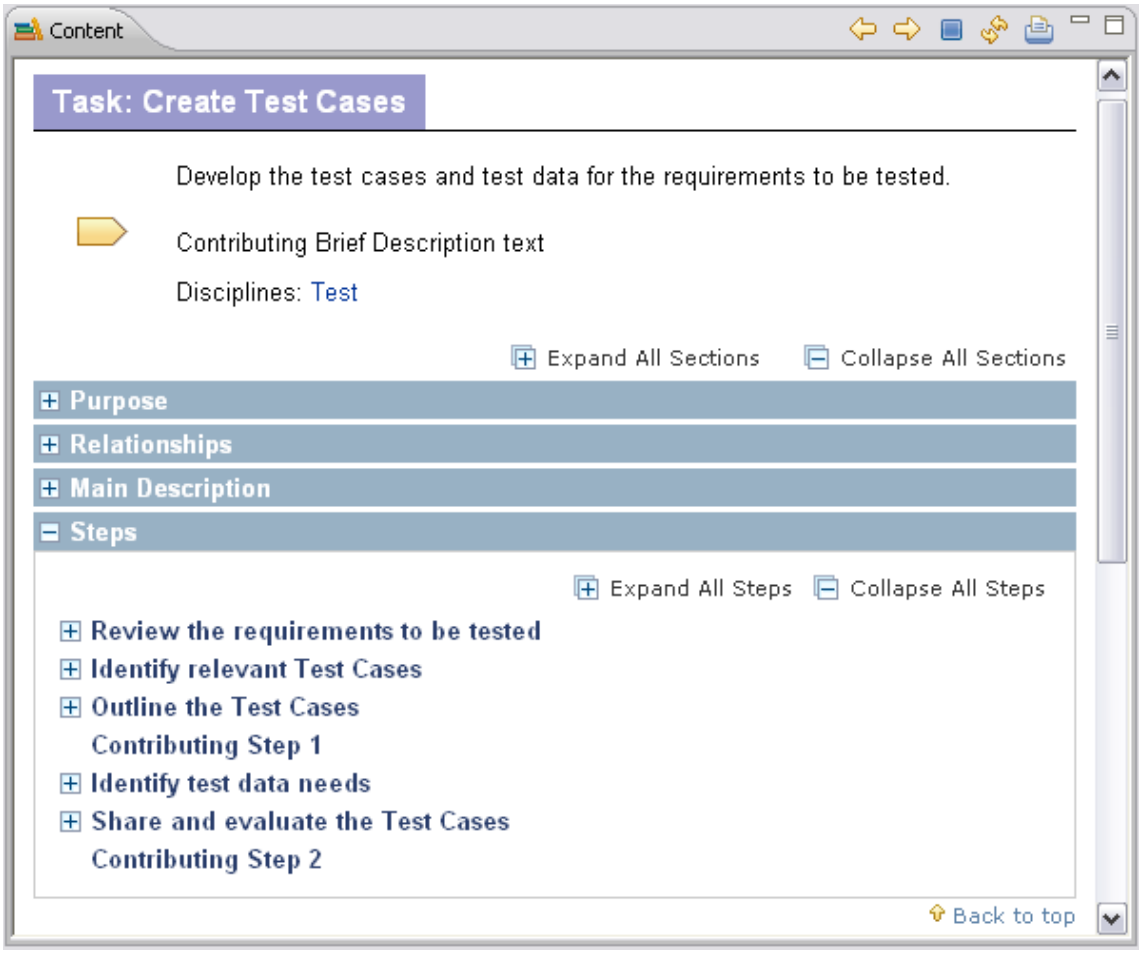
By now you should start to see the general pattern with this variability mechanism. Recall that contributions append text in appropriate fields from one content package to another.

To contribute to a task:

1. Switch back to the **Authoring** perspective.
2. Create a new task in the *contribution_test* content package.
3. Use the following attributes for this task:
   - **Name**: "*my_contributing_task*"
   - **Presentation Name**: "*My Contributing Task*"
   - **Brief description**: "*Contributing Brief Description text*"
   - **Purpose**: "*Contributing Purpose text*"
   - **Main Description**: "*Contributing Main Description text*"
4. Click the **Steps** tab and add two new steps named "*Contributing Step 1*" and "*Contributing Step 2*".
5. Click the **Description** tab. In the **Content Variability** section, select **Contributes** as **Variability type**. Click **Select** to the right of **Base** in the Content Variability section, The Select Dialog: Tasks window opens.
6. Select *create_test_cases* and click **OK**. This will be the base element for our contribution.
7. Go back to the **Steps** tab and click **Order**. Use the **Up** and **Down** buttons to reorder the steps. Note that you can now insert your new steps into the original sequence of steps inherited from the base. This is a good example of the power of variability contributions.



8. Save the new task by closing the editor panel.
9. Switch to the Browsing perspective. Make sure that you are using ***my.basic.configuration***.
10. Expand **Disciplines** → **Test**. Click ***the Create Test Cases*** task. The content panel shows a preview of the generated page. Look at the steps to see how your new material was contributed. You should see the text that you included in the contributing task displayed in the ***Create Test Cases*** task.



11. You can also try specifying additional artefacts, guidance, and additional performer roles in the contributing task (using the Authoring perspective) and then view the result in the Configuration view to see the behaviour.

4.3.5. Extend a Role

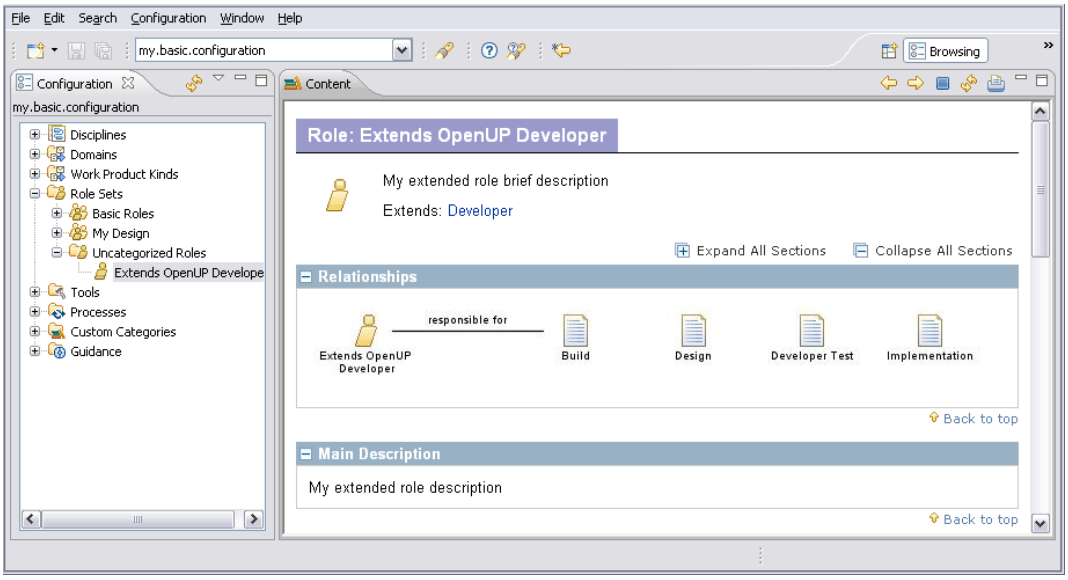
The goal of this exercise is to extend base method content associated with a role by using Extends variability.

Extends variability works differently than contributes variability in that the method content element that extends the base method element inherits the attributes of the extended base element.

To extend a role:

1. Switch back to the **Authoring** perspective.
2. Create a new content package under *my_plug-in*. Name the new content package "*extends_test*". Close the editor panel and save your changes.
3. Create a new role under the *extends_test* content package.
4. Use these attributes for the new role:
   - **Name:** "*my_extends_developer*"
   - **Presentation name:** "*Extends OpenUP Developer*"
   - **Brief description:** "*My extended role brief description*"
   - **Main description:** "*My extended role description*"

5. Use the **Variability type** menu to select **Extends** then click **Select**. The Select Dialog: Roles window opens.
6. Select the *developer* role and click **OK**. This is now the base role extended by our customisation.
7. Save the new role by closing the editor panel.
8. Update the configuration to include the new content package by doing the following:
   a. Double-click the *my.basic.configuration* icon in the Library view panel. The configuration editor panel opens.
   b. Click the **Plug-in and Package Selection** tab.
   c. Expand *my_plug-in* → **Method Content** → **Content Packages**.
   d. If they are checked, clear *contribution_test* and *my_content_package*. We do this to make sure that no contributions are coming from other content packages.
   e. Make sure that *extends_test* is checked.
   f. Close and save the changes to the configuration.
9. Switch to the **Browsing** perspective. Make sure that you are using *my.basic.configuration*.
10. In the **Configuration** view, select *Extends OpenUp Developer* in the **Uncategorised Roles** folder under the **Role Sets** folder. You will see in the HTML page that content entered in the extending role has been replaced, whereas content not provided has been inherited from the base artefact. Note that the extended role Developer remains unchanged.

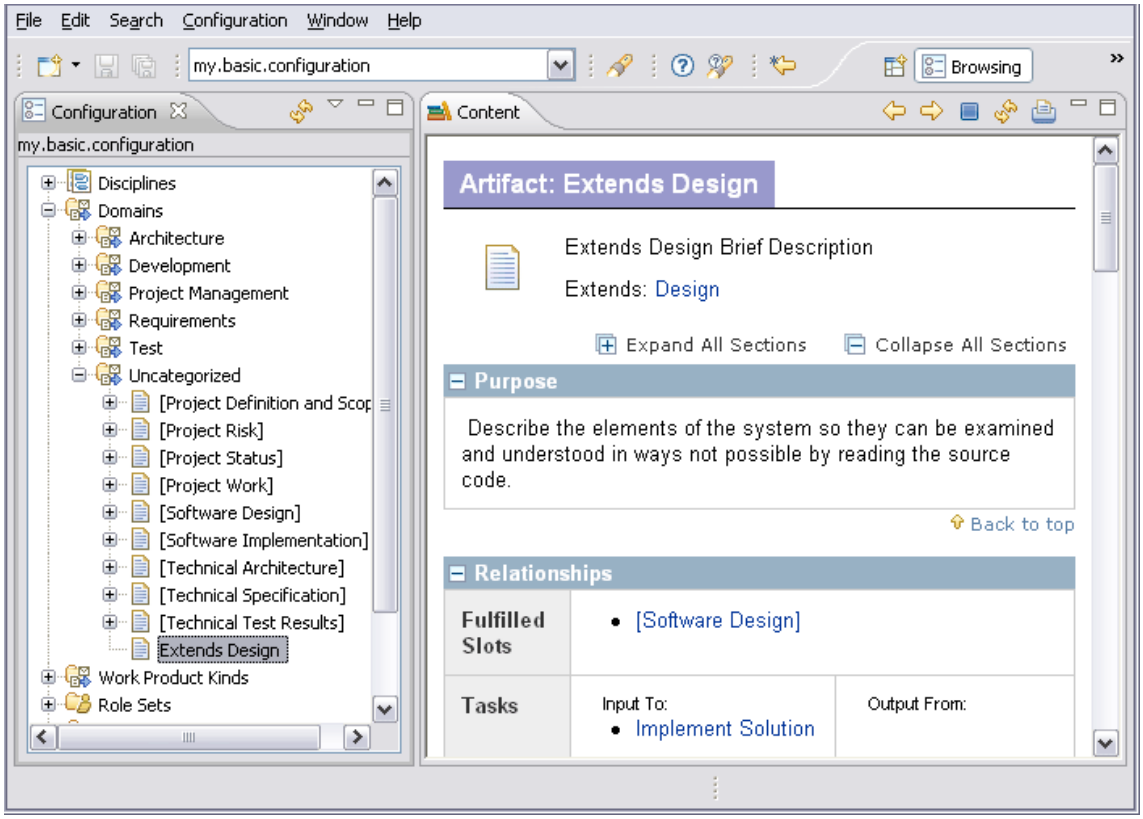


4.3.6. Extend a Work Product

The goal of this exercise is to customise a base method work product, using Extends variability.

To extend a work product:

1. Switch back to the **Authoring** perspective.
2. In the Library view under *extends_test* content package create a new work product of type artefact.
3. Use these attributes for the new artefact:
   - **Name:** "*my_extends_design*"
   - **Presentation name:** "*Extends Design*"
   - **Brief description:** "*Extends Design Brief Description*"
4. Use the **Variability type** menu to select **Extends** then click **Select**. The Select Dialog: Artefact window opens.
5. Choose the work product *design* as the base and click **OK**.
6. Close the editor window to save your changes.
7. Switch to the **Browsing** perspective.
8. In the **Configuration** view select the *Extends Design* work product in the **Uncategorised** folder under **Domains**. You will see in the page preview that content entered in the extending artefact has replaced the content in the base artefact, whereas content not provided has been inherited from the base artefact.



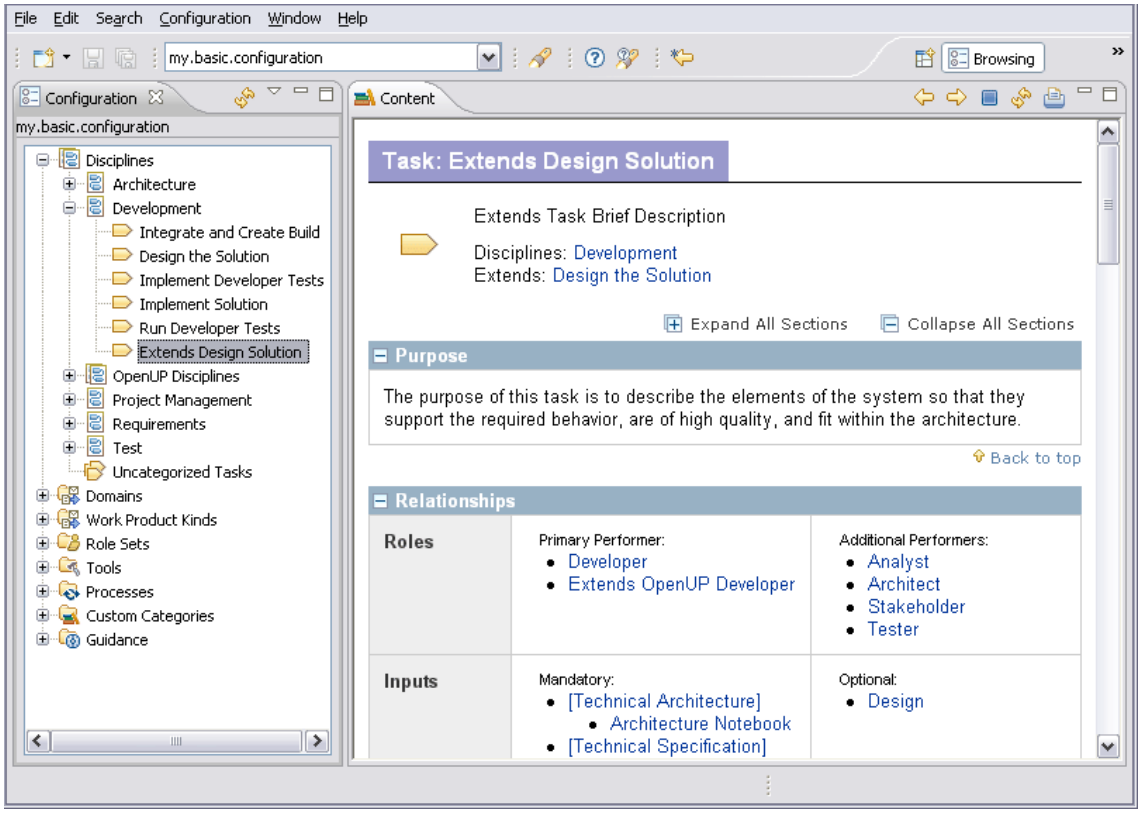
4.3.7. Extend a Task

The goal of this exercise is to extend a base method content associated with a task by using Extends variability.

To extend a task:

1. Switch back to the Authoring perspective.

2. Create a new task in the **extends_test** content package.
3. Use these attributes for the new task:
   - **Name:** "*my_extends_design_solution*"
   - **Presentation name:** "*Extends Design Solution*"
   - **Brief description:** "*Extends Task Brief Description*"
4. Use the **Variability type** menu to select **Extends** then click **Select**. The Select Dialog: Tasks window opens.
5. Select *design_solution* to be the base task. Click **OK** to close the selection window.
6. Click the **Steps** tab and add two new steps:
   - "Extend Step 1"
   - "Extend Step 2"
7. Click the **Roles** tab and add the role that we created in Exercise 4, *my_extends_developer*, as the **Primary Performer**.
8. Click the **Work Products** tab and add *my_extends_design* as an output.
9. Click the **Guidance** tab and add a guidance. You can pick anyone that you want here.
10. Click the **Categories** tab to add the new task to an appropriate category. Click **Add** next to the **Disciplines** field, and then select *development_discipline*. Click **OK** to save changes and close the selection window.
11. Close the task editor panel to save your changes.
12. Switch to the **Browsing** perspective.
13. In the **Configuration** view panel, expand **Disciplines** → *Development* and click the *Extends Design Solution* task. You will see in the generated page that method content and relationships are inherited from the base task and customised by extending content and relationships. Note also that the extended task remains unchanged.



4.3.8. Replace a Role

The goal of this exercise is to extend base method content associated with a role using **Replaces** variability.

To replace a role:

1. Switch back to the **Authoring** perspective.
2. Create a new content package in *my_plug-in*. Name the new content package "*replace_test*". Save your changes by closing the editor panel.
3. Create a new role in the content package that we just created.
4. Use these attributes for the new role:
   - **Name**: "*my_replace_tester*"
   - **Presentation name**: "*My Replace Tester*"
   - **Brief description**: "*My replaced brief description*"
   - **Main description**: "*My replaced main description*"
5. Use the **Variability type** menu to select **Replaces** then click **Select** to the right of this section. The Select Dialog: Roles window opens.
6. Select *tester* to be the base role. Click **OK** to save changes and close the selection window.
7. Save your changes by closing the role editor panel.
8. Update *my.basic.configuration* by doing the following:
   a. Double-click the *my.basic.configuration* icon in the Library view panel. The configuration editor opens.

b. In the **Plug-in and Package Selection** tab, expand *my_plug-in* → **Method Content** → **Content Packages**.

c. If they are checked, clear *contribution_test* and *extends_test*.

d. Check *replace_test*. This should be the only content package selected in *my_plug-in*. This is done so that there are no conflicting extensions in the same method element.

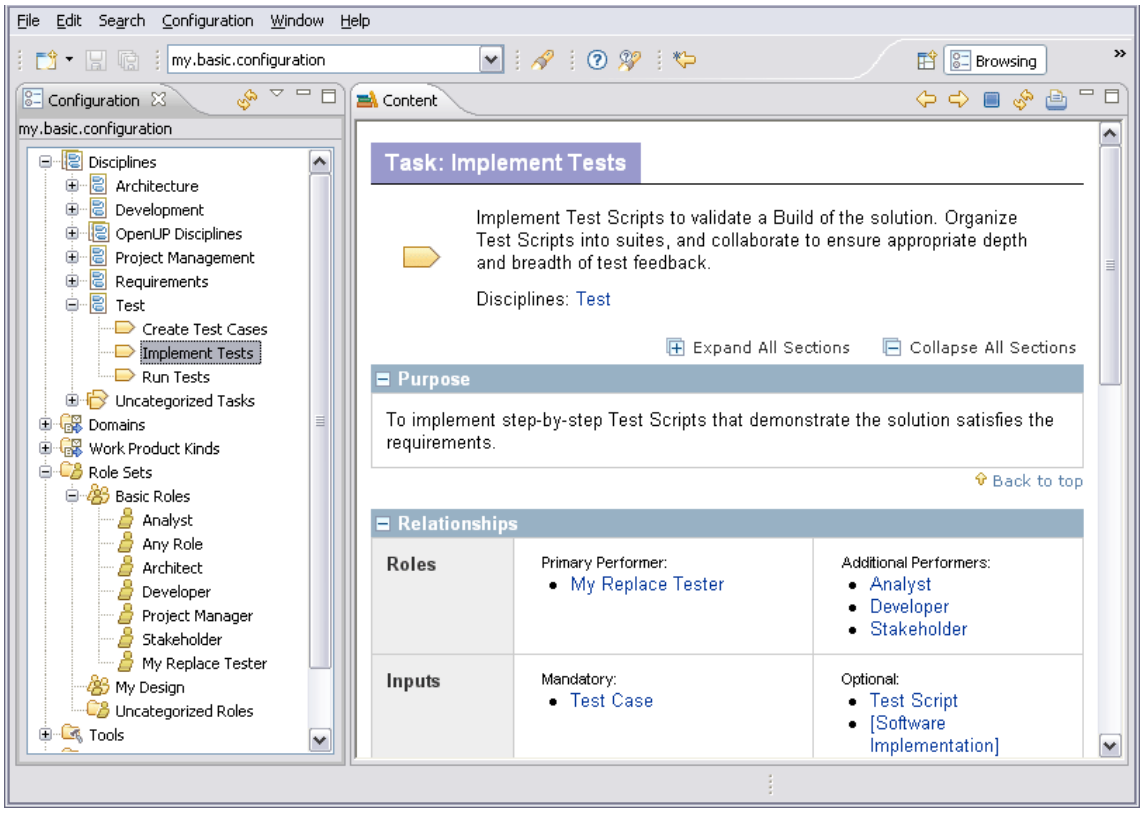
e. Close the configuration editor panel to save your changes.

9. Now switch to the **Browsing** perspective.

10. Expand **Role Sets** → *Basic Roles*.

In the **Configuration** view you will see that there is no tester role in the Basic Roles folder. It was replaced by the *My Replace Tester* role.

Also notice that other content elements in the base plug-in that referenced the tester role now reference the replacing role instead. If you open the **Disciplines** folder and view the task *Implement Test*, you will see that the **Primary Performer** is now *My Replace Tester*. Replace works in a similar way for tasks, artefacts, guidance and categories.



4.3.9. Extend and Replace a Role

The goal of this exercise is to extend base method content associated with a role using **Extends and Replaces** variability.

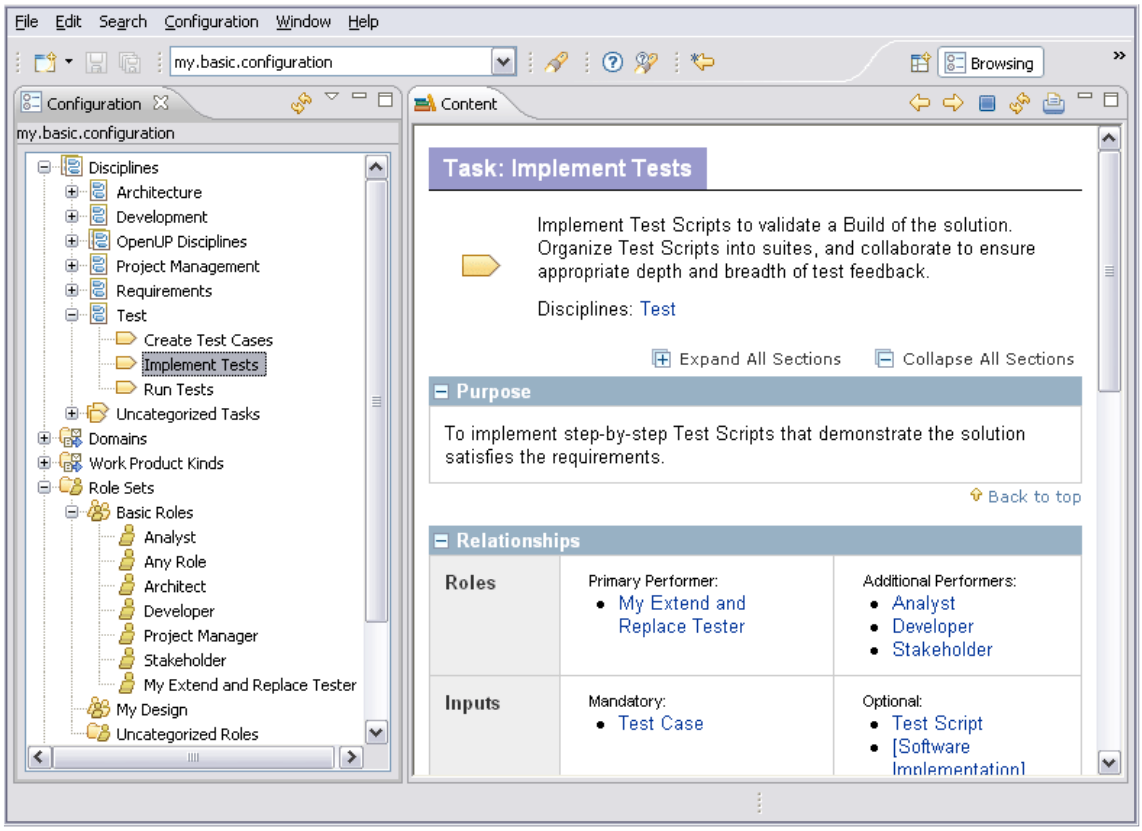
To extend and replace a role:

1. Switch back to the **Authoring** perspective.

2. Create a new content package in *my_plug-in*. Name the new content package "*extend_and_replace_test*". Save your changes by closing the editor panel.
3. Create a new role in the content package that we just created.
4. Use these attributes for the new role:
   - **Name**: "*my_extend_replace_tester*"
   - **Presentation name**: "*My Extend and Replace Tester*"
   - **Brief description**: "*My extend and replace brief description*"
   - **Main description**: "*My extend and replace main description*"
5. Use the **Variability Type** menu to select **Extends and Replaces** then click the **Select** button in this section. The Select Dialog: Roles window opens.
6. Select *tester* to be the base role. Click **OK** to save changes and close the selection window.
7. Save your changes by closing the role editor panel.
8. Update *my.basic.configuration* by doing the following:
   a. Double-click the *my.basic.configuration* icon in the **Library** view panel. The configuration editor opens.
   b. In the Plug-in and Package Selection tab expand *my_plug-in* → **Method Content** → **Content Packages**.
   c. Clear all selected content packages.
   d. Check *extend_and_replace_test*. This should be the only content package selected in *my_plug-in*. This is done so that there are no conflicting extensions in the same method element.
   e. Close the configuration editor panel to save your changes.
9. Now switch to the **Browsing** perspective.
10. Open **Role Sets** → *Basic Roles*.

    In the Configuration view you will see that there is no tester role in the Basic Roles folder. It was replaced by the **My Extend and Replace Tester** role.

    Also notice that other content elements in the base plug-in that referenced the tester role now reference the extend and replacing role instead. If you open the **Disciplines** folder and view the task *Implement Test*, you will see that the **Primary Performer** is now *My Extend and Replace Tester*. Extends and Replaces works in a similar way for tasks, artefacts, guidance and categories.



This lesson concludes the "Reuse method content" tutorial. Proceed with next tutorial, "Working with processes".

4.4. Work with Processes

This tutorial contains a summary of key concepts followed by five exercises.

Learning objectives

Upon completion of this tutorial you should be able to do the following:

- Browse existing process content
- Create a delivery process
- Create a process diagram

Time required

The estimated time to complete this tutorial is about 2 hours.

Prerequisites

Reuse method content tutorial.

Lessons in this module

- Concepts

This topic provides basic background information needed to complete this tutorial.

- Browse Process Content

  The goal of this exercise is to gain a basic understanding of processes.

- Create a Delivery Process

  The goal of this exercise is to create a delivery process using core method content directly.

- Use Capability Patterns

  The goal of this exercise is to create a new delivery process using capability patterns.

- Create a Process Diagram

  The goal of this exercise is to use diagram features.

4.4.1. Concepts

This topic provides basic background information needed to complete this tutorial.

Using process authoring, a Process Engineer can incorporate method elements into process structures, for example, a work breakdown structure format familiar to Project Managers. The processes can be included in a configuration to be published as part of the published Web site, and can be exported to Microsoft® Project.

In **method authoring**, the Process Engineer defines roles, tasks, work products and guidance, in addition to the relationships between these elements.

In **process authoring**, the Process Engineer defines additional lifecycle elements, such as activities (summary tasks), phases, iterations, and milestones, which can then be used to compose the core elements into processes. A complete process corresponding to a project plan, or a phase, is a **delivery process**. The OpenUp process is an example of a delivery process. We can also create smaller more granular sections of process, termed **capability patterns**, which can be used as building blocks to compose delivery processes more easily.

Each time a task is included in a process, a copy of that task is created in the context of the process. This is a **task descriptor**. The same task can be included any number of times in the same process. This allows, for example, the same OpenUp task to be included in every iteration within each OpenUp Phase. A task descriptor can also ***modify*** the base task without actually changing the task. ***For example, roles and work products can be added or suppressed; steps can be suppressed or re-sequenced***.

Roles and work products are also included in processes as role descriptors and work product descriptors. Roles and work products can also be customised to fit with the context of the process in which they are used.

Process Diagrams

EPF Composer provides three types of process diagrams:

- **Activity diagrams:** These diagrams show the subordinate activities in a higher-level activity. They also show the sequence relationships between those activities.

- **Activity detail diagrams:** These diagrams show tasks in an activity with their performing roles along with input and output work products. Activity detail diagrams are similar to workflow detail diagrams.
- **Work product dependency diagrams:** These diagrams illustrate work product dependencies on other work products.

All three types of diagrams are generated and synchronised with the associated work breakdown structure. Changes to the process structure using the diagram editor will be automatically reflected in the work breakdown structure.

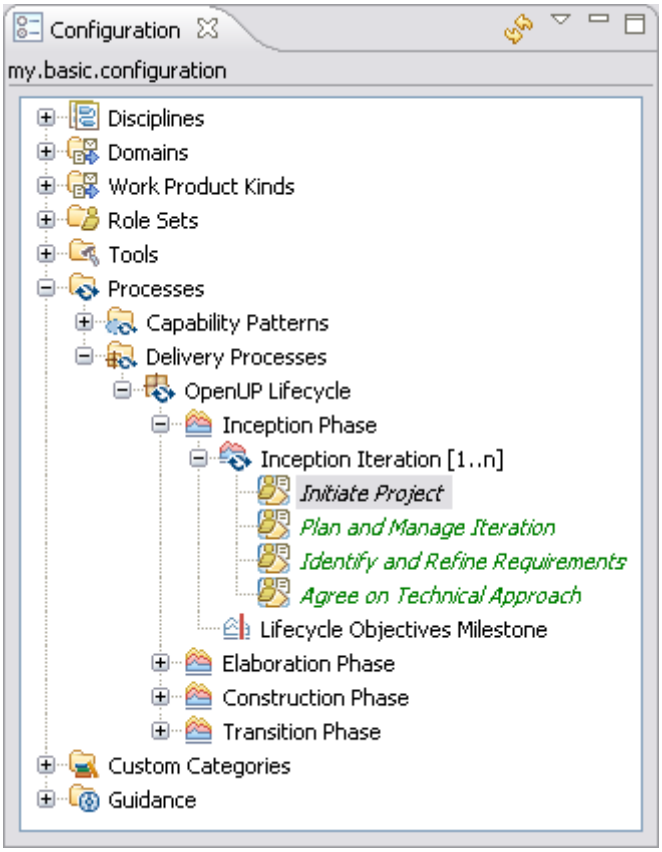
4.4.2. Browse Process Content

The goal of this exercise is to gain a basic understanding of processes.

To browse process content:

1. Switch to the **Browsing** perspective.
2. From the configuration menu in the main tool bar, click *my.basic.configuration*. If you do not have this configuration, go back and complete the *Add a plug-in to a configuration* exercise in the *Create method content* tutorial.
3. In the **Configuration** view panel, expand the **Processes** folder and then expand **Capability Patterns** → *Phase Iteration Templates*. Click the *Inception Phase Iteration* icon.
4. A capability pattern contains a large amount of information is displayed over four tabs:
   - Description
   - Work Breakdown Structure
   - Team Allocation
   - Work Product Usage

   Explore the information on each tab.
5. To view a delivery process, expand **Processes** → **Delivery Processes** → *OpenUp Lifecycle* → *Inception Phase* → *Inceptions Iteration [1..n]*. Click **Initiate Project**. The display for a delivery process is similar to a capability pattern.



4.4.3. Create a Delivery Process

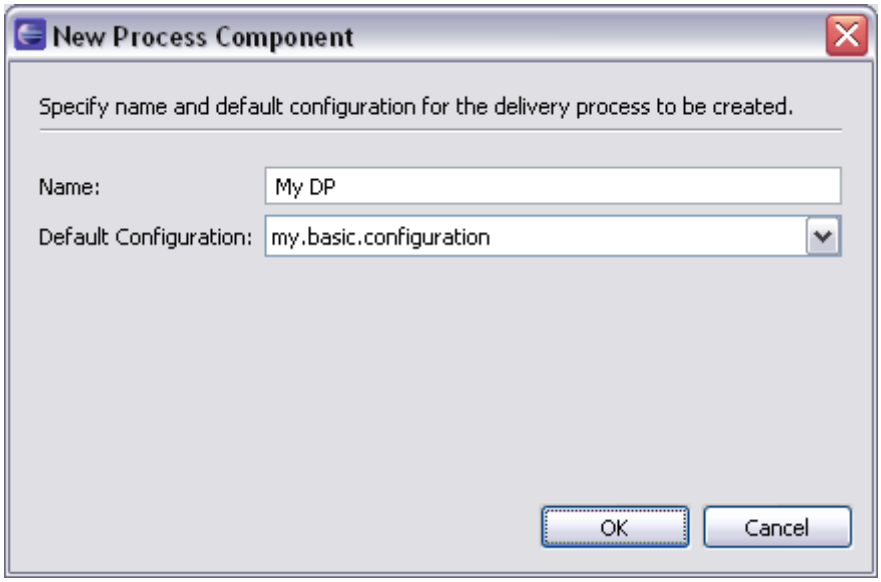
The goal of this exercise is to create a delivery process using core method content directly.

In process authoring the definitions of the method content elements are not included directly in the work breakdown structure, but local references termed descriptors are created, which refer back to the content elements in the method library. Descriptors contain the references to the original method elements as well as additional information that is relevant to the local process.

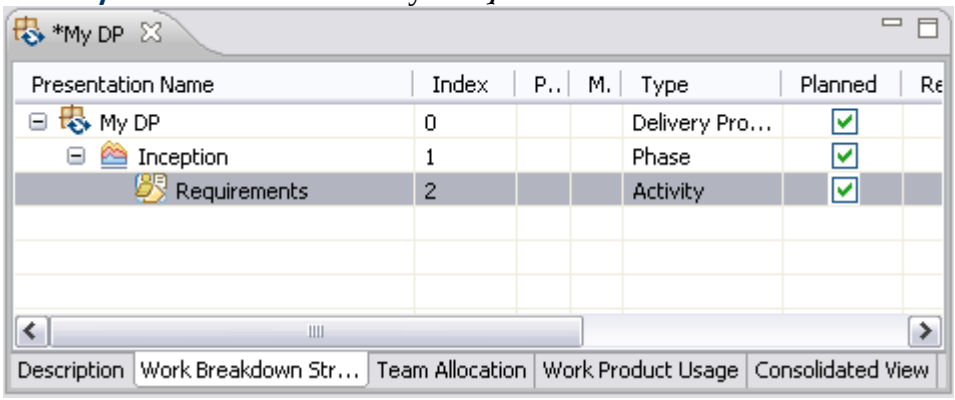
Descriptors provide a mechanism whereby relationships are defined in core method content authoring, such as roles associated with a task, input and output, and work products that can be defined or changed locally within the process.

To create a delivery process:

1. Make sure that you are using the Authoring perspective.
2. In the **Library** view, create a new delivery process under *my_plug-in* by right-clicking **Delivery Processes**, then clicking **New → Delivery Process**.
3. When you create a process, you are asked to provide a **name** and select **a default configuration** to be used with the process. Use the name "*My DP*" and select *my.basic.configuration*. If you have not created this configuration, select *publish.openup*.



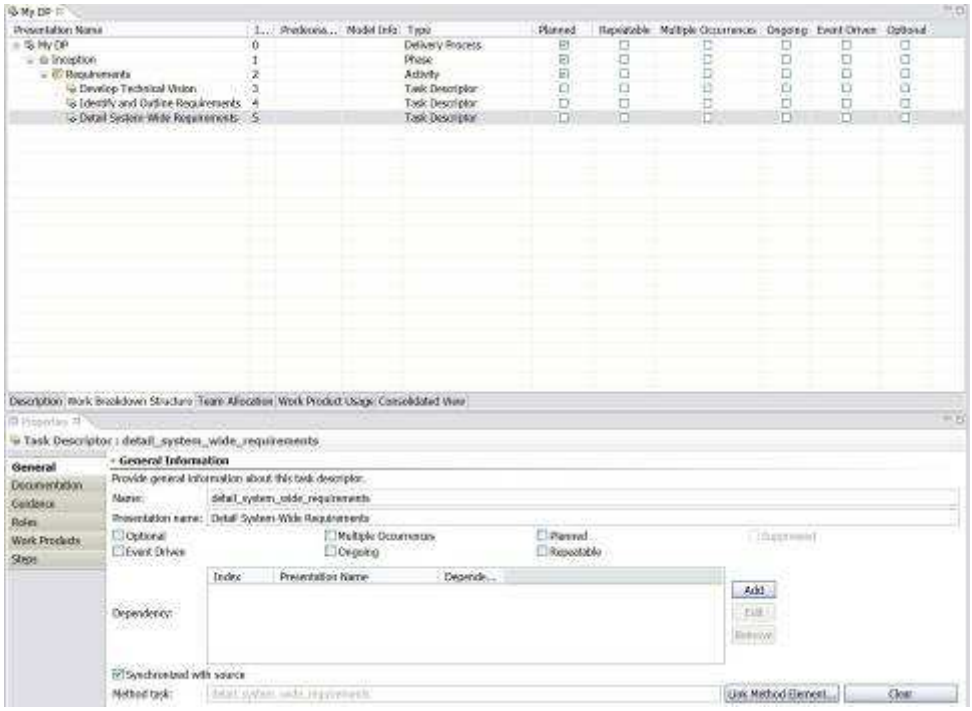
4. Click the **Work Breakdown Structure** tab.
5. To create an inception phase in the work breakdown structure view, right-click *My DP*, then select **New Child** → **Phase**. Call the phase "*Inception*".
6. We will create a "Requirements Management" activity under Inception. In the work breakdown structure view, right-click *Inception* and click **New** → **Child** → **Activity**. Name the new activity "*Requirements*".



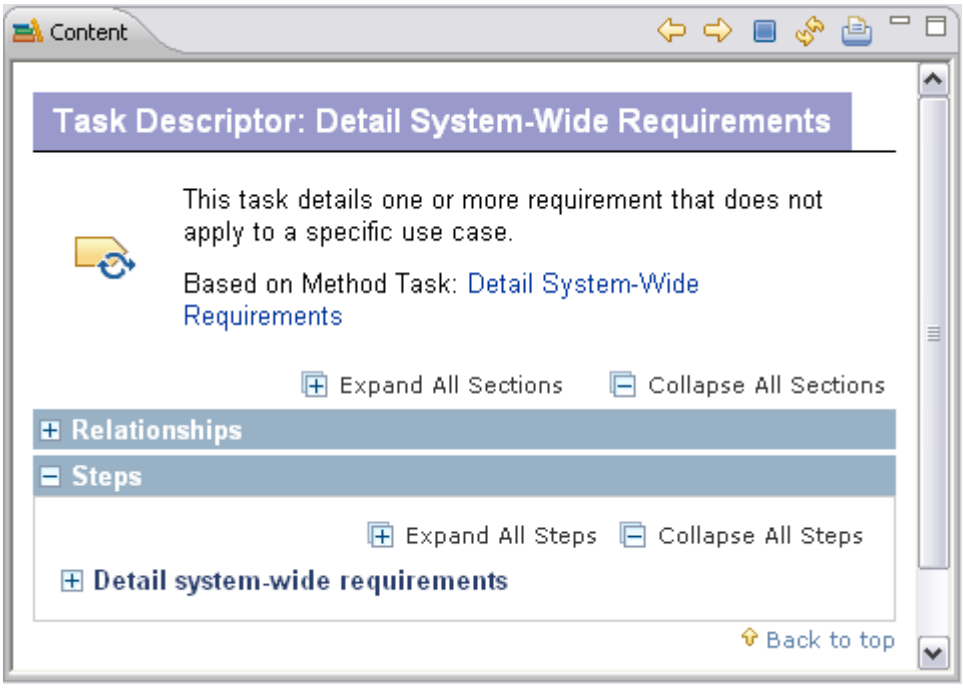
7. Now we are going to add some tasks to the Requirements activity. In the **Configuration** view, open **Disciplines** → *Requirements*. Drag and drop the following tasks onto the requirements activity that you just created:
   - *Develop Technical Vision*
   - *Identify and Outline Requirements*
   - *Detail System-Wide Requirements*

   When you include these tasks in your process, the associated roles and work products are also included in the process.
8. You can review the information for each method element in the **Properties** view[4]. You may have to select and right-click the method element, and select the **Show Properties View** to open the Properties view.

[4] The Properties View is the editor for activities, task descriptors, roles descriptors, and work product descriptors contained in processes.



9. Customise the steps from the *Detail System-Wide Requirements* task for the *Inception* phase. Click the *Detail System-Wide Requirements* task descriptor, and open the **Properties** view. Click the **Steps** tab.
10. Notice that there are three steps for this task:
    - *Detail system-wide requirements*
    - *Detail glossary terms*
    - *Achieve concurrence*
11. Remove the last two steps, close the **Properties** view and save the *My DP*.
12. Switch to the **Browsing** perspective and click **Processes > Delivery Processes** then double-click **My DP**. The **Work Breakdown Structure** tab is displayed.
13. Expand *Inception* → *Requirements* and click on *Detail System-Wide Requirements*. Only the first step (*Detail system-wide requirements*) from the core method task is included.



14. Make other changes to the tasks in the delivery process that you just created and see how they are reflected when you view the process in the browsing perspective.

4.4.4. Use Capability Patterns

The goal of this exercise is to create a new delivery process using capability patterns.

We will add some capability patterns to the delivery process that you created in the previous lesson. First we will copy a capability pattern, and then we will extend a capability pattern.

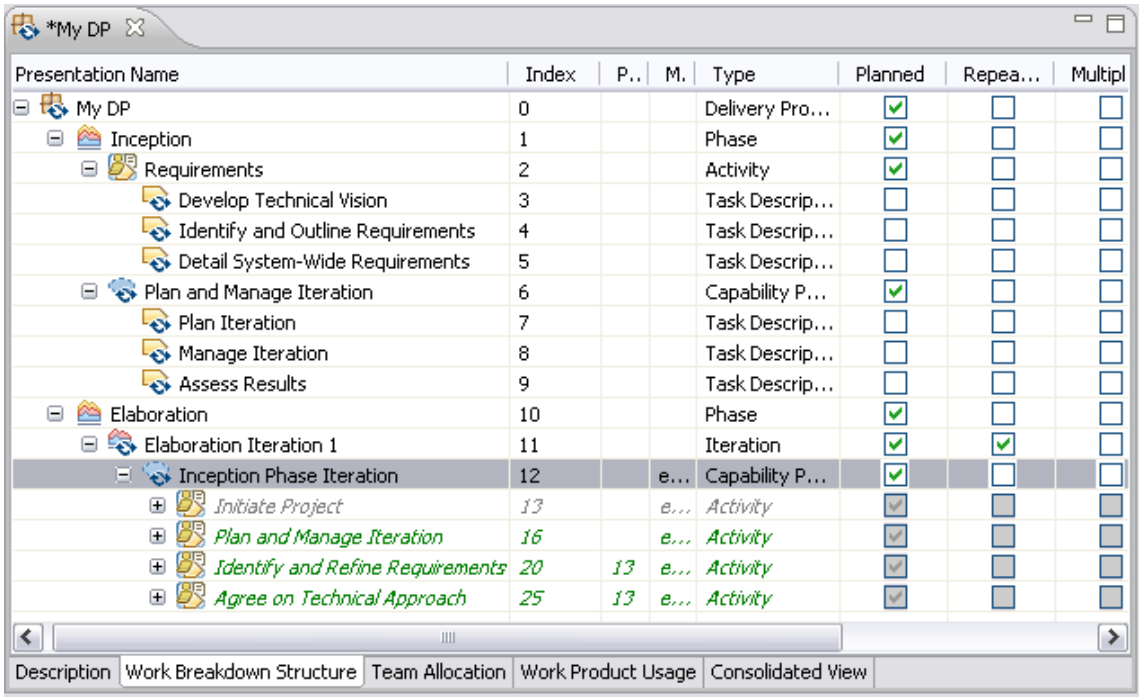
To add a capability pattern using Copy:

1. Switch to the **Authoring** perspective and open the *My DP* delivery process that you created earlier in this tutorial.
2. In the **Configurations** view, expand **Processes** → **Capability Patterns** → *Management*.
3. Drag and drop the *Plan and Manage Iteration* capability pattern onto the *Inception* phase in *My DP* delivery process. A menu opens, prompting you to select **Copy**, **Extend** or **Deep Copy**.
4. Select **Copy**.
5. Expand *Plan and Manage Iteration* capability pattern to the task descriptor level. Note that the task descriptor names are displayed in black. This indicates that the tasks are available for you to edit. **Note:** ***Any future changes to the capability pattern will not be propagated to your process***. You can edit the task in the **Properties** view.

To add a capability pattern using **Extends**:

6. Create a new phase in the *My DP* delivery process. Right-click the first row with *My DP* and select **New Child** → **Phase**. Call the phase "*Elaboration*".

7. Create a new iteration in the Elaboration phase. Right-click the *Elaboration* phase and select **New Child → Iteration**. Call the iteration "*Elaboration Iteration 1*".
8. In the **Configurations** view, expand **Processes** > **Capability Patterns** > **Phase Iteration Templates**.
9. Drag and drop the *Inception Phase Iteration* capability pattern onto the *Elaboration Iteration 1* iteration in *My DP*.
10. Select **Extend**.
11. Expand the *Inception Phase Iteration* capability pattern that you added to the task descriptor level. Note that the task names are displayed in green italic. This indicates that the task descriptor is in a part of a process defined elsewhere in a capability pattern. ***Subsequent changes to the capability pattern will be propagated to your process***. The **Properties** view shows the task details but they cannot be edited.
12. Right-click a task descriptor in the extended capability pattern and select **Suppress**. The task descriptor is disabled and the text colour becomes gray. The task descriptor will not appear in the published version of the delivery process, nor will it be included in an export of the process.



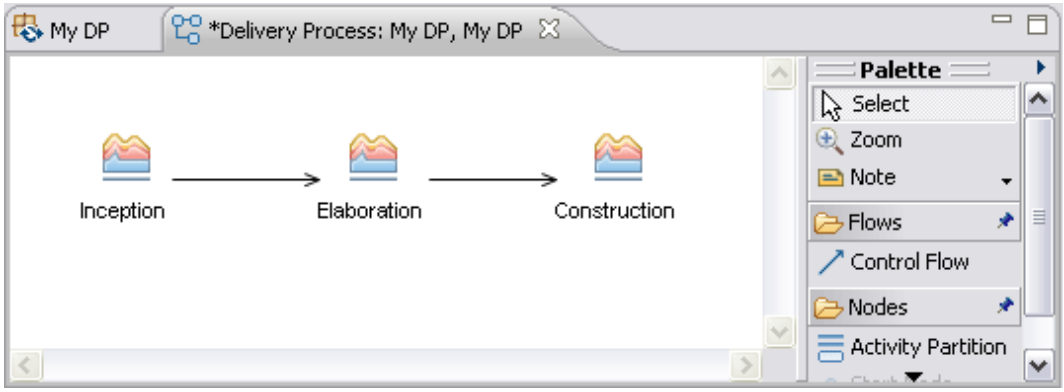
4.4.5. Create a Process Diagram

The goal of this exercise is to use diagram features.

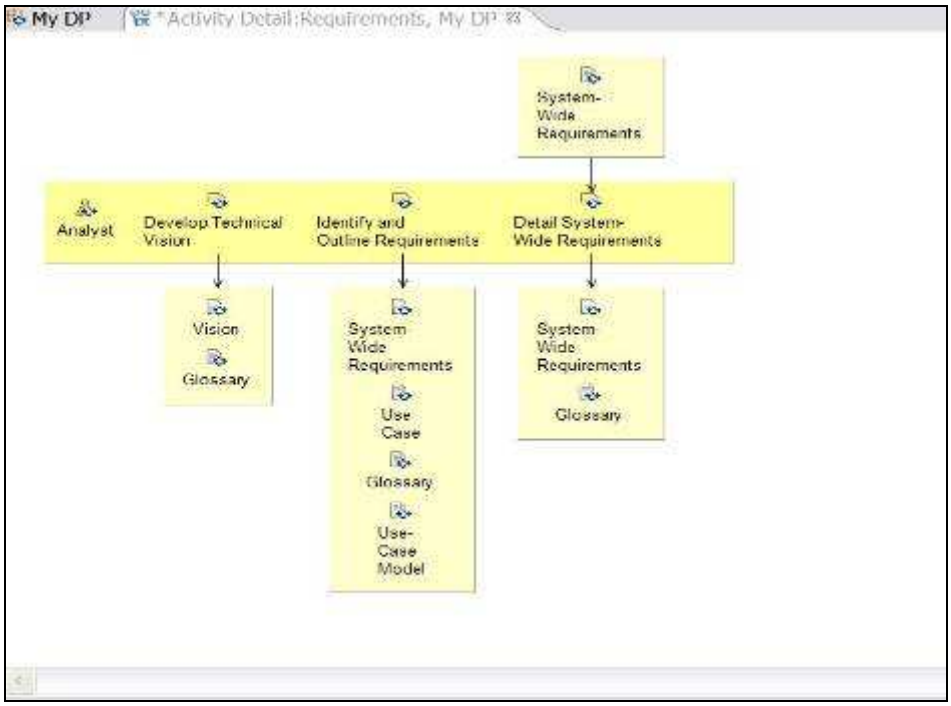
To create a process diagram:

1. Switch to **Authoring** perspective.
2. Open the *My DP* delivery process that you have been working on.
3. In the **Work Breakdown Structure** tab right-click *My DP* and select **Diagrams → Open Activity Diagram**. Click **OK** to create a new diagram. The diagram editor should open, showing the top-level activities (phases).

4. Expand the palette on the right of the diagram view if not already expanded. All elements required to create an activity diagram are available on the palette.
5. Right-click the diagram workspace and select **Add → Phase** to add a new phase called "*Construction*" to the diagram. The name will initially be *New Phase*. To change the name, right-click on the node, select **Show Properties View**, click on the **Advanced** tab; select the Name and edit the name field.
6. Using the **Control Flow** tool, create links between the phases to show the sequence of the phases. Click on Control Flow, click on the first node, keep the mouse button pressed and move the cursor to the second node and release the button. You may name the control flow link.



7. Switch back to the Work Breakdown Structure view of My DP and notice the new phase. Also notice that the links you added on the diagram are reflected in the **Predecessors** column.
8. Close the diagrams that you opened.
9. In the **Work Breakdown Structure**, right-click the **Requirements** activity and click **Diagrams → Open Activity Detail Diagram**. The Activity Detail Diagram is automatically generated. It shows the following:
   - The roles that perform tasks in the activity
   - The tasks that the roles perform
   - The input and output work products for each task



You can change the layout of the diagram but you cannot add or remove any elements from the diagram.

10. Close the diagram that you opened.
11. In the **Work Breakdown Structure**, right-click the ***Requirements*** activity and click **Diagrams → Open Work Product Dependency Diagram**. The ***work products used in the activity*** are automatically added to the diagram. Using the Work Product Dependency tool, add a dependency link between two work products. This indicates that the work product at the arrow end of the link is dependent on the other work product, that is, the other work product needs to be created in order to create the second work product.
12. Click the **Work Product Descriptor** tool in the palette and add a new work product descriptor to the diagram.
13. Switch back to the delivery process and open the **Work Product Usage** view. Notice the new work product in the ***Requirements*** activity.
14. After you have edited a diagram, you can choose to include it or exclude it when publishing the process in a configuration. In the Work Breakdown Structure, right-click the ***Requirements*** activity and select **Diagrams → Publishing Options**. Check or clear the types of diagram listed. If a diagram type is not listed, it means that you have not edited that diagram type for the activity yet.

4.5. Publish Method Content

This tutorial contains a brief summary of concepts followed by four guided exercises.

Learning objectives

Upon completion of this tutorial you should be able to do the following:

- Publish a predefined method configuration.
- Create a custom category.
- Publish a custom method configuration as a standalone Web site.

Time required

The estimated time to complete this tutorial is about 45 minutes.

Prerequisites

This tutorial uses content we created in the Create Method Content and Reuse Method Content tutorials. If you have not completed those tutorials you should do so before continuing.

Lessons in this module

- Concepts

  This topic provides background information needed to complete this tutorial.
- Publish A Method Configuration

  The goal of this exercise is to publish the method configuration that we created in an earlier exercise.
- Create a Custom Category

  The goal of this exercise is to use the custom categories feature.
- Create a Custom Method Configuration

  The goal of this exercise is to create a new customised method configuration. Later in this tutorial we will publish this configuration.
- Publish a Custom Method Configuration

  The goal of this exercise is to publish a method configuration.

4.5.1. Concepts

This topic provides background information needed to complete this tutorial.

EPF Composer is essentially a content management and publishing application. Its output is a Web site with method guidance and processes that can be used by a project team.

A method configuration is a structured set of content packages selected from different method plug-ins containing the method and process content that will be included in the published Web site. Upon publication, the method configuration is combined with different Navigation views that define the Web site's navigation structures and content organisation.

The **Browsing** perspective renders the method configuration content in Web (HTML) format, thereby providing a preview of the method configuration before it is published.

Using custom categories you can categorise content according to any scheme. Custom categories can be used to compose publishable views, thereby providing a means to organise method content for publishing.

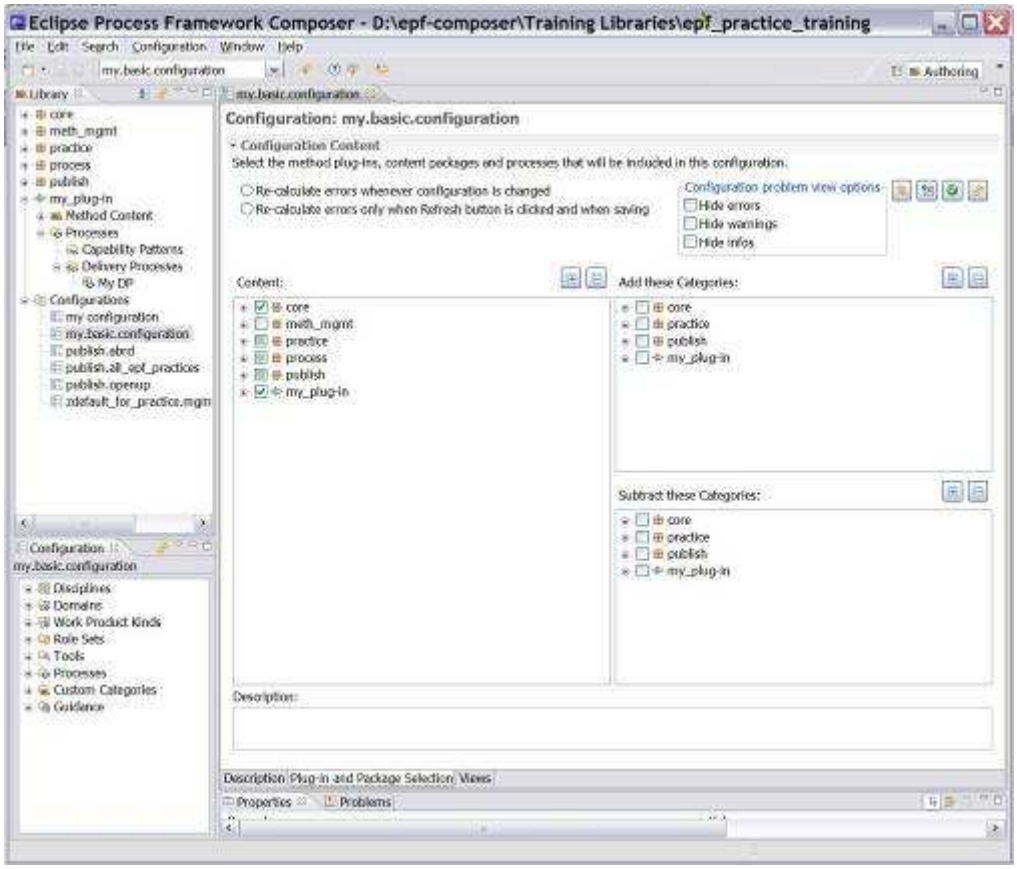
A **View** (**Navigation View**) is a ***custom category that is designed for publication***. Required packages and content elements are assigned to a custom category. The custom category can then be added as a view to a configuration.

4.5.2. Publish a Method Configuration

The goal of this exercise is to publish the method configuration that we created in an earlier exercise.

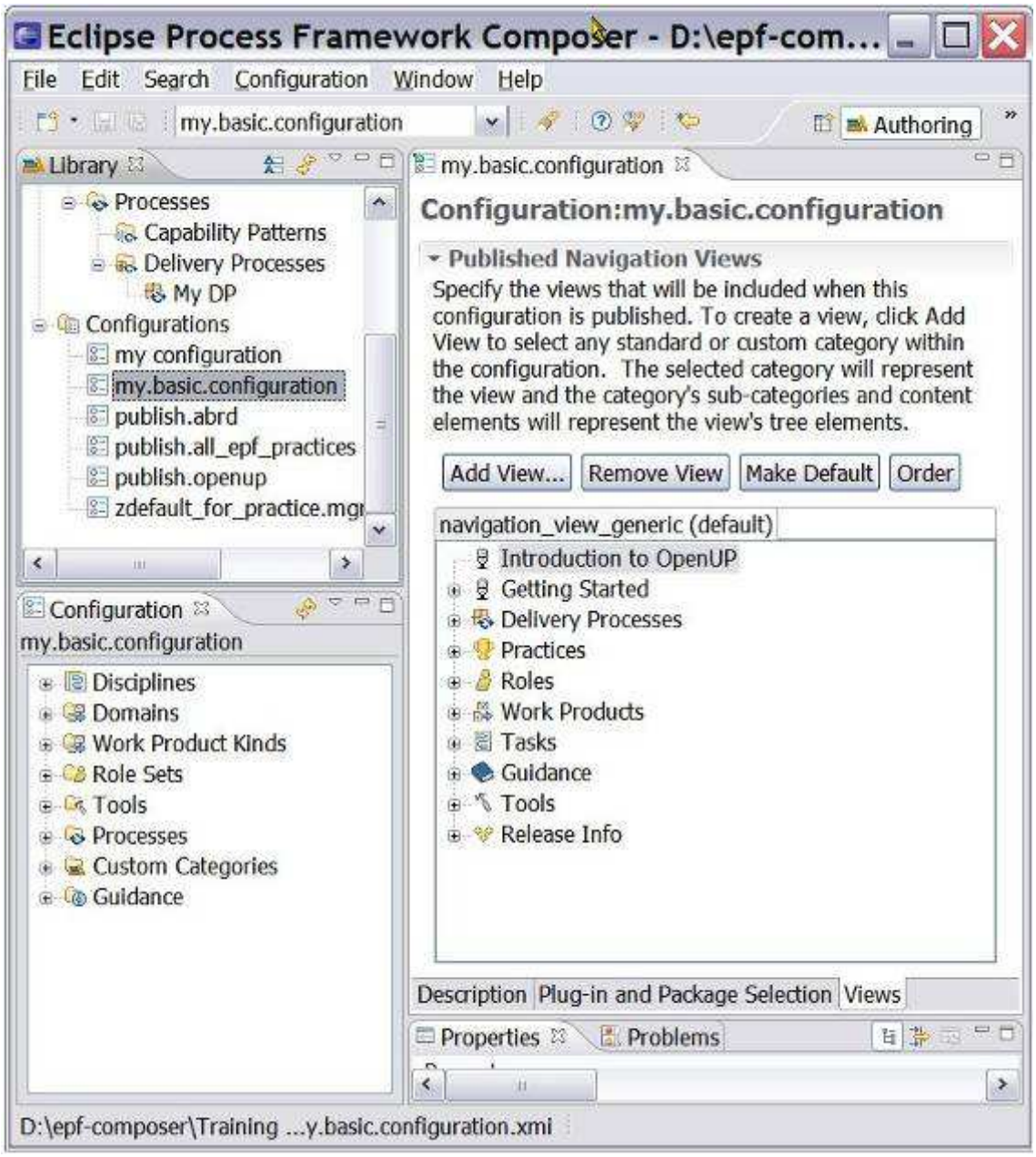
To publish a method configuration:

1. Switch to the **Authoring** perspective if you are not already using it.
2. Select the configuration "*my.basic.configuration*", if it is not already selected, using the configuration selection menu in the main tool-bar.
3. In the **Library** view panel, expand the **Configurations** folder and double-click *my.basic.configuration*. **The configuration editor** opens in the right panel, showing the **Description** tab.
4. Click on the **Plug-in and Package Selection** tab. The tab shows the method plug-ins and their content packages that have been included in the configuration.



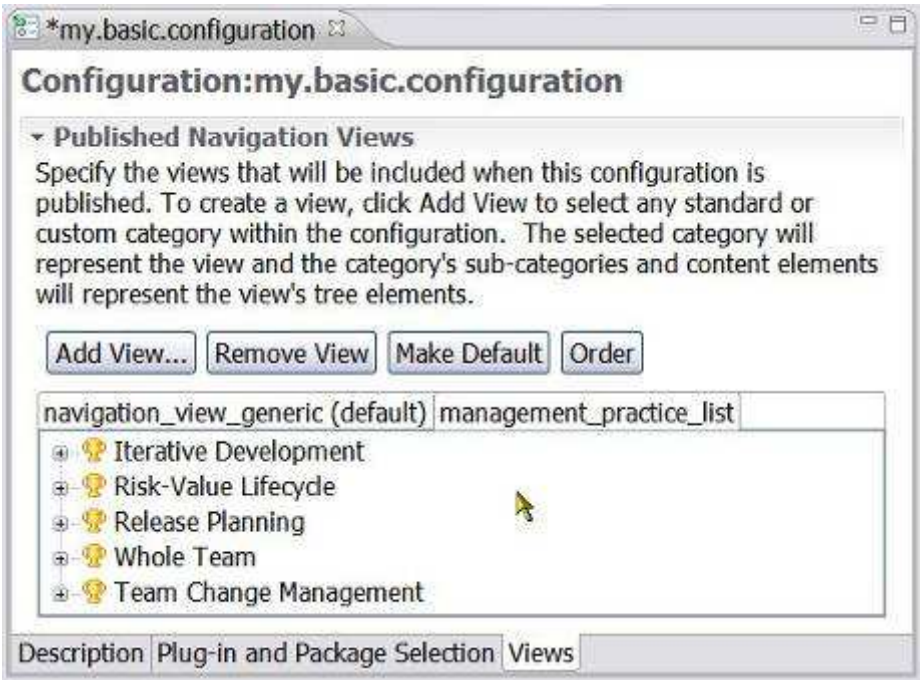
The selection for publication of plug-ins and packages in the configuration can be adjusted by selecting and clearing them in the tree display.

5. Expand the **Custom Categories** node in the **Configuration** view and locate the views associated with this configuration.
6. At the bottom of the **Configuration editor** panel, select the **Views** tab.



7. The only view shown is the "***navigation_view_generic***". This view resides in the method plug-in called "***core.default.nav_view.base***" under the **Method Content** → **Custom Category** node. When we earlier copied and pasted the method configuration ***publish.openup*** and named the copy ***my.basic.configuration***, this view was "inherited".
8. Click on **Add View** and the **Select Dialog: Select Categories** pops up with Method Element Type: **Custom Category** selected. Select Management Practices for example:

9. The view tab now contains two views (navigation views), the ***navigation_view_generic*** and the the ***management_practice_list***.



10. One of the views can be made the default view by selecting the **View** and clicking **Make Default**. The view selected will be the one displayed on top of all others when configuration is published as a Web site. To summarise: To create a view;

click **Add View** to select any standard or custom category within the configuration. The selected **category** will represent the view and the category's sub-categories and content elements will represent the view's tree elements.

11. Close the **Configuration** editor. You will be prompted to save your changes if you made any.
12. In the main tool bar, use the **Configuration** menu and select **Publish**.
13. Select *my.basic.configuration* and click **Next**.
14. Step through **Publish Method Configuration** wizard accepting the defaults and publish the method configuration. You may want to use your own title on the **Select publishing options** page. It will take a few minutes to generate the Web site. When the generation is complete, the Web browser applet launches, so that you can view the resulting published Web site. Accept the security warnings when the browser launches.

    Note: The published Web site has a richer organisation and layout compared to the simpler view of content shown in the Configuration view.

4.5.3. Create a Custom Category

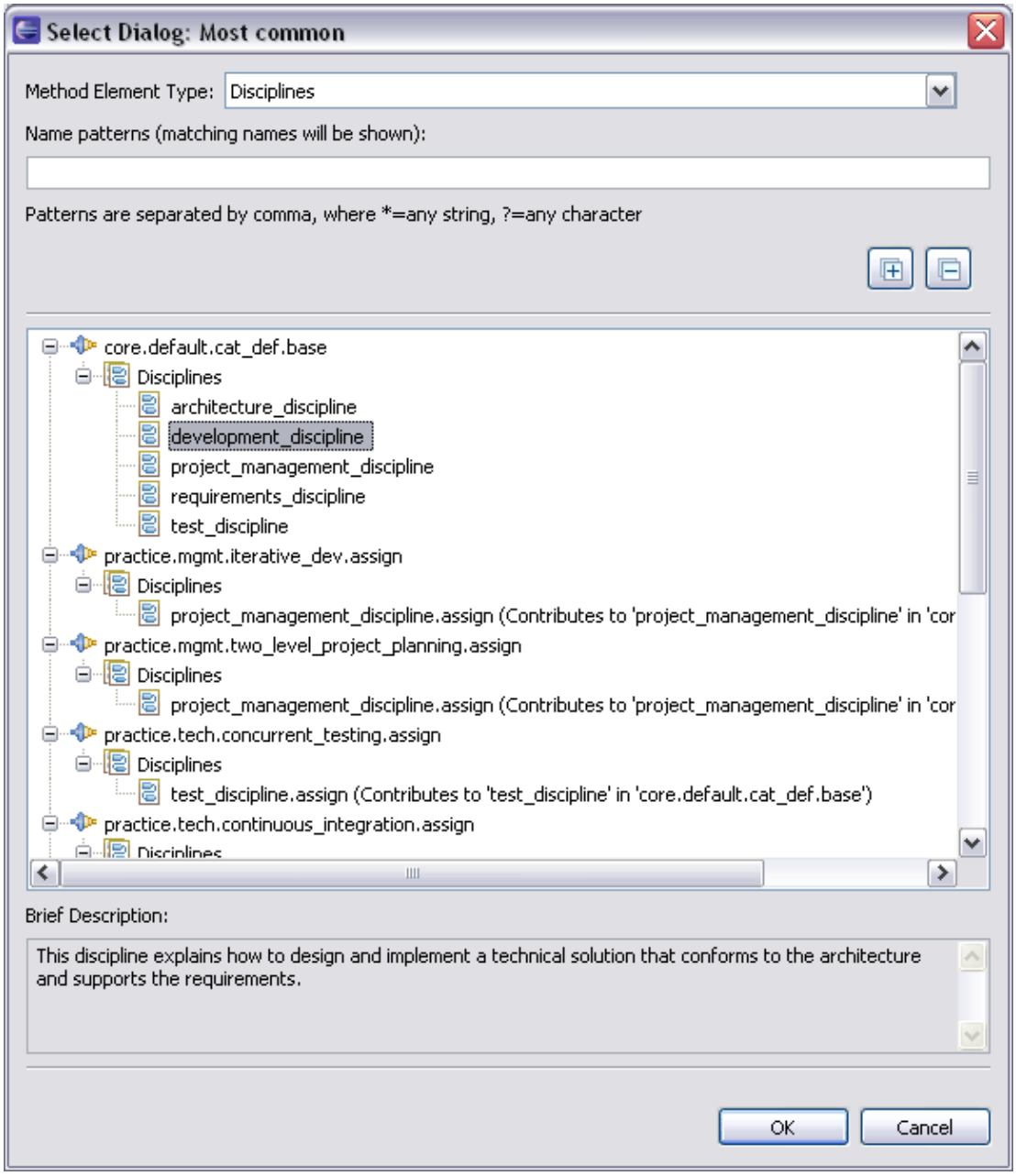
The goal of this exercise is to use the custom categories feature.

To create a custom category:

15. Make sure that you are in the **Authoring** perspective.
16. Expand the node in the **Library** view panel for *my_plug-in* then expand the **Method Content** folder. If you have not already created this method plug-in, create it now, or go through the Create Method Content[5] tutorial.
17. Create a custom category named "*my_custom_category*": Right-click the **Custom Categories** folder, select **New → Custom Category**.
18. Click the **Assign** tab and click the **Assign** button. The **Select Dialog: Most common** window opens.
19. If you have previously created content within **my_plug-in**, add your method content elements, such as **Roles**, **Artefacts**, **Tasks**, and **Guidance** into the custom category. If you do not have this package, add content from another area in the library.

    Remember: You can select multiple items by using Shift or Control key.
20. Add the *development_discipline* discipline, which is under **Disciplines** in the *core.default.cat_def.base* method plug-in. Click **OK**.

[5] This tutorial contains a brief summary of key concepts followed by ten exercises.



21. Close the editor panel and save your changes.
22. From the **Library** view double-click to open one of the elements that you assigned to the new custom category. Observe how the custom category is reflected in the **Categories** tab.

4.5.4. Create a Method Configuration

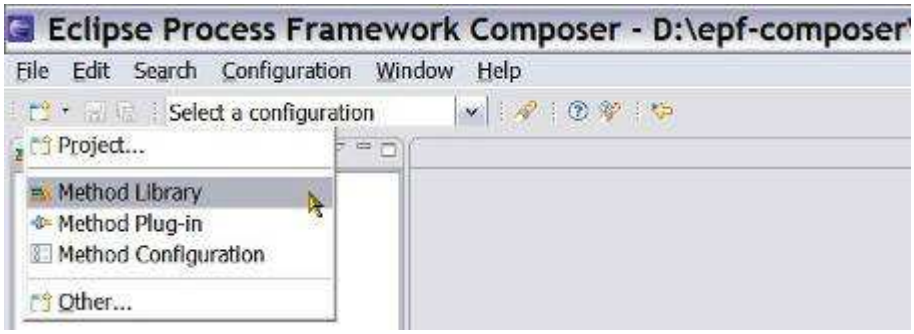
The goal of this exercise is to create a new method configuration. Later in this tutorial we will publish this configuration.

To create a method configuration:

1. Make sure that you are in the **Authoring** perspective.
2. There are three ways to create a new method configuration.
   a. Click **File** → **New** → **Method Configuration**.

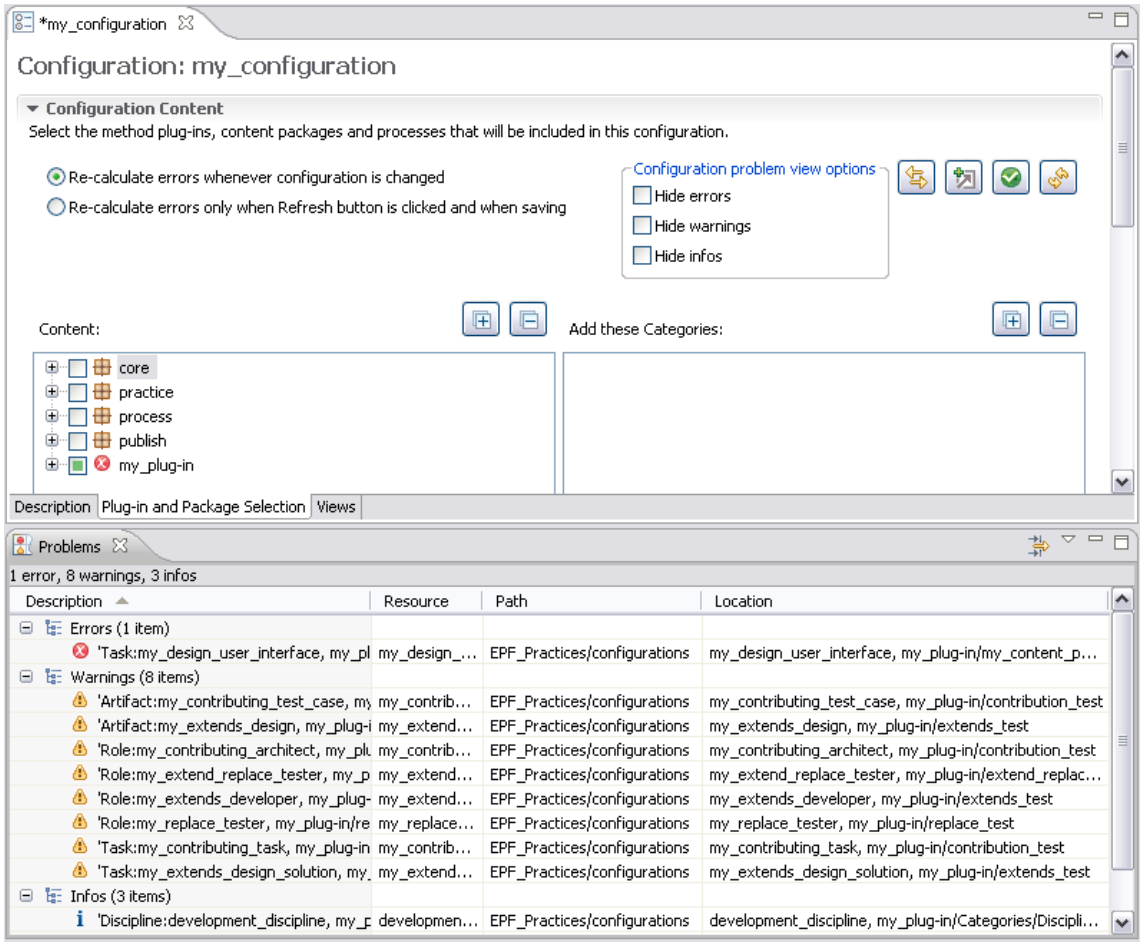
b. In the **Library** view, right-click the **Configurations** folder and click **New** → **Method Configuration**.

c. You can also click the Down Arrow of the '**New**' icon in the toolbar and select **Method Configuration** from the drop down list.



3. Name the new configuration "*my_configuration*".
4. Click the **Plug-in and Package Selection** tab. In the list of plug-ins, check *my_plug-in*, which you created in the Create Method Content tutorial. This method plug-in has OpenUp as a base and references parts of its content and has elements that extend and replace base content. If you do not have this method plug-in, complete the Create Method Content[6] tutorial before continuing.
5. When you select *my_plug-in*, some errors and warnings are displayed in the **Problems** view. If the Problems view is not shown, click on **Windows** on the main toolbar, then **Other** and select **Problems**. To fix these problems click the **Fix errors and warnings** button.

[6] This tutorial contains a brief summary of key concepts followed by ten exercises.



6. Drill down into the content packages in the *my_plug-in* plug-in by clicking the **+** signs. Clear the *extends_test* and *replace_test* content packages if you have created them in earlier tutorials. Clearing these avoids conflicts that may occur because the same base element is affected by the method plug-ins.
7. Click the **Views** tab.

   Any custom category can be added as a view to a configuration. You can create any number of custom categories. They are flexible in terms of the content that you can add to a custom category and in the order in which the content appears.
8. Click **Add View** and add *my_custom_category* as a view in the configuration. Experiment with adding other custom categories as views.
9. Make *my_custom_category* the default view by selecting the *my_custom_category* tab and clicking **Make Default**.
10. Save the configuration by closing the configuration editor.
11. In the configuration selection box at the top of the screen, click *my_configuration*. The items in the configuration view change.
12. Switch to the **Browsing** perspective. Explore the content in the Configuration view. You should see the content that you created in your method plug-ins and some inherited content.

4.5.5. Publish a Custom Method Configuration

The goal of this exercise is to publish a method configuration.

To publish a custom method configuration:

1. Use the **Configuration** menu in the main tool bar and click **Publish**.
2. In the list of configurations, click *my.basic.configuration* and then click **Next**.
3. Step through the **Publish** wizard, accepting the defaults, and publish the configuration. It will take a few minutes to generate the Web site. Assuming that you have the correct Java environment, the browser applet should launch so that you can view the published Web site. Accept any security warnings when the applet launches.

Note: The published site has a richer organisation and layout compared to the simpler view of the configuration content shown in the Configuration view.

Eclipse Process Framework (EPF) Composer

User Manual

5. Key Concepts

Contents

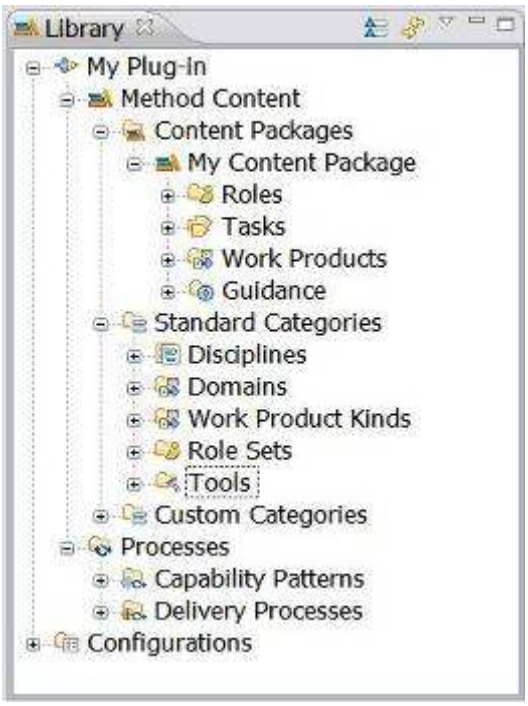
5.1. Method Library Schema

The EPF Composer imposes a strict schema, with a structure of "virtual folders", (nodes) being created automatically in response to user action. The folders are "virtual" to the extent that they do not always correspond to folders in the file system. We will often simply write "Folder" or "Node". Each of these will be explained in the subsequent chapters.

The word (Create) in the following table showing the Library structure indicates that the EPF Composer has created an empty folder to contain user-defined content. Schematically, the Library has the following structure:

Table 1 - Library Structure

What you are creating will be nested under the parent folder of the type you are creating, i.e. a role will be nested under the parent folder (node) Roles, a Task under the parent folder (node) Tasks and so forth, resulting in the following **Library view**:



All method plug-ins will have this schema.

5.2. Method Library

All method elements are stored in a method library. The method library contains 1) method plug-ins and 2) method configurations.

1. **Method Plug-ins**: All method content is contained in method plug-ins and all method content must be organised into one or multiple method plug-ins.
2. **Method Configurations**: A method library also has one or more method configurations that filter the library and provide smaller working sets of library content for the end user.

   A method configuration is a selection of method plug-ins and their content. Method configurations are used for (a) creating **processes** and (b) for defining which elements will be **published**:

   - ***Processes***: A method configuration defines a logical subset of a method library and when a process is created, a method configuration must be specified for the process. A method configuration is the list of method plug-ins used to generate a specific instance of process guidance. The process will be created or "authored" against the specified method configuration.
   - ***Published***: The method configurations limit the view to a subset of the library, i.e. the selections made in the method configuration determine the content of the published Web site.

To illustrate the relationship between plug-ins and configurations, it can be argued that a method library is analogous to a warehouse full of parts that are used to assemble various products. Method configurations are built from subsets of elements in the method library. Method configurations represent the various products, such as cars, that can be assembled from parts in the warehouse. While most cars require unique parts that are used by a specific model, but the warehouse has large numbers of parts that can be used in more than one model.

When a new method library is created, it has initially no method plug-ins and an empty configuration.

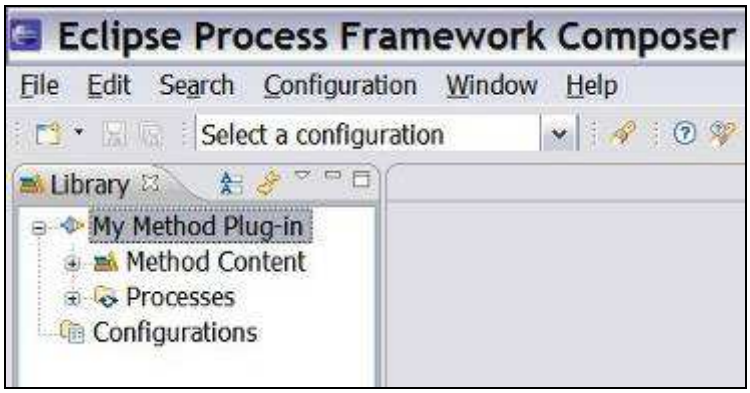
Related topics

Method Content Elements
Content Categories
Capability Patterns
Method Configurations

5.3. Method Plug-in

All method content is organised in method plug-ins, contained in a method library. A method plug-in is a container for method[7] and process[8] content. The method plug-in has the following structure or schema:

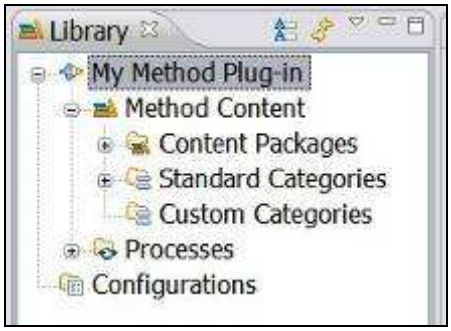
[7] Method content provides step-by-step explanations, describing how specific development goals are achieved, independent of the placement of these steps within a development lifecycle. Processes take these method elements and relate them into semi-ordered sequences that are customised to specific types of projects.



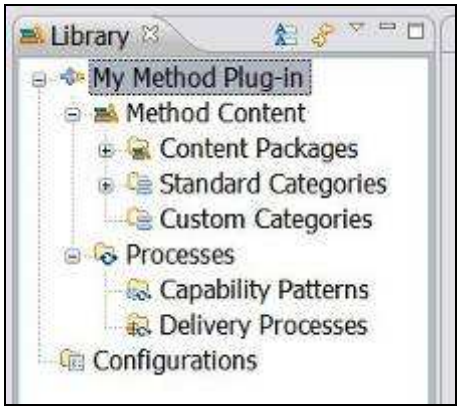
Method plug-ins contain a Method Content sector, represented with the icon and a Processes sector, represented with the icon . The Method Content schema is structured in:

- Content Packages
- Standard Categories
- Custom Categories[9]

The Library view shows the Method Content structure:



Processes are structured into process fragments called capability patterns and full lifecycle processes called delivery processes.



[8] A process describes how a particular piece of work should be done. The work may have a relatively small scope, in which case it can be described as a capability pattern, or may address a full project lifecycle, in which case it can be described as a delivery process. A process can reuse method elements and combines them into a structure and sequence for carrying out work.

[9] Method content and process elements are organised into logical categories. The categories can appear in your final, published Web site as navigation views. There are two types of category: standard and custom.

A method plug-in can be standalone and without reference to other plug-ins. It can also reuse the content created in other plug-ins by referencing them and modifying or extending the content; or adding its own content to the other plug-ins.

A method plug-in can also play a purely supporting role. Supporting method plug-ins provide reusable content for other method plug-ins. The content stored in a supporting method plug-in is only visible and published for a method configuration if other content, which is not in a supporting plug-in, references it.

For example, you maintain a list of standardised work products for your organisation in a supporting method plug-in. You only want to publish these work products when they are actually used by other plug-ins, such as the ones that define the method content and processes for a specific category of projects. Hence, if any elements in these plug-ins such as a task modelling having a relationship to such a work product as an input or output, then the work product will be published. Other work products of that supporting plug-in that are not used by your content will not be published.

Architectural Reference Model

The method library can contain any number of method plug-ins and you will have to decide how to split your subject area into smaller sub-areas, each sub-area represented by a method plug-in. Each sub-area can then be developed independently with each sub-area represented by a method plug-in, and the sub-areas can subsequently be aggregated to form a complete solution, represented by the method library. The approach imply an architecture-based reference model for the larger subject area. The method plug-ins you create will mimic the architecture reference model you have chosen.

Related topics

Method Content Packages[10]
Create a Method Plug-in[11]

5.4. Method Content Elements

Method content elements describe roles, the tasks that they perform, the work products that are used and produced by those tasks, and supporting guidance. Method content elements provide step-by-step explanations, describing how specific development goals are achieved, independently of the placement of these steps within a process lifecycle. Processes take these method elements and relate them into semi-ordered sequences that are customised to specific types of projects.

The four method content elements are:

- Tasks
- Roles
- Work Products

[10] A method content package is a container for method elements. Elements are organised in method packages to structure a large scale of method content and processes and to define a mechanism for reuse.

[11] All content is organised in method plug-ins. With method plug-ins and method packages, you can organise your content at a level of granularity that meets your needs for authoring and reusing content.

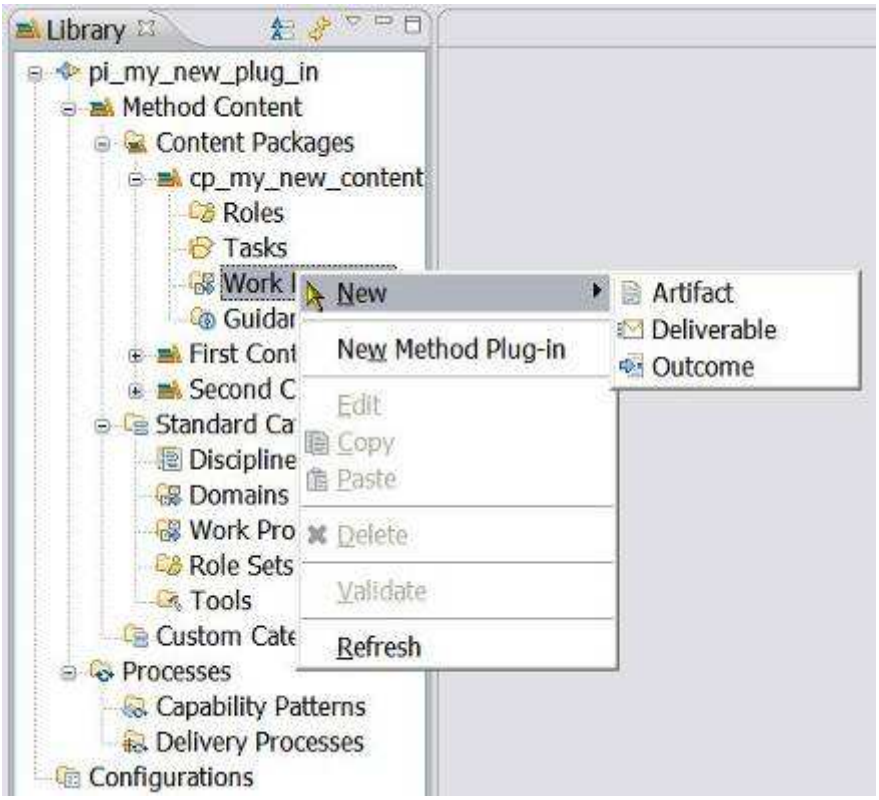
- Guidance

A Process Engineer ***authors these elements***, defines the ***relationships*** between them, and then ***categorises*** them.

For example, a software development project that develops an application from scratch performs development tasks such as "Develop Vision" or "Use Case Design" similar to a project that extends an existing software system. However, the two projects perform the tasks at different points in time and with a different emphasis, that is, they perform the steps of these tasks at different points of time and perhaps apply individual variations and additions.

Method content elements are contained within method Content Packages that, in turn, are contained within Method Plug-ins. In order to separate your own content from any original industry standard library content, you should always create new method content in a method plug-in that you produce. Creating method content in a method plug-in of your own also you to update your standard library with new releases of the industry standard library without affecting the content that you have created in your own plug-ins.

There are three types of **work products**: artefacts, outcomes and deliverables.



An **artefact** is a tangible work product that is consumed, produced, or modified by one or more tasks. Artefacts may be composed of other artefacts. An **outcome** is an intangible work product that may be a result or state. It may also be used to describe work products that are not formally defined. A **deliverable** is a collection of work products, usually artefacts, used to define typical or recommended content in the form of work products packaged for delivery.

5.5. Guidance Elements

In addition to roles, tasks and work products, EPF Composer supports the addition of guidance elements. Guidance elements are supplementary free-form documentation such as documented related to specific practices, white papers, concept descriptions, guidelines, templates, examples, and so on.

Guidance is a general term for supplemental information that can be added to most **Method and Process elements**. Guidance elements can also be associated with other guidance elements.

Guidance elements can be attached to the following method **content elements**:

- Work products
- Tasks
- Roles

Guidance elements can be attached to the standard method **content categories**:

- Disciplines
- Domains
- Work Product Kinds
- Role Sets
- Tools

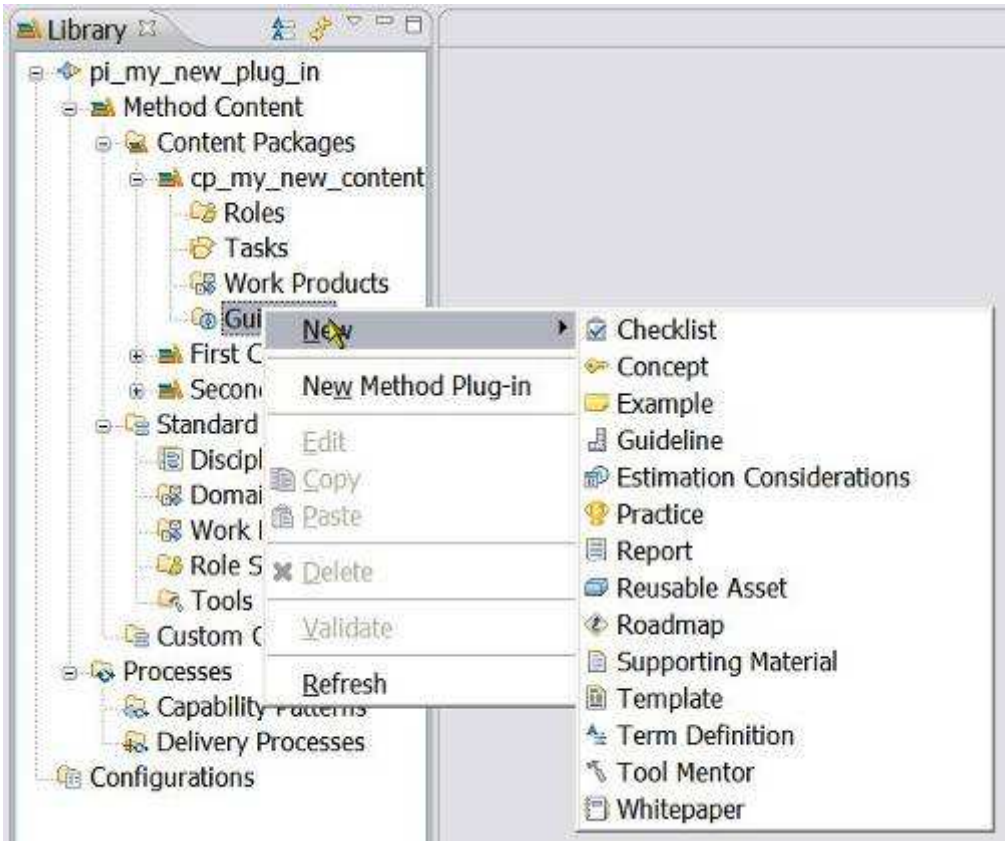
Guidance elements can be attached to the following **process elements**:

- Processes (Capability Patterns and Delivery Processes)
- Activity Descriptors (Iterations, Phases and Activities)
- Task Descriptors[12]

Since the guidance elements can applied to either method or process elements, they straddle both sides of the equation, aligning method content with their use in processes.

[12] The core content of activities is task descriptors and the steps of which they are composed. In addition, activities also contain role descriptors, work product descriptors, and milestones.

The list of the **fourteen types** of guidance elements:



The fourteen guidance element types: Checklist, Concept, Example, Guideline, Estimation Consideration, Practice, Report, Reusable Asset, Roadmap, Supporting Material, Template, Term Definition, Tool Mentor and White Paper.

Adding guidance is an easy way to tailor information for specific projects. For example, a type of Guidance called a Guideline can be associated to a Work Product that explains how your project uses that Work Product. For more information about attaching Guidance elements to specific types of elements, see Variability Associations.

Types of guidance

Table 2 - Guidance Element Types

Term	Description
Checklist	▪ Identifies a series of items that need to be completed or verified. Checklists are often used in reviews, such as walkthroughs or inspections.
Concept	▪ Outlines key ideas associated with basic principles underlying the referenced item. Concepts normally address more general topics than guidelines and span across several work product, tasks, or activities
Example	▪ Examples provide a model, a representative form or a typical example of a completed work product.
Guideline	▪ Provides additional detail about how to perform a particular task or grouping of tasks, or that provides additional details, rules, and recommendations about work products and their properties.

Term	Description
	▪ Among others, a guideline can include details about best practices and different approaches for doing work, about how to use particular types of work products, information about different subtypes and variants of the work product and how they evolve throughout a lifecycle, discussions about skills the performing roles should acquire or improve upon, and measurements for progress and maturity.
Estimation Considerations	▪ Provides sizing measures or standards for sizing the work effort associated with performing a particular piece of work and instructions for their successful use. It may be comprised of estimation considerations and estimation metrics.
Practice	▪ Represents a proven way or strategy of doing work to achieve a goal that has a positive impact on the work product or process quality. ▪ Practices are defined orthogonally to methods and processes. They could summarise aspects that impact may different parts of a method or specific process.
Report	▪ A predefined template of a result that is generated on the basis of other work products as an output from some form of tool automation. An example for a report would be a use case model survey that is generated by extracting diagram information from a graphical model and textual information from documents and combines these two types of information into a report.
Reusable Asset	▪ Provides a solution to a problem in a given context. The asset may have a variability point, which is a location in the asset that might have a value provided or customised by the asset consumer. The asset has rules for usage that are the instructions describing how the asset should be used.
Roadmap	▪ Describes how a process is typically performed. Processes can often be much easier understood by providing a walkthrough of a typical instance of the process. ▪ In addition to making the process practitioner understand how the work in the process is being performed, a roadmap provides additional information about how activities and tasks relate to each other over time.
Supporting Material	▪ Used as a category for other types of guidance that are not specifically defined elsewhere. It can be related to all kinds of content elements, including other guidance elements.
Template	▪ Provides for a work product a predefined table of contents, sections, packages, headings, a standardised format, in addition to descriptions about how the sections and packages are supposed to be used and completed. Templates cannot only be provided for documents, but also for conceptual models or physical data stores.
Term Definition	▪ Terms define concepts used to enhance the Glossary. A term definition is not directly related to content elements, but its relationship is being derived when the term is used in the content elements description text.
Tool Mentor	▪ Shows how to use a specific tool to accomplish some piece of work, in the context of, or independently from, a task or activity.

Term	Description
White Paper	▪ A concept guidance that has been externally reviewed or published and can be read and understood in isolation of other content elements and guidance.

Related topics

Method Content Categories[13]
Create Method Content
Create a Method Plug-in[14]
Variability Associations
Create Guidance[15]

5.6. Method Content Packages

A method content package is a container for method content elements (Tasks, Roles, Work Products and Guidance). Elements are organised in method content packages to structure a large amount of method content and processes and to define a mechanism for reuse.

Method elements from one content package can ***reuse elements*** from other content packages by defining a **link** between them. For example, a work product defined in one content package can be used as an input for tasks defined in another content package, ensuring that no redundant definitions of the same elements are required. In addition, maintenance of method content is greatly improved as changes can be performed in only one place.

Although a method content package is a container for method elements, its structure is enhanced by assigning standard and custom categories to its elements.

Decomposition

A method content package is a container for method content elements:

- Tasks
- Roles
- Work Products
- Guidance

How many containers (Content Packages) should there be, ideally? The question about decomposition and granularity was already an issue in the discussion about how many method plug-ins is the right amount and the same question extends to Content Packages. Just as there can be any amount of method plug-ins in the Method Library, there can be any number of Content Packages in each method plug-in.

---

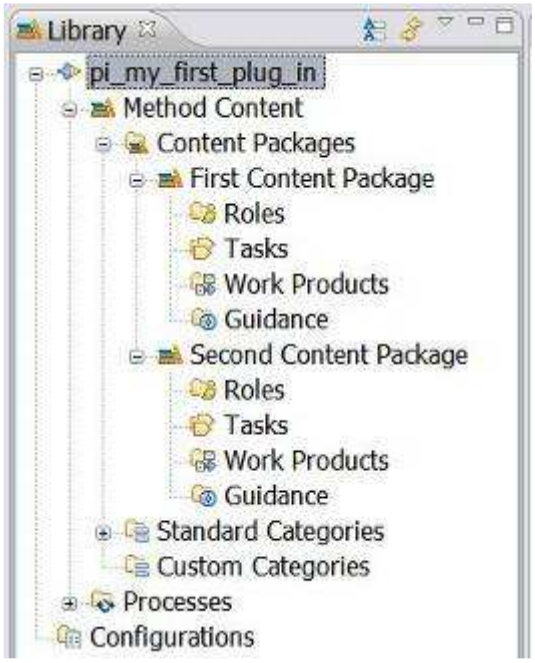
[13] Method content and process elements are organised into logical categories. The categories can appear in your final, published Web site as navigation views. There are two types of category: standard and custom.

[14] All content is organised in method plug-ins. With method plug-ins and method packages, you can organise your content at a level of granularity that meets your needs for authoring and reusing content.

[15] Guidance provides information about how to perform a role, how to create a work product, how to perform your task, and so on.

The question about the number and composition of Method Plug-ins and Content Packages is particularly complex because of the method content variability feature of the EPF composer, mimicking the object derivation in the software engineering. The method plug-ins can reference other plug-ins, thus extending them. The method content elements in one content package can be linked to other method content elements in other content packages with the link having a variability type and in addition, some method content elements can specify other method content elements as mandatory input or output. It is an architectural issue, with considerations like semantic consistency, coherence and coupling being central considerations.

Multiple Content Packages:



Related topics

Create a Content Package[16]
Create a Method Plug-in
Create Method Content

5.7. Method Content Variability

Method content variability allows method content elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional. If an element supports variability, the specification is shown at the bottom of the element's Description view.

You can use variability to customise configurations that use method content and processes that you do not own and cannot directly modify. When these content packages are upgraded, you can import them and then reapply the customisation that you made earlier in a single step, without going through each element

[16] Method content is organised into content packages that are contained in method plug-ins.

Variability generally affects two characteristics of a method element; its **attributes** and its **relationships** with other content elements. There are therefore three factors to be considered when using variability:

- **Attributes**: Element data types such as Main Description.
- **Incoming Associations**: Associations from other elements. The associated element may have one or more references to the subject element.
- **Outgoing Associations**: Associations to other elements. The subject element may have one or more references to the associated element.

For a complete list of supported associations for each type of element, see Associations Impacted by Variability.

Variability Type

Variability type describes how one element affects another through variability associations. **The five types of variability associations** are:

Not Applicable

The element is a base element and does not affect another element through variability. This is the default value of an element's variability type.

Contributes

A contributing element adds to the base element. Contributes provides a way for elements to contribute with their properties to their base element without directly changing any of its existing properties, such as in an additive fashion.

The base appears in the published Web site but the contributing element does not. In and out relationships from the contributing element are added to the base. Text from the contributing element is appended to corresponding base sections.

When the contribution is resolved during publication or by using the Browsing perspective, the base element is logically replaced with an element that combines the attributes and associations of the contributing element with the base element.

Replaces

A replacing element replaces parts of the base element. The Replaces variability association provides a mechanism for an element to replace a base element without directly changing any of the base element's existing properties. The replacer appears in the published Web site but the base element does not. “Out Relationships” in the replacer are left untouched, and those of the base element are ignored. “In Relationships” from the base are added to the replacer. Text in the replacer is left untouched, and the base element's text is ignored.

Extend

An extending element inherits characteristics of the base element. Both the extender and the base element appear in the published Web site. Out relationships from the base are added to the extender. In relationships in the extender are left untouched, the base element's are ignored. Text is added from the base if the extender has no value defined for the given section.

Extends associations provide a mechanism for method plug-ins to reuse elements from a base plug-in through a kind of inheritance. Attribute values and

associations are inherited from the "based-on" element to the extending element. The result is that the extending element has the same properties as the "based-on" element, but might define its own additions.

Extends and Replaces

This variability relationship combines the effects of extends and replace variability into one variability type. Whereas the replaces variability completely replaces all attributes and outgoing association instances of the base variability element with new values and instances, or removes all values or association instances if the replacing element does not define any, extends and replaces variability only replaces the values that have been redefined and leaves all other values of the base element as is.

Browsing Variability Relationships

You can use the graphical display to navigate to any of the variability elements; the complete hierarchy of variability associations for an element can be displayed graphically.

Related topics

Associations Impacted by Variability
Browsing Variability Relationships[17]
Contributes Variability[18]
Replaces Variability[19]
Extends Variability[20]
Extends and Replaces Variability
Category Variability
Activity Variability
Create Capability Patterns
Capability Patterns Reuse

5.8. Method Content Categories

The EPF Composer allows you to categorise your content based on a set of predefined categories, called standard categories. You can for example categorise your tasks into development disciplines, or your work products into domains. You can also create your own categorisation schemes, called custom categories, for your content with your own user-defined categories that allow you to index the content in any way you want.

---

[17] You can use the graphical display to navigate to any of the variability elements.

[18] A contributing element adds to the base element. Contributes provides a way for elements to contribute their properties into their base element without directly changing any of its existing properties, such as in an additive fashion. When the contribution is resolved during publication or by using the Browsing perspective, the base element is logically replaced with an element that combines the attributes and associations of the contributing element with the base element.

[19] The Replaces variability association provides a mechanism for an element to replace a base element without directly changing any of the base element's existing properties.

[20] Extends associations provide a mechanism for method plug-ins to reuse elements from a base plug-in through a kind of inheritance. Attribute values and associations are inherited from the "based-on" element to the extending element. The result is that the extending element has the same properties as the "based-on" element, but might define its own additions.

Categories are used to create navigation views for the final, published Web site.

Standard Categories

Standard categories provide a means to categorise method content in line with best practices for creating structured methods.

Unlike custom categories, standard categories, by definition, are linked to specific types of method content. There are standard categories for grouping tasks into disciplines, roles into role sets and work products into domains. The categories can only be assigned to the specified types of method content, for example, disciplines can only contain tasks.

Method Content	Standard Categories	Description
Tasks	**Disciplines**	▪ A discipline is a collection of tasks that are related to a major area of concern within the overall IT environment. ▪ You can organise disciplines by using discipline groupings.
Work Products	**Domains**	▪ A domain is a logical hierarchy of related work products that are grouped together based on timing, resources, or relationship. ▪ A domain categorises many work products, but a work product can only belong to one domain. You can divide domains into sub-domains.
	Work Product Kinds	▪ A work product can have many work product kinds. For instance, you might want to have a series of work product kinds that correspond to the overall intent of work products,

Method Content	Standard Categories	Description
		such as specification, plan, or model.
Roles	**Role Sets**	▪ A role set is used to group roles with certain commonalities. ▪ For example, you can set up a role set named "Analyst" to group together roles such as Business Process Analyst, System Analyst, and Requirements Specifier. Each of these roles work with similar techniques and have overlapping skills, but might be responsible for performing certain tasks and creating certain work products. ▪ Role sets can be organised by using role set groupings.
Tool Mentor (Guidance)	**Tools**	▪ The tools type is a container for tool mentors. A tool mentor is a type of guidance that shows how to use a specific software application to accomplish a piece of work. ▪ Tools can also provide general descriptions of a tool and its general capabilities.

Custom Categories

You can categorise content according to any scheme using custom categories. Custom categories are used to compose **Navigation Views**, thereby providing a means to organise method content for publishing. A Navigation View is a custom category that is designed for publication. Required content packages and content elements are assigned to a custom category. The custom category can then be added as a view to a method configuration, showing the required content packages and content elements assigned to that custom category.

Custom categories can also be displayed with the elements that they are categorising. For example, you could create a custom category that logically organises content that is related to your development organisation department, such as a Testing category that groups together all roles, work products, tasks, and guidance elements that are related to testing.

You can organise custom categories in a hierarchy, which means that you can create a category as a child of another category. Child categories can be referenced by more than one parent category.

Related topics

Method Configurations

5.9. Method Configurations

A method configuration is a selection of method plug-ins and method packages in a method library.

A method configuration defines a working set of packages within the method library that limits your view to a subset of the library. Elements that comprise the selected

configuration are displayed in the Configuration view[21]. Method configurations are used for creating processes and for publication by defining which elements are published or not published on the Web site in HTML format.

A method configuration consists of the components:

- A ***description*** of the configuration.
- A ***selection*** from the set of *plug-ins and packages* of the elements that will be part of the configuration.
- A ***selection*** of *categories* for the categorised elements that will be ***added*** to the set of elements of the configuration, in addition to the elements of the selected plug-ins and packages.
- A ***selection*** of *categories* of which categorised elements are ***subtracted*** from the set of elements of the configuration defined earlier.
- A ***selection*** of *views* to be published in the Web site.

In a method configuration, you are able to select and deselect content packages, process, and categories available in the method library's set of plug-ins. The selections that you make help determine the content of your published Web site. A configuration is given a name and then saved so that it can be changed and republished later. Before creating a method configuration, assess your needs and goals for the configuration.

There are two ways to create a method configuration:

- Create a new method configuration.
- Create a method configuration by copying an existing configuration.

Configurations can be created by selecting plug-ins and packages and then adding or subtracting specific elements in content categories. This provides a way to remove whole groups of elements, such as all work products in a specific domain, or all tasks in a specific discipline.

Configurations will be specified in the following four-step procedure:

1. **Select the plug-ins to be considered for the configuration definition.**

   All additional selections in the following Steps 2 to 4 ***must be included*** in these plug-ins. If categories selected in Steps 3 and 4 are comprised of elements that are defined both inside the selected plug-ins and of elements that are defined in other plug-ins, then configurations will only consider the elements that are within the selected plug-ins.

2. **Select method content packages to be included in the configuration definition.**

   As a refinement to the method plug-in selection, the specific method packages determine which packages should be included into the interpretation of the configuration. For a selected package, every element directly residing inside that package shall be interpreted as part of the configuration.

[21] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

3. **Select content categories to be added to the configuration definition.**

   As an additional refinement to the category definition created with Steps 1 and 2, you can select custom or standard categories whose elements shall be interpreted as part of the configuration. Step 2 required that all elements that are physically stored within the same package are part of the configuration, this steps allows adding individual elements of a category to a configuration.

4. **Select content categories to be subtracted from the configuration definition.**

   As an additional refinement to the category definition created with Steps 1 to 3, you can select custom or standard categories whose elements will not be part of the configuration. In other words, you can subtract whole sets of individual elements from a configuration by assigning them a given category and listing the category in this step.

Advantages of this approach include:

- Increased flexibility in selecting complete packages.
- Ability to remove individual elements from a configuration.
- Ability to remove whole categories from a configuration in a single operation.

Related topics

Method Library[22]
Configuration View[23]
Create a Method Configuration[24]

5.10. Process Management

5.10.1. Process Description

A process describes how a particular piece of work should be done and defines the sequences of tasks performed by roles and the work products produced over time.

In method authoring, the Process Engineer defines roles, tasks, work products and guidance, in addition to the relationships between these elements. In process authoring, the Process Engineer defines the work to be done, the results to be produced, responsibilities for the roles and additional lifecycle elements, such as iterations, phases, activities and milestones. He incorporates the corresponding method elements, combining them into process structures and sequences for carrying out work.

5.10.2. Process Views

To assign method content to a process, there is a choice of working in different **process views** (there are three: work breakdown structure, team allocation view and work

[22] A method library is a container for method plug-ins and method configuration definitions. All method elements are stored in a method library.

[23] Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

[24] Method libraries can be comprised of content from many types of methods and whole families of different processes. A method configuration defines a logical subset of a method library. You use method configurations to define the scope of your authoring work and when publishing or exporting content.

product usage). Each view supports a different approach for creating a process. If requested, the editor updates the other process views semi-automatically using wizards that prompt for decisions on selection of method content elements.

The primary process authoring view can be any of the following:

- **Work Breakdown Structure View**: The process is created by defining a work structure, a hierarchical breakdown of work. From there, the iterations and activities are created first and the activities are subsequently populated by applying tasks from the method content.
- **Team Allocation View**: The process is created by defining which teams and roles will participate in activities and finding the relevant work products and tasks. In a process that has been already created, the roles can be reviewed and tasks or work products can be added to the existing process.
- **Work Product Usage View**: The process is created by defining which work products should be created in activities and finding responsible roles and tasks and roles from there. In a process that has been already created, the work products can be reviewed and tasks or roles can be added to the process.

5.10.3. Capability Patterns and Delivery Processes

The work may have a relatively small scope, in which case it can be described as a capability pattern, or may address a full project lifecycle, in which case it can be described as a delivery process. Capability patterns are used as building blocks to compose delivery processes, they describe reusable clusters of activities in common process areas. A delivery process describes a complete and integrated approach for performing a specific type of project.

A capability pattern does not relate to any specific phase or iteration of a development lifecycle, and should not imply any. In other words, a pattern should be designed in a way that it is applicable anywhere in a delivery process, thereby enabling its activities to be flexibly assigned to whatever phases there are in the delivery process to which it is being applied.

5.10.4. Process and Default Configuration

A method configuration must be specified for the process at its creation, neither capability patterns nor delivery processes can be created without being anchored to a method configuration.

A method configuration defines a logical subset of a method library and contains the list of method plug-ins used to generate a specific instance of process guidance. The processes will be created or "authored" against the specified method configuration.

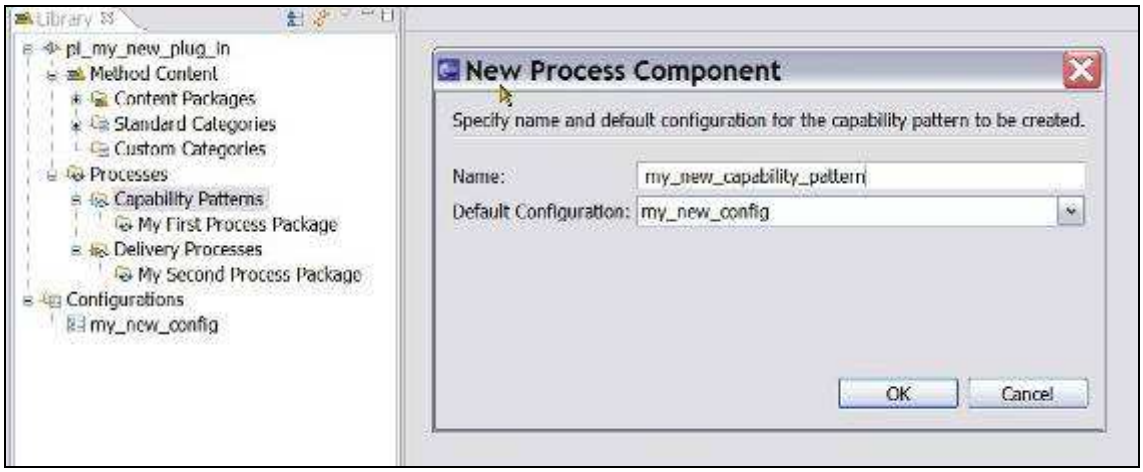
Your process can contain content from many different method plug-ins, not only from the method plug-in, which contains it. When you author your the process, you therefore need to create and/or assign a configuration that defines the set of elements and relationships that must be made visible to the process. This **process-authoring configuration** is referred to as the **default configuration for the process** and it should define the set of method plug-ins, content packages, and other processes from the method library that will be referred to by the process.

The method configuration editor is used to select which method plug-ins, content packages, and categories of elements will be included or excluded from the method

library into the method configuration. The selections restrict the scope of the content used as a basis for defining your process. These selections also determine the content of the published Web site.

The processes can be included in a configuration to be published as part of the published Web site, and can be exported to Microsoft® Project.

A default configuration needs to be selected for the creation of a capability pattern or a delivery process:



More than one configuration can be added, using the capability pattern or delivery process editors. One configuration must be selected as the default; it must be a super-set of all the other configurations already added (*But then, why add the others?*).

When activating a process' editor, the default behaviour is to prompt for the process' default method configuration if a different configuration is selected or if none is selected.

This default behaviour can be changed through **Windows** → **Preferences** → **Method** → **Authoring** → **Process Editor** and selecting the option of **Always** or **Never** switching to the process' default configuration.

As already indicated, method configurations are used also to specify working sets of content and processes defining which elements are published or not published on the Web site.

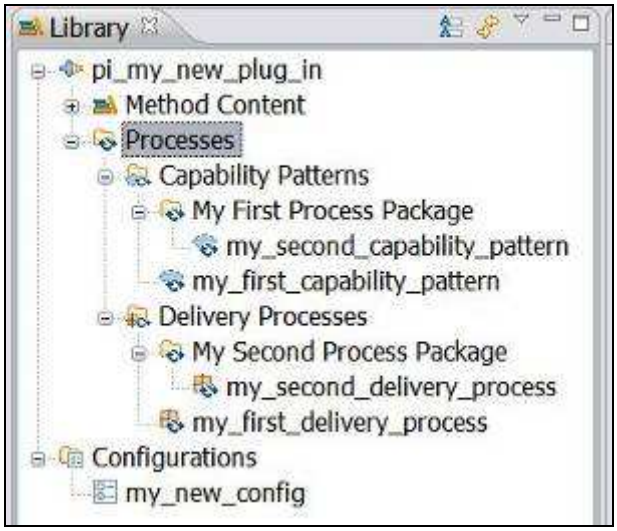
5.10.5. Process Packages

Right-click **Capability Patterns** and select **New** and a choice comes up between creating a new **Process Package** or a new **Capability Pattern**. Right-click **Delivery Processes**, the choice comes up between creating a new **Process Package** or a **Delivery Process**.

Process Packages can be created both under Capability Patterns and under Delivery Processes. New Capability Patterns can be created both directly under Capability Patterns and under any Process Package. New Delivery Processes can be created both directly under Delivery Processes and under any Process Package.

Process Packages is used in the EPF Composer as a grouping concept and it is not clear if they have a role beyond providing visual groupings.

Capability Patterns, Delivery Processes and Process Packages:



5.10.6. Process Diagrams

EPF Composer provides three types of process diagrams:

- **Activity Diagrams** show subordinate activities as part of a higher-level activity. They also show the sequence relationships between those activities.
- **Activity Detail Diagrams** show tasks in an activity with their performing roles along with input and output work products. Activity detail diagrams are similar to workflow detail diagrams.
- **Work Product Dependency Diagrams** illustrate work product dependencies on other work products.

All three types of diagrams are generated and synchronised with the associated work breakdown structure. Changes to the process structure using the diagram editor will be automatically reflected in the work breakdown structure.

5.10.7. Descriptors

Processes do not directly include core method elements but create local references termed descriptors that refer back to the elements in the method library. Descriptors contain additional information relevant to the specific context of the process where they are inserted and a link to the core method element on which they is based. Roles, tasks and work products are all included in processes as descriptors and they can be customised to fit within the context of the process in which they are used. Each element can be represented by multiple descriptors in a process. Descriptors enable reuse of method elements in multiple different contexts.

For example, each time a task is included in a process, a reference object to that task is created in the context of the process. This is called a task descriptor. The same task can be referenced any number of times in the same process. In other words, one task can have many task descriptors in a process. A task descriptor can contain additional information that modifies the base task without actually changing it. For example, roles and work products can be added or suppressed, and steps can be suppressed or re-sequenced.

5.10.8. Process Content Summary

Table 3 - Summary of Process Content

Term	Description
Activity	▪ Activities are the main building blocks for processes. An activity is a collection of work breakdown elements such as task descriptors, role descriptors, work product descriptors, and milestones. Activities can include other activities. The three activity types are Iterations, Phases and Activities.
Task Descriptor	▪ A task is an assignable unit of work. Every task is assigned to a specific role. The duration of a task is generally a few hours to a few days. Tasks usually generate one or more work products.
Role Descriptor	▪ A role is a well-defined set of related skills, competencies, and responsibilities. Roles can be filled by one person or multiple people. One person may fill several roles. Roles perform tasks.
Work Product Descriptor	▪ Work product is a general term for task inputs and outputs, descriptions of content elements that are used to define anything used, produced, or modified by a task. There are three types of work product: Artefact, Outcome and Deliverable
Capability Pattern	▪ Capability patterns are a special type of process that describes a reusable cluster of activities in common process areas, they express and communicate process knowledge for a key area of interest. ▪ Capability patterns are used as building blocks to assemble delivery processes or larger capability patterns, ensuring optimal reuse and application of the key practices they express.
Delivery Process	▪ A delivery process describes a complete and integrated approach for performing a specific type of project
Process Package	▪ Process Packages provide visual groupings of processes. ▪ Process Packages can be created both under Capability Patterns and under Delivery Processes. ▪ New Capability Patterns can be created both directly under Capability Patterns and under any Process Package. New Delivery Processes can be created both directly under Delivery Processes and under any Process Package.
Guidance	▪ Guidance elements are supplementary free-form documentation such as white papers, concept descriptions, guidelines, templates, examples, and so on. Guidance can be added to most method and process elements.

Related topics

Process Authoring Overview
Capability Pattern Reuse
Create Capability Patterns
Create Delivery Processes[25]

[25] A delivery process describes a complete and integrated approach for performing a specific type of project. A delivery process describes what is produced, how it is produced and the required staffing for the entire project lifecycle.

6. Getting Started with Method Authoring

Contents

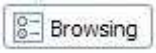
6.1. User Interface

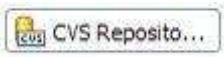
The following are the main features of the user interface:

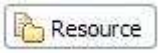
- **Configuration selection box** 

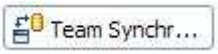
  Use this box to select a configuration[26] in which to work.

- **Open Perspective menu** 
  Use this pull-down menu to select a perspective. The following perspectives can be selected:
  - Authoring Perspective 

  - Browsing Perspective 

  - CVS Repository Exploring 

  - Resource 

  - Team Synchronising 

- **Library view**

[26] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

The Library view displays the Method Plug-ins and Configurations contained in a Method Library.

- **Configuration view**

  The Configuration view[27] shows the content elements in a library filtered by a configuration.

- **Content Editors and Preview**

  When you are working in the Authoring Perspective, the window on the right side of your screen contains content editors that you can use to create or modify element types. When the Browsing Perspective is open, the window on the right contains a preview of your content, as it will appear in a published Web site.

Related topics

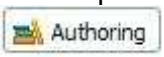
Authoring Perspective
Browsing Perspective
Library View
Configuration View
View Method Content
Search for Content

6.2. Authoring Perspective

The **Authoring** perspective provides views and functions to navigate and author method content and processes. You must be in the Authoring perspective to create or modify any element types.

The Authoring perspective provides two views in separate panels: the **Library View** and the **Configuration View**. Double-click any element in the Library View or Configuration View to open the editor panel on the right. The editor panel contains several tabs through which you can edit information about the element you have selected.

Selecting the Authoring Perspective

Use the Open Perspective  menu in the main tool bar to select Authoring 

Related topics

Configuration View[28]
User Interface
Browsing Perspective

[27] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

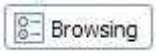
[28] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

6.3. Browsing Perspective

You can use the **Browsing** perspective to preview and navigate through a method configuration without making any changes.

The browsing perspective contains the Configuration View[Error! Bookmark not defined.], which shows the content in the currently selected configuration. Click any element in the Configuration panel to preview the element in the content view, as it will appear in a published Web site. The content view provides browser-like navigation features. Click any link in the displayed page to go to that page. Use the buttons in the content view toolbar to perform familiar browser actions, such as back or refresh.

Selecting the Browsing Perspective

Click Open Perspective  and click Browsing 

.

Related topics

Authoring Perspective
Configuration View[29]
User Interface

6.4. Library View

The **Library** view displays the method content that is available in the current method library, which is organised into sets of method plug-ins and configurations.

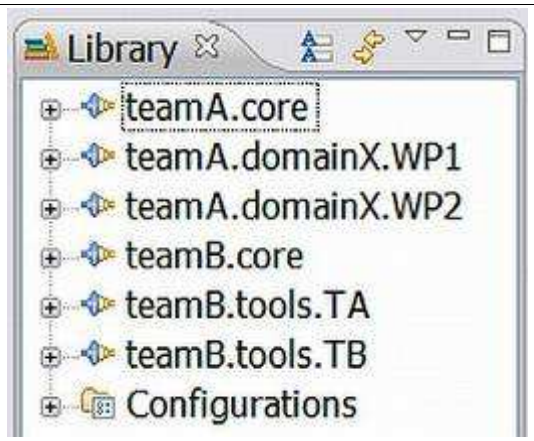
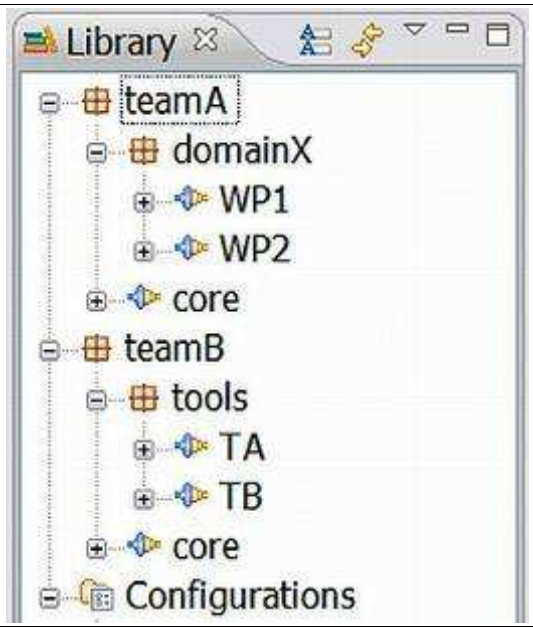
The Library view is available in the **Authoring** perspective and it is not accessible in the **Browsing** perspective. It is organised in visual packages that are logically created and sorted based on the plug-in punctuation within the names. Using dots in the name of the plug-ins creates logical packages that are used for presentation in the Flat and Hierarchical view.

For example, if you create and name the following plug-ins:

- teamA.domainX.WP1
- teamA.domainX.WP2
- teamA.core
- teamB.tools.TA
- teamB.tools.TB
- teamB.core

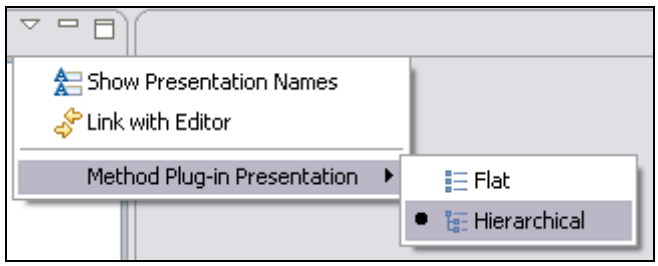
[29] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

The Library view will display the plug-ins in a flat and hierarchical presentation as shown here:

Flat Representation	Hierarchical Presentation
	

When using the **Flat** presentation, the highest-level library content unit is the **Method Plug-in** . The flat presentation displays all the plug-ins in alphabetical order. In the **Hierarchical** presentation of the **Library view**, the highest-level library content unit is called a **logical package**, with the icon . The logical packages are groupings of method plug-ins. The logical packages are created simply by using dots in the method plug-in names. The six method plug-ins are listed in the Hierarchical presentation according to the naming structure created with the dots in the name. If you have many plug-ins in your method library, you can switch the Method Plug-in presentation to hierarchical.

To switch between Flat and Hierarchical presentations, click the **triangle** (down arrow) dropdown menu ▾ in the Library tool bar and select **Method Plug-in Presentation**, then choose **Flat** or **Hierarchical**.



You can choose to display the "physical names" (the creation of a plug-in creates a file with the same name) or to show presentation names by switching the name display.

There are two ways of switching the way the name is displayed:

- Click the button in the Library view.

- Click the Library view drop-down menu ▾ (triangle or down arrow) and select Show Presentation Names.

Related topics

Method Library[30]
Authoring Perspective
Browsing Perspective
Method Configurations[31]

6.5. Configuration View

The **Configuration** View displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

After you select a method configuration from the Configuration box in the toolbar, the Configuration view displays the content from the selected configuration. The configuration view does not show any physical folders, these are only shown in the Library view.

Related topics

Method Library[32]
Authoring Perspective
Browsing Perspective
Method Configurations[33]

6.6. View Method Content

You can browse method content in both the Authoring Perspective and Browsing Perspective. While **authoring** method content you can see a preview of the published page for the content hat you are editing by clicking the **Preview** tab at the bottom of the Content Editor. Click any link in a displayed page to go to that page. Use the buttons in the editor toolbar to perform familiar browser actions, such as Back or Refresh. While **browsing** method content in the Browsing perspective, you can see a preview of any method and process content in a method configuration as it appears in a published Web site.

To browse and preview method content in the Authoring Perspective:

1. Click **Open Perspective** and select **Authoring**. The Authoring perspective opens with the Library View[34] and Configuration View[35] on the left and the

[30] A method library is a container for method plug-ins and method configuration definitions. All method elements are stored in a method library.

[31] A method configuration is a selection of method plug-ins and method packages in a method library.

[32] A method library is a container for method plug-ins and method configuration definitions. All method elements are stored in a method library.

[33] A method configuration is a selection of method plug-ins and method packages in a method library.

Content view on the right. When a method element is selected, the appropriate editor opens in the right panel.

2. To explore the contents in both the **Library view** and **Configuration view**, expand the different content packages and method plug-ins down to their method elements. Double-click an element to open it in the editor.
3. Click the **Preview** tab at the bottom of the content editor to preview the content of any selected element.

   Note: The Preview page in the content **editor does not resolve content variability relationships**. If you are extending, contributing to, or replacing a base content item, you can only see the result of this in the **Configuration view** in the **Browsing** perspective. For more information about content variability, see Method Content Variability[36].

To browse and preview method content in the Browsing Perspective:

1. Click **Open Perspective** and select **Browsing**. The Browsing perspective opens with the **Configuration view** on the left and the **Content view** on the right.
2. Select a **Method configuration** from the configuration pull-down list in the toolbar. The content of the method configuration is displayed in the **Content view** as it is displayed in a published Web site.
3. Explore and preview the content in the Configuration view by expanding the **Categories** and other folders down to method elements. Click a method element to open it in the content view.

Related topics

Authoring Perspective
Browsing Perspective
Configuration View
Method Content Variability
User Interface

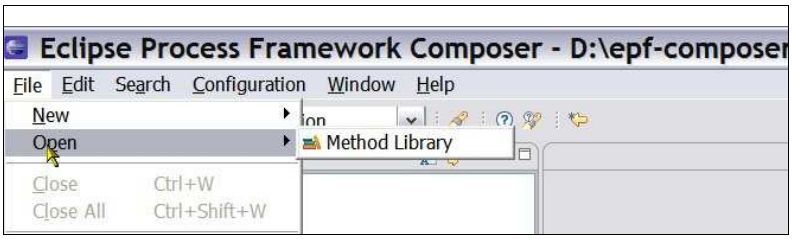
[34] The Library view displays the method content that is available in the current method library, which is organised into sets of method plug-ins and configurations.

[35] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

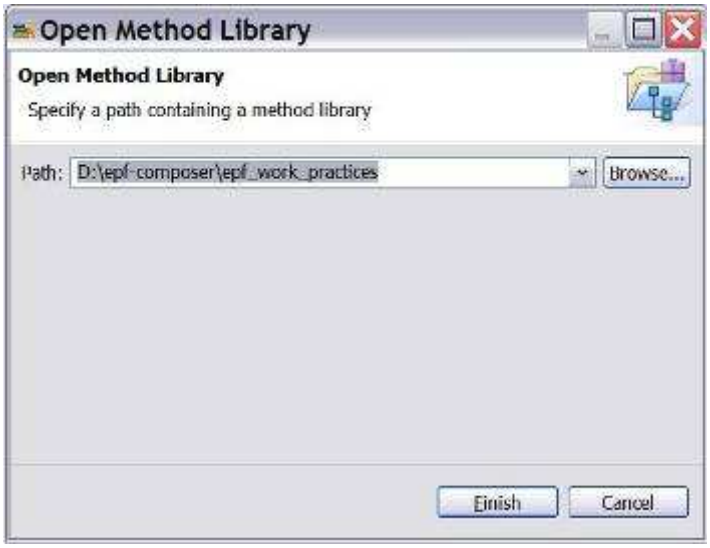
[36] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

6.7. Open an Existing Method Library

An existing method library is opened by selecting **File** → **Open** → **Method Library**:



Then, select the folder with the existing method library content, for example the "*epf_work_practices*" folder or any other folder with existing method library content.



The "**Copy Library**" dialogue window then pops up with the message "The library you are opening is a default library supplied with the composer."

Figure 26. Copy Library

What this message attempts to say (not very clearly however!), is that you have the choice of either using the files in the "*epf_work_practices*" folder or creating a separate folder (using the operating system's file utility) and letting the epf-composer copy the existing files over to the new folder. In the latter case, you will have two sets of files, one working set that you can modify and another with the original pristine and unchanged files. When you want to work with an existing set of library files, click on **Skip**.

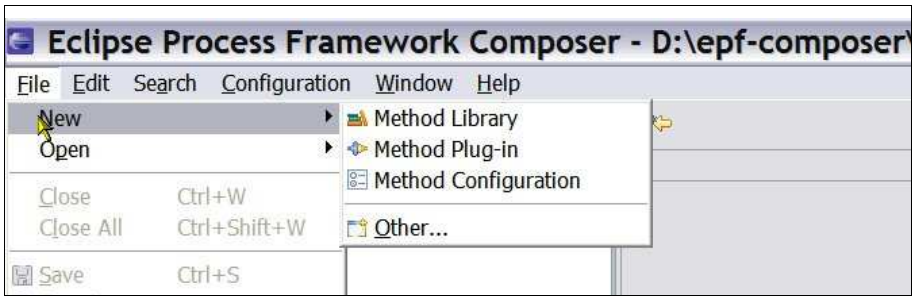
> *There should be an option of turning this nagging feature/bug off. It would seem that simply copying a folder or files, using the operating system, offers the same functionality. Since most users will be more familiar with windows than with EPF Composer, the need for this dialog box is not very clear.*

6.8. Create a New Method Library

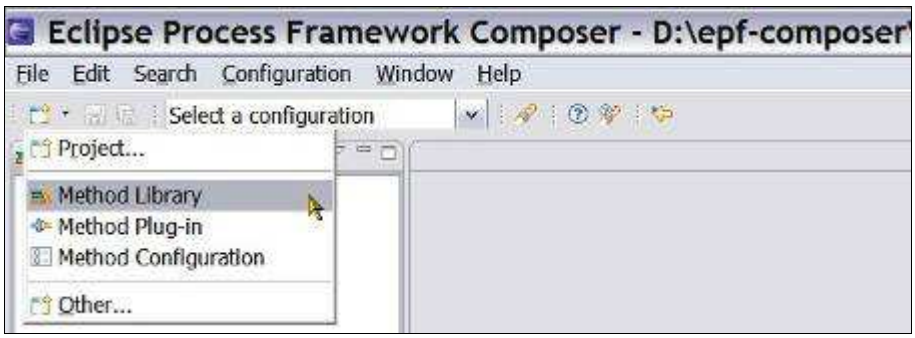
All method content is contained in the method library.

To create a new method library:

1. To create a new method library, make sure you are in the Authoring Perspective.
2. You can begin the method library creation process in two ways:
   a. Click **File** → **New** → **Method Library**.

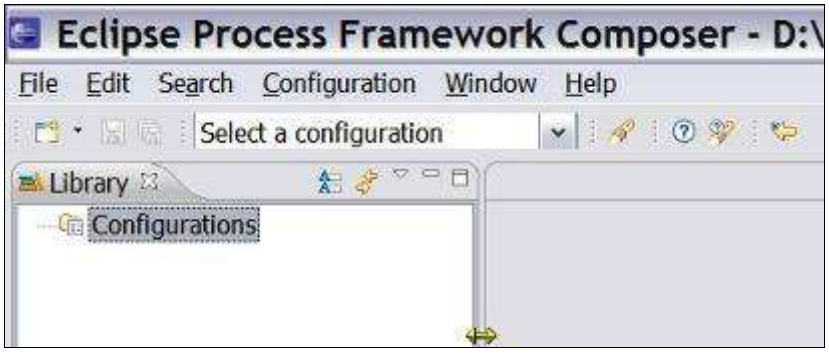


   b. You can also click the Down Arrow of the '**New**' icon in the toolbar and select **Method Library** from the drop down list.



3. The **New Method Library** dialogue box pops up and asks you for the folder where you want to create the new method library. Notice that the library by itself does not have a name. There is a description field you may fill out. Create a new folder or use an already existing one. Select the folder and click on **Finish**.

When a new method library is created, it has initially no method plug-ins and an empty method **Configurations**:



Since all method content is contained in method plug-ins, the next step after the creating of a new method library is either importing an existing method plug-in or creating a new method plug-in.

Related topics

Authoring Perspective
Method Plug-in

6.9. Create a Method Plug-in

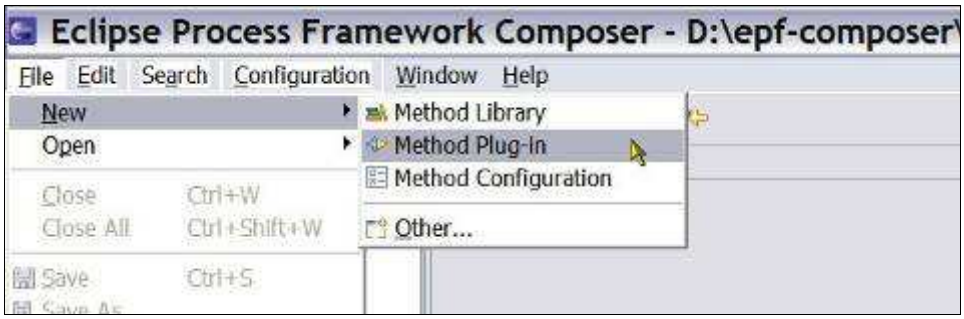
Steps to Create Plug-in

All method content is organised in method plug-ins, contained in a method library.

To create a new method plug-in:

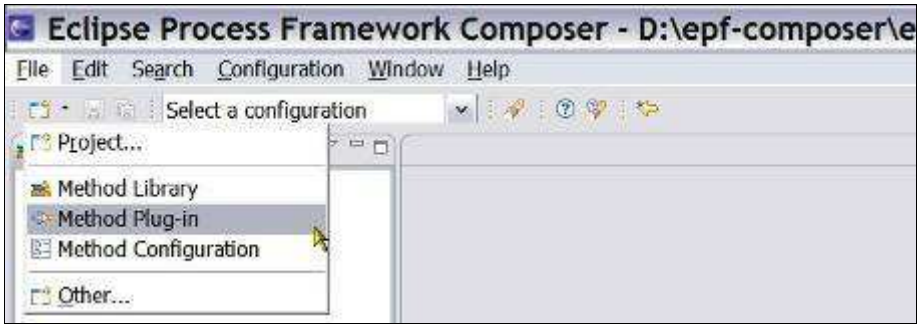
4. Make sure that you are in the Authoring Perspective.

5. You can begin the method plug-in creation process in one of three ways:
   a. Click **File** → **New** → **Method Plug-in**.

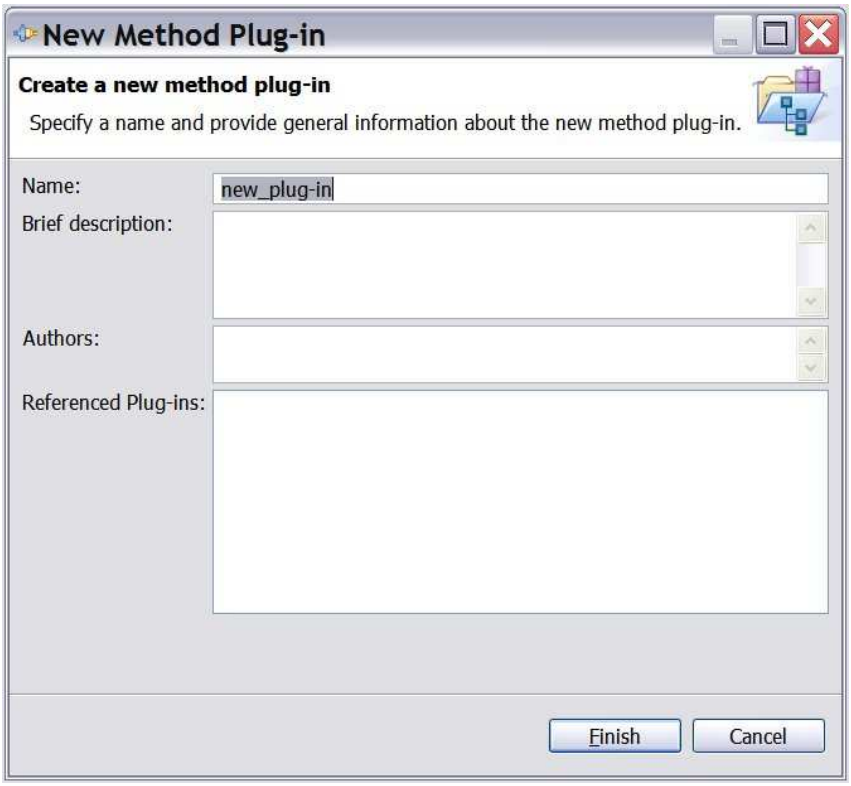


   b. In the Library view, either right-click **Configurations** and select **New Method Plug-in** or right-click an existing method plug-in and select **New Method Plug-in** from the pop-up menu.

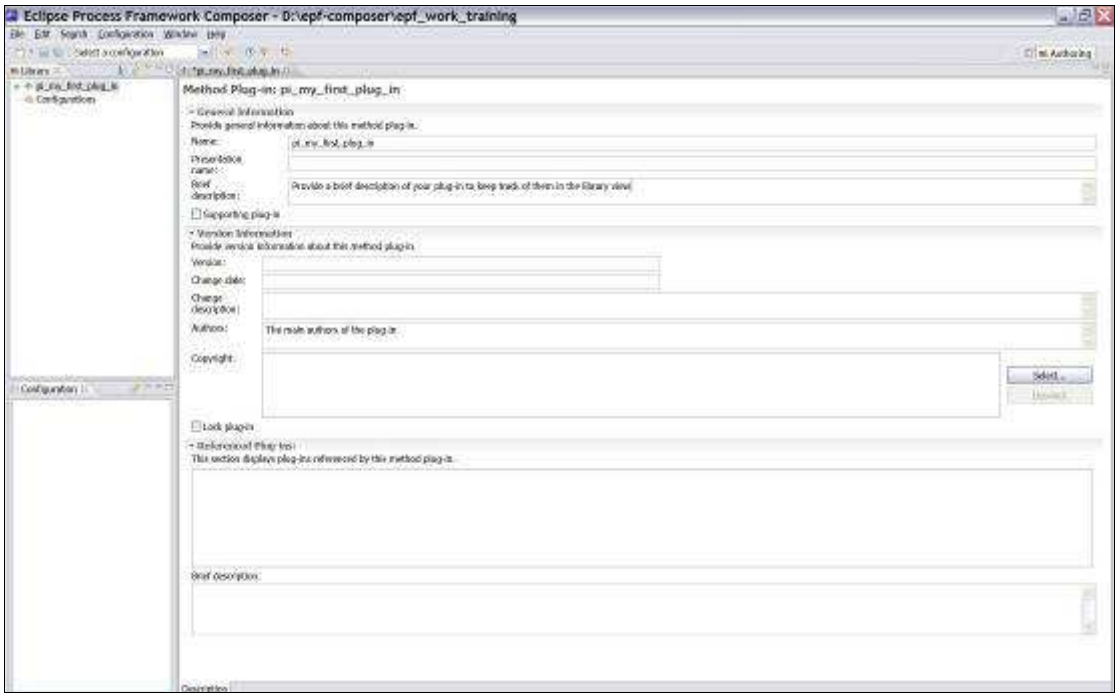
   c. Click the Down Arrow of the '**New'** icon  in the toolbar and select **Method Plug-in** from the drop-down list.



6. The **New Method Plug-in** wizard opens.

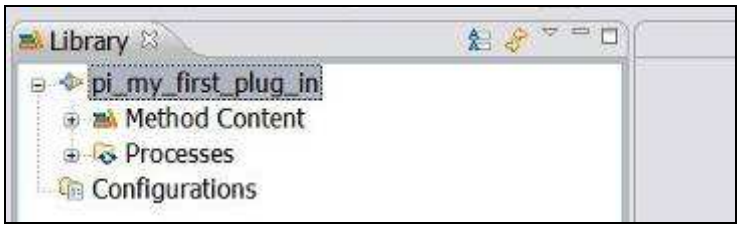


7. In the **Name** field, type a name for your new method plug-in.
8. Provide a **Brief description** of the method plug-in and a list of its **Authors** to help you and others keep track of the method plug-ins in the library. You can return to edit this information at any time.
9. In the **Referenced Plug-ins** field, select one or more method plug-ins. As a result, your plug-in will be an extension to the referenced plug-ins. This allows you to reuse content from the selected method plug-ins, extending them with your own content. If this is the first method plug-in in the library, the field will be disabled.
10. Click **Finish**. Your new method plug-in is in the **Library View** with the other method plug-ins and the method plug-in editor opens. You can open the method plug-in editor any time by double-clicking the method plug-in in the Library view.

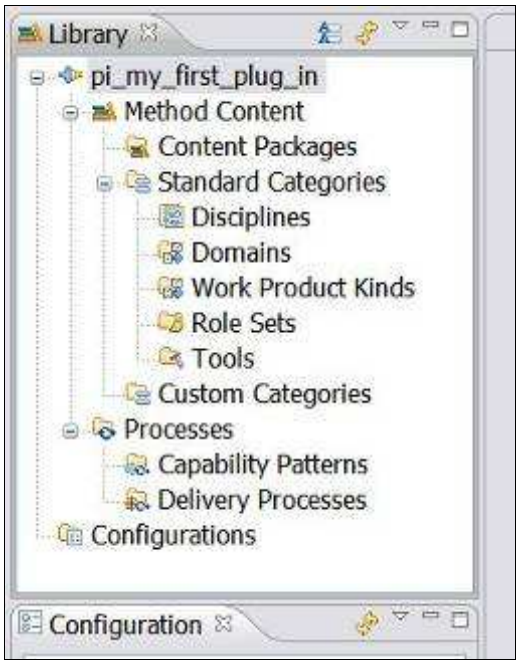


11. The method plug-in editor contains already the information you entered in **New Method Plug-in** wizard.
12. You can then enter additional information, the name of most of the fields are self-explanatory, such as version information, copyright field and an additional brief description field for the referenced plug-ins.
13. There is a **Supporting plug-in** tick box for restricting the plug-in to be a support plug-in. A supporting method plug-in provides reusable content for other method plug-ins. The content that is stored in a supporting method plug-in is only visible and published for a method configuration if other method content references it (and if these are not supporting plug-ins themselves).
14. The method plug-in can be locked to prevent further changes to the content by selecting the **Lock plug-in** box.
15. When you create a new method plug-in or method element or modify an existing element, a ***** symbol is displayed left to the name of the plug-in or element in the tab, indicating that it needs to be saved. There are four ways to save an it:
    - Close the editor and confirm you want to save
    - Click the disk icon in the toolbar
    - Use the shortcut "***ctrl+s***"
    - Click **File** → **Save**
16. A folder named "*pi_my_first_plug_in*" is created in the method library folder. If you rename the name of the method plug-in later, the folder name will change correspondingly.

The **Library view** now contains:



Once expanded, the method plug-in shows the standard structure of all method plug-ins:



Each of these concepts will be explained in the following chapters.

Related topics

Authoring Perspective
Method Plug-in[37]
Method Content Packages[38]
Create a Method Content Package[39]

Decomposition

Some considerations about how to decompose your subject area into method plug-ins may be helpful. Decomposition splits the subject area into smaller areas. Each sub-area is addressed independently and the models of the sub-areas are subsequently aggregated

[37] All content is organised in method plug-ins. With method plug-ins and method packages, you can organise your content at a level of granularity that meets your needs for authoring and reusing content.

[38] A method content package is a container for method elements. Elements are organised in method packages to structure a large scale of method content and processes and to define a mechanism for reuse.

[39] Method content is organised into content packages that are contained in method plug-ins.

to form a complete solution. The decomposition / composition imply a reference model[40] for the larger subject area.

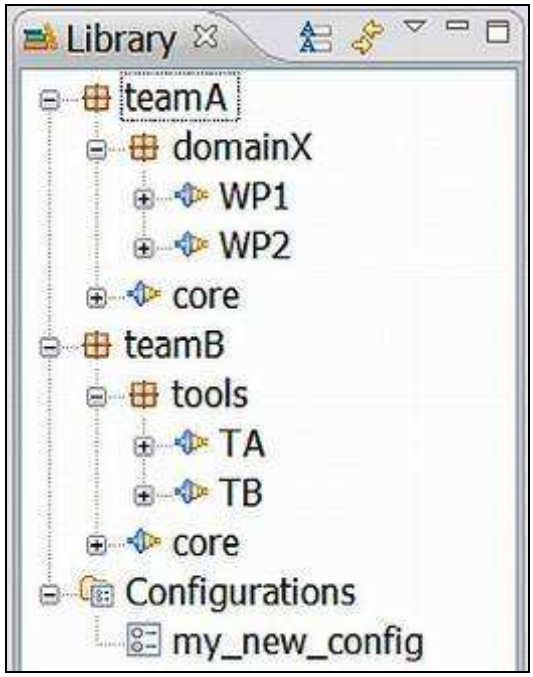
In addition to splitting the subject area itself, it can be useful to create models of each sub-area at multiple levels of abstraction, creating a hierarchical reference model of the subject area. Such an approach is very useful when the subject area is very complex.

The method plug-ins that you create will mimic the reference model and the decomposition you have chosen. The hierarchical decomposition must be reflected in the naming of the method plug-ins, the dots in the name of the plug-ins reproduce the logical groupings of the hierarchical reference model.

The hierarchical view, available in the Library view, will display the hierarchical decomposition. The **Library view** shows all method content in the current library. If for example the names the following plug-ins are as follows:

- teamA.domainX.WP1
- teamA.domainX.WP2
- teamA.core
- teamB.tools.TA
- teamB.tools.TB
- teamB.core

In this case the Library view will display the plug-ins in the hierarchical presentation as shown here:

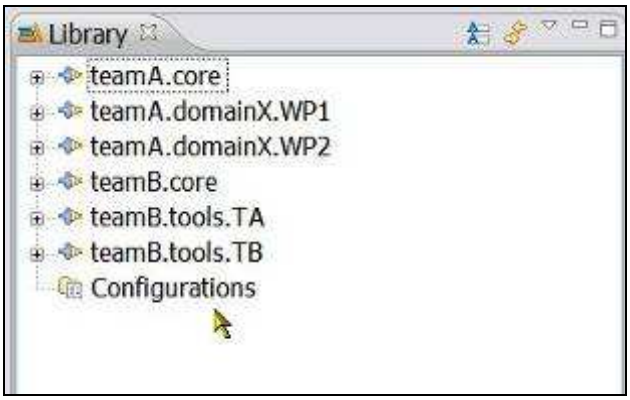


In the **Hierarchical** presentation in the **Library view**, highest-level library content unit is called a **logical package**, with the icon . The logical packages are groupings of method plug-ins and represent different levels in the hierarchical reference model of the subject area. A logical package can therefore contain other logical packages. Every

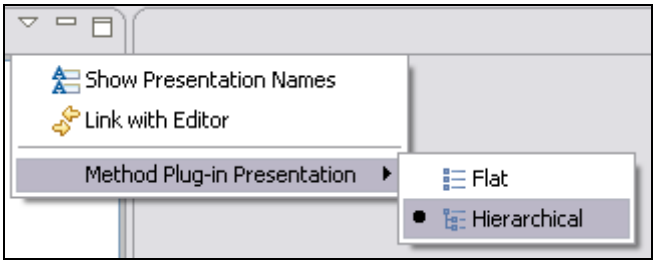
[40] A reference model is an abstract representation of the entities and relationships involved in a complex domain and form the conceptual basis for the development of more concrete models and ultimately implementations.

hierarchical level, down to but not including the method plug-ins, is handled as logical packages. The logical packages are created simply by using dots in the method plug-in names. The six **Method Plug-ins** in the example are listed in the Hierarchical presentation according to the naming structure created with the dots in the name.

When using the **Flat** presentation, the highest-level library content unit is the **Method Plug-in** and the same six method plug-ins will be displayed as:



To switch between Flat and Hierarchical presentations, click the down arrow ▾ in the Library tool bar and select **Method Plug-in Presentation**, then choose **Flat** or **Hierarchical**.



With method plug-ins you can organise your content at a level of granularity and according to your reference model that meet your needs for authoring and reusing content.

6.10. Create a Method Content Package

A method **Content Package** is a container for method elements, such as Roles, Tasks, Work Products and Guidance. All **Method Content Packages** are contained in **Method Plug-ins**.

When you reuse content created and managed by others, you should always create a new Method Content Package[41] and Method Content Elements[42] in a Method Plug-in[43] that you yourself produce and manage. This separates your content from already

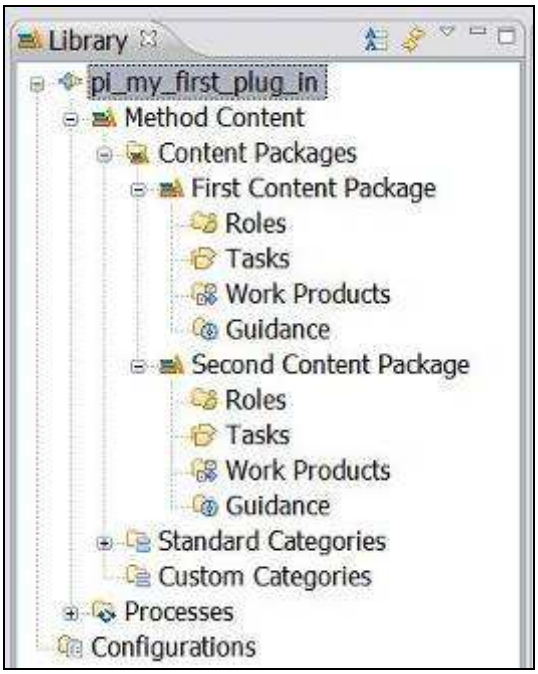
[41] A method content package is a container for method elements. Elements are organised in method packages to structure a large scale of method content and processes and to define a mechanism for reuse.

[42] Method content elements provides step-by-step explanations, describing how specific development goals are achieved, independent of the placement of these steps within a development lifecycle. Processes take these method elements and relate them into semi-ordered sequences that are customised to specific types of projects.

[43] All content is organised in method plug-ins. With method plug-ins and method packages, you can organise your content at a level of granularity that meets your needs for authoring and reusing content.

existing content created and controlled by others and allows you to update the library with their new library releases without affecting the content that you have created in your own plug-ins.

All method Content Packages are stored under the "**Content Packages**" heading.



Each method **Content Package** has the following schema (structure of virtual folders, sectors or nodes):

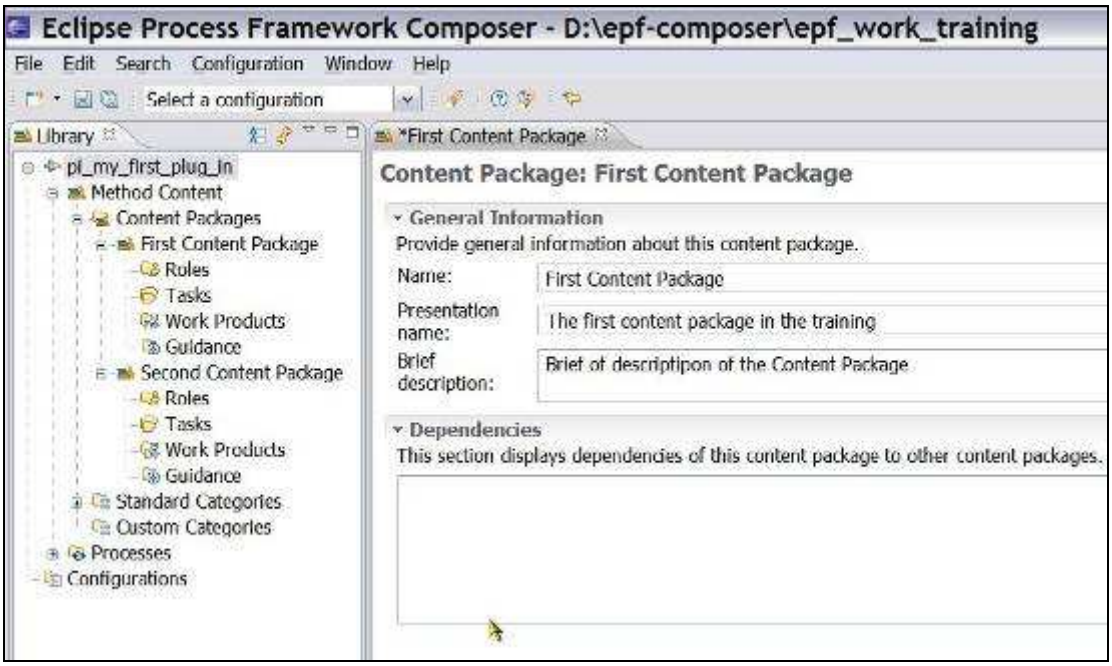
- Roles
- Tasks
- Work Products
- Guidance

Remember: You cannot create a new content package or any other element in a **locked** method plug-in.

To create a method content package

1. Find your method plug-in in the Library View[44].
2. Navigate into the plug-in to find **Content Packages**. This folder contains all content packages with method elements.
3. Right-click the content package and click **New → Content Package**.
4. The Content Package editor pops up:

[44] The Library view displays the method content that is available in the current method library, which is organised into sets of method plug-ins and configurations.



5. Under General Information, give the new content package a unique name and provide a brief description.

   Important: You cannot create a new content package or any other element in a **locked method plug-in**.

6. Save your new content package. There are four ways to save a package or an element:
   - Close the editor and confirm you want to save
   - Click on one of the disk icons (current or all) in the toolbar
   - Use the shortcut "***Ctrl+s***"
   - Click **File → Save**

   Remember: When you create a new element or modify an existing element, a ***** symbol is displayed in the tab next to the name of the element, indicating that the element needs to be saved.

Important: The tool creates nodes for Task, Role, Work Product, and Guidance under the new content package.

Related topics

Method Plug-in
Create a Method Plug-in
Method Content Packages
Method Content Elements
Create Method Content Elements
Guidance Elements[45]

[45] Guidance elements is a general term for supplemental information that can be added to most method elements. Guidance elements can also be associated with other guidance elements.

6.11. Create a Method Configuration

A method configuration defines a logical subset of a method library. You use method configurations to define the scope of your authoring work and when publishing or exporting content.

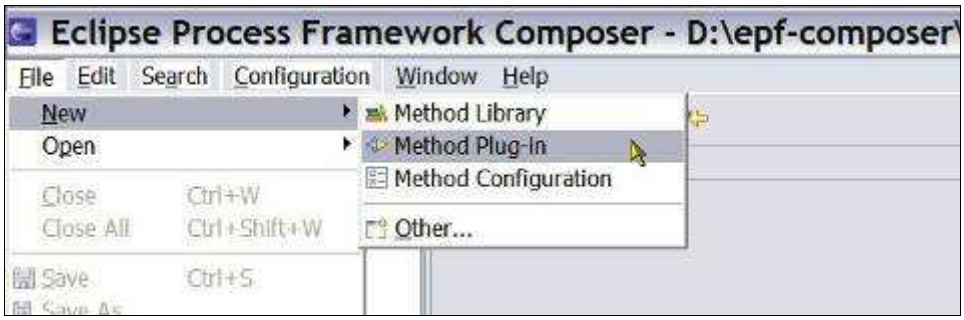
When you create a **process** you must **specify at least one method configuration** against which this process is authored. See Method Configurations Overview for more details. When activating a process editor, the default behaviour is to prompt for the process' default method configuration if a different configuration is selected or if none is selected.

You use the method configuration editor to create and modify method configurations. In this editor, you select which method plug-ins, content packages, and categories of elements you want to include to or exclude from the method library into the method configuration. The selections that you make restrict the scope of the content that you use as a basis for defining your process. These selections also determine the content of the published Web site.

Method configurations are assigned names and are saved so that they can be reused later. In addition to creating new method configurations, as described below, you can also simply copy and paste a configuration by right-clicking the method configuration that you want to copy and afterwards right-clicking the top-level Configurations folder and clicking Paste. You will get a complete copy of the configuration with a new name that you can modify for your needs.

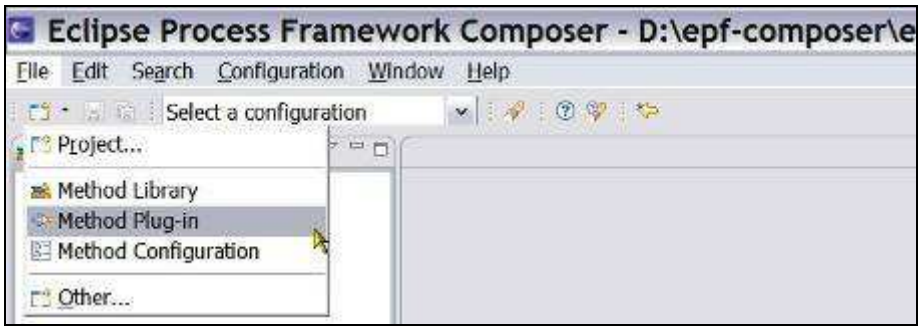
To create a new method configuration:

1. Make sure that you are in the Authoring Perspective.
2. You can begin the method plug-in creation process in one of three ways:
   a. Click **File** → **New** → **Method Configuration**.

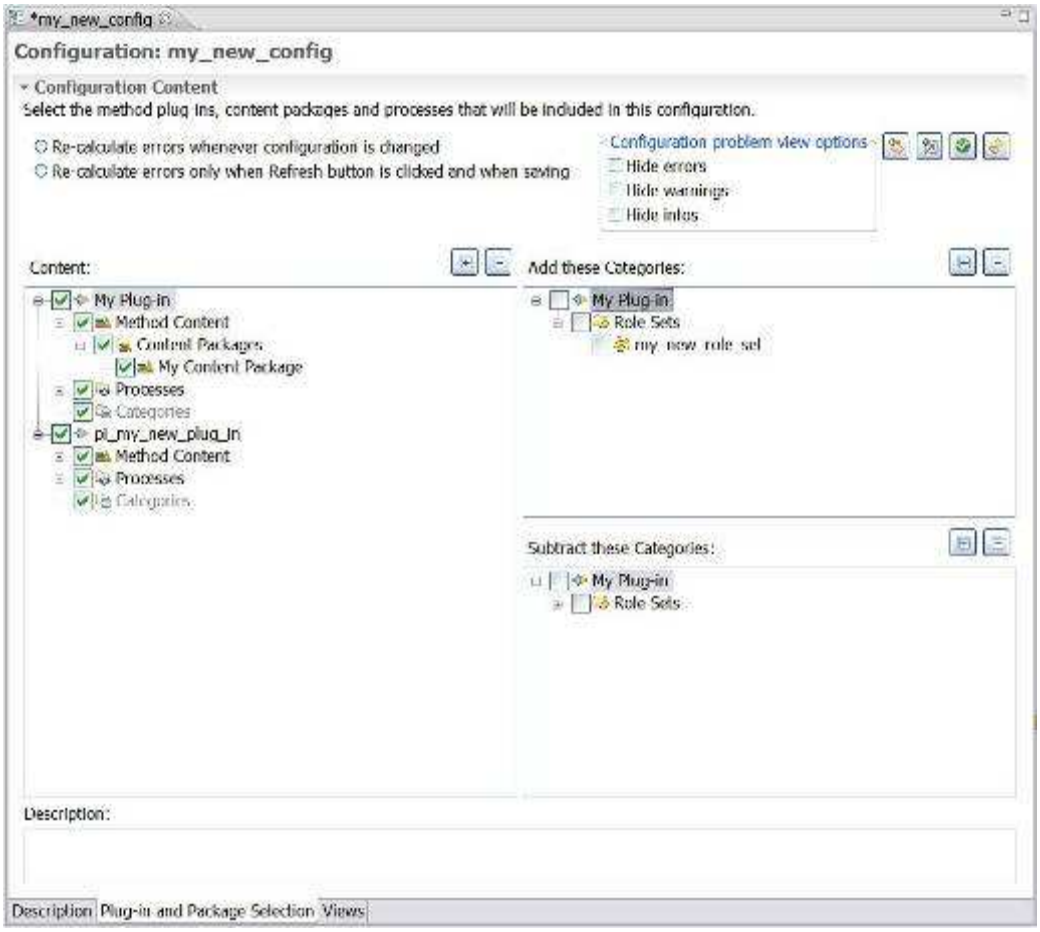


   b. In the **Library** view of the **Authoring** perspective, right-click the **Configurations** folder and click **New** → **Method Configuration**.

   c. Click the Down Arrow of the '**New'** icon in the toolbar and select **Method Configuration** from the drop-down list.



3. In the **Create a new method configuration** dialog, type a name and description for your configuration and click Finish. The Configuration editor is opened and you will see your new configuration name in the Configurations folder in the Library view tree.
4. At the bottom of the editor, click the **Plug-in and Package** Selection tab.



5. The method **Content** selection section on the left shows a list of all Method Plug-ins, their content packages, and processes. Use the check boxes to add or remove plug-ins, packages, and processes to or from your configuration. Expand each plug-in to select and clear individual packages and processes. You can use the Expand All  and Collapse All  buttons to browse the hierarchy.
6. Select **Categories** on the right to be added to or removed from the configuration definition. You can add or remove elements that have been categorised with **standard** or **custom categories**.

On the right are all the method plug-ins that were selected on the left in the method **Content** selection section. As you browse, notice that the sub-trees **list all categories that are defined in the selected method plug-ins**. ***When you select a category, all elements in this category are added to the configuration, regardless of whether the content packages where these elements are stored were selected in the list on the left.*** In this way, you can add elements to your configuration based on "logical" criteria. For example, you might want to add all of the Template guidance that was in a category called "High Ceremony" to your configuration regardless of which content packages these templates were stored in.

When you select a category in the right bottom box, then all elements that were in the category are removed (or subtracted) from the configuration; regardless if the content packages in which these elements are stored were selected in the list on the left or not. This allows you to remove systematically content that was categorised with that category from your configuration.

7. Save the method configuration:
   - Close the editor and confirm you want to save
   - Click on one of the disk icons (current or all) in the toolbar
   - Use the shortcut "***Ctrl+s***"
   - Click **File** → **Save**
8. **Navigation Views**: Define the views for the method configuration. ***A view is a navigation tree browser in a published method configuration***. Every published configuration can have several views that are displayed as tree browsers in individual sliders. ***The structure of the view is defined as a Custom Category***.
   a. In the **Configuration** editor, click the **Views** tab.
   b. Click **Add View**.
   c. If needed, click the plus (+) sign to expand the **Custom Categories** folder, and select the category that you want to use as your view. Click the (+) sign to expand your chosen category and view its contents. Press and hold the Ctrl key to select multiple views.
   d. Click **OK**. The window closes and the views that you selected are added to the configuration.
9. Select the view that you want to display as your method configuration's start-up view and click **Make Default**. The start-up view is the first view shown when a published configuration opens for the first time.
10. Click **Order** to open a window so that you can change the order of how the views are displayed in the published site.
11. Click **File** → **Save**

Related topics

Method Configurations[46]
Copy a Method Configuration[47]
Method Content Package[48]

[46] A method configuration is a selection of method plug-ins and method packages in a method library.

[47] It is easier to copy an existing configuration rather than to create a new one.

Create a Method Plug-in[49]

6.12. Copy a Method Configuration

It is easier to copy an existing configuration rather than to create a new one.

To copy a method configuration:

1. Make sure that you are working in the Authoring Perspective.
2. In the **Library** view, expand the **Configurations folder**.
3. Right-click the method configuration that you want to copy and click **Copy**.
4. Right-click the **Configurations folder** and click **Paste**. Type a name for the new configuration. The copied method configuration is pasted into the Configurations folder.
5. Double-click the newly pasted method configuration to open its configuration editor.
6. In the configuration editor window, enter a new name for your configuration in the Name field. If you click anywhere in the Library view or Configuration View[50], you will see your new configuration name in the Configurations folder in the Library view tree.
7. Type a description for your configuration in the **Description** field, then click **File** → **Save All** to save your new method configuration.
8. At the bottom of the editor window, click the **Plug-in and Package** Selection tab.
9. The method configuration selection section shows a list of all method plug-ins, their content packages, and processes. Use the check boxes to add or remove plug-ins, packages, and processes to or from your configuration. Expand each plug-in to select and clear individual packages and processes.
10. Click **File** → **Save All**.
11. Define **Views** (Navigation Views) for the method configuration. A view is a navigation tree browser in a published configuration. Every published configuration can have several views that are displayed as stacked tree browser tabs. The structure of the view is defined as a custom category
    a. In the Configuration Editor, click the **Views** tab.
    b. Click **Add View**.
    c. If needed, click the (+) sign to expand the Custom Categories folder and select the category that you want to use as your view. Click the (+) sign to expand your category and view its contents. Press and hold the **Ctrl** key to select multiple views.

---

[48] A method content package is a container for method elements. Elements are organised in method packages to structure a large scale of method content and processes and to define a mechanism for reuse.

[49] All content is organised in method plug-ins. With method plug-ins and method packages, you can organise your content at a level of granularity that meets your needs for authoring and reusing content.

[50] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

d. Click **OK**. The window closes and the views that you selected are added to the configuration.

12. Select the view that you want to display as your configuration's start-up view and click Make Default. The start-up view is the first view shown when a published configuration is displayed for the first time.
13. Click **File** → **Save All**.

Related topics

Method Configurations[51]
Create a Method Configuration[52]
Method Plug-in[53]
Method Content Package[54]
Create a Method Plug-in[55]

6.13. Search for Content

To search for content:

1. Use one of the following three options to search for method content.
   - Click **Search** → **Search**.
   - Click **Search** → **File**.
   - Click the **Search**  icon in the tool bar.

   The **Search Window** opens.
2. Click either the **File Search** tab or the **Method Search** tab, and set your criteria.
   - Using **File Search**: Type a text string to search for within the files (optional), or type a file name pattern to use for matching file names. You must use a pattern. Use ***** if you want to search for the text in all files. Leave all other options as default.
   - Using **Method Search**: In the method library, you can search method content for text strings in the element's documentation, or search elements by name.

To search content using the **Method Search** use one of the following search criteria:

- **Text**: Type a string in the text field to search all documentation fields for the method content and processes.
- **Method element name or presentation name pattern**: Type a name pattern in the field to match both element names and presentation names.

---

[51] A method configuration is a selection of method plug-ins and method packages in a method library.

[52] Method libraries can be comprised of content from many types of methods and whole families of different processes. A method configuration defines a logical subset of a method library. You use method configurations to define the scope of your authoring work and when publishing or exporting content.

[53] All content is organised in method plug-ins. With method plug-ins and method packages, you can organise your content at a level of granularity that meets your needs for authoring and reusing content.

[54] A method content package is a container for method elements. Elements are organised in method packages to structure a large scale of method content and processes and to define a mechanism for reuse.

[55] All content is organised in method plug-ins. With method plug-ins and method packages, you can organise your content at a level of granularity that meets your needs for authoring and reusing content.

- **Scope**: In the Scope section, you can restrict your search to a subset of element types. Define your subset of element types that the search should look for by selecting the boxes that correspond to the types that you want to include in your search.

3. Click **Search**. The Search view opens with your search results.
4. In the **Search** view, you can double-click an element to open its respective editor. If you have Link with Editor selected in the Library view, it displays where the selected element is in the library.

Related topics

Method Content Elements[56]

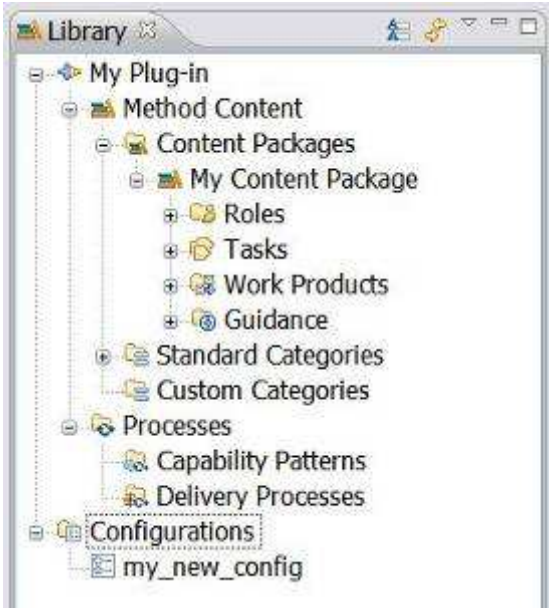
[56] Method content provides step-by-step explanations, describing how specific development goals are achieved, independent of the placement of these steps within a development lifecycle. Processes take these method elements and relate them into semi-ordered sequences that are customised to specific types of projects.

7. Create Method Content

Contents

7.1. Create Method Content Elements

Method Content provides step-by-step explanations, describing how specific development goals are achieved independent of the placement of these steps within a process. Processes take the method elements and relate them into semi-ordered sequences that are customised to specific types of projects.



Method content elements are:

- Role[57]: contains roles.
- Task[58]: contains tasks.

[57] A role defines a set of related skills, competencies, and responsibilities of an individual or individuals.

- Work Product[59]: contains artefacts, deliverables or outcomes.
- Guidance[60]: contains checklist, concept, example, guideline, estimation considerations, practice, report, reusable asset, roadmap, supporting material, template, term definition, tool mentor, and whitepaper.

To create a method content element:

1. In the **Library** view, expand the **Content Package** in which you want to create a new method content element.
2. Right-click the folder containing the **type of method content element** you want to create and select **New**, then select the element that you want to create from the sub-menu. The new element is created and its respective editor opens.

   Note: The element you are creating will be nested under the parent folder of the type of method content you are creating, i.e. a role will be nested under the parent folder Roles, a Task under the parent folder Tasks and so forth.

Related Topics

Method Content Elements [61]
Create a Role
Create a Task
Create a Work Product
Guidance Elements[62]
Create Guidance Elements
Method Content Variability[63]

7.1.1. Create a Role

A role defines a set of related skills, competencies, and responsibilities of an individual or individuals.

To create a role:

1. Expand the **Content Package** in which you want to create the role.
2. Right-click **Roles** and select **New** → **Role**. The role editor opens with the Description tab set as the default.
3. In the **Name** field, type a unique name for your role. The role name is the name that is in the Library view.

---

[58] A task is an assignable unit of work. Every task is assigned to a specific role. The granularity of a task is generally a few hours to a few days and usually affects one or a small number of work products.

[59] A work product is a term that is used to describe task inputs and outputs.

[60] Guidance provides information about how to perform a role, how to create a work product, how to perform your task, and so on.

[61] Method content provides step-by-step explanations, describing how specific development goals are achieved, independent of the placement of these steps within a development lifecycle. Processes take these method elements and relate them into semi-ordered sequences that are customized to specific types of projects.

[62] Guidance is a general term for supplemental information that can be added to most Method and Process elements. Guidance elements can also be associated with other guidance elements.

[63] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

4. In the **Presentation name** field, type a presentation name. The presentation name is the name that is in your published content and in the Configuration View[64]. You can also make this name appear in the Library View[65] by toggling **the Show Presentation Names** button in the **Library** view toolbar.
5. In the **Brief Description** field, type a short description of the new role.
6. In the **Main Description** field, type a more detailed description of the new role. The main description is a more detailed version of the description you typed into the Brief Description field. There are three ways you can create text for the description:
   - Directly, by typing it manually in the editor.
   - Copy from another similar role and then modify by using the editor.
   - Copy from an HTML source such as a published Web site.

   Tip: You can use the **Rich Text Editor** to edit or enter the text for any field that has the Rich Text Editor icon  . Click the icon to access the Rich Text Editor. Click the icon again to close the Rich Text Editor. For more information about the editor, see Rich Text Editor[66]
7. Under **Version Information**, provide any pertinent version information about the role.
8. Click the **Work Products** tab and click **Add** to the right of the **Responsible** for field.
9. Select one or more work products from the list. A description of the work product that you select is in the Brief Description field at the bottom of the window.

   Tip: A work product is displayed if it is the output of a task, which the role performs. The list of work products in the Work products that are **output** of tasks that this role performs is computed and cannot be changed with the role editor.
10. Click **OK**. The window closes and the Responsible for field is populated.

    Remember: Selected elements in an Add/Remove section display both the element name (plug-in name) and the "path" (package name) to that element.
11. Click the **Guidance** tab. Use this part of the editor to add and remove guidance elements for the role. To add guidance, click **Add**, select the guidance you want to add, and click **OK**. To remove guidance, select it in the Guidance field and click **Remove**. When you select a guidance element, the brief description of the guidance is displayed.
12. Click the **Categories** tab. ***A role can be included in one or more role sets and any number of custom categories.*** Open the appropriate Select window by clicking **Add** next to the category lists. You can remove role sets or custom categories by selecting them in the appropriate box and then clicking Remove.

---

[64] The Configuration View displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

[65] The Library view displays the method content that is available in the current method library, which is organized into sets of method plug-ins and configurations.

[66] The rich text editor provides text formatting functions that alter the appearance of published content. The CSS style sheet in a Web site controls the appearance of text at a global level, but the rich text editor can override those controls for specific text elements.

13. Click the **Preview** tab to view the newly created role and then save by closing the editor.

Related topics

Method Plug-in[67]
Configuration View[68]
Rich Text Editor[69]
Create Method Content
Method Content[70]
Create a Task[71]
Create a Work Product[72]
Create Guidance[73]
Variability[74]

7.1.2. Create a Task

A task is an assignable unit of work. Every task is assigned to a specific role. The granularity of a task is generally a few hours to a few days and usually affects one or a small number of work products.

To create a new task:

14. Expand the **Content Package** in which you want to create the task.
15. Right-click Tasks and select **New → Task**. The task editor opens with the Description tab as the default.
16. In the **Name** field, type a unique name for your task. The task name is the file name that appears in the Library View[75].
17. In the **Presentation name** field, type a presentation name. The presentation name is the name that appears in your published content and in the Configuration

[67] All content is organized in method plug-ins. With method plug-ins and method packages, you can organize your content at a level of granularity that meets your needs for authoring and reusing content.

[68] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

[69] The rich text editor provides text formatting functions that alter the appearance of published content. The CSS style sheet in a Web site controls the appearance of text at a global level, but the rich text editor can override those controls for specific text elements.

[70] Method content provides step-by-step explanations, describing how specific development goals are achieved, independent of the placement of these steps within a development lifecycle. Processes take these method elements and relate them into semi-ordered sequences that are customized to specific types of projects.

[71] A task is an assignable unit of work. Every task is assigned to a specific role. The granularity of a task is generally a few hours to a few days and usually affects one or a small number of work products.

[72] A work product is a term that is used to describe task inputs and outputs.

[73] Guidance provides information about how to perform a role, how to create a work product, how to perform your task, and so on.

[74] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

[75] The Library view displays the method content that is available in the current method library, which is organized into sets of method plug-ins and configurations.

View[76]. You can also make this name appear in the Library view by toggling the **Show Presentation Names** button in the **Library** view toolbar.

18. In the **Brief Description** field, type a short description of the new task.
19. Type a more detailed description of the new task in the **Main Description** field. The main description is a more detailed version of the description that you entered in the Brief Description field. There are three ways that you can create text for the description:
    - Directly, by typing it manually in the editor.
    - Copy from another similar role and then modify by using the editor.
    - Copy from an HTML source such as a published Web site.

    Tip: You can use the Rich Text Editor to edit or enter the text for any field that has the Rich Text Editor icon. Click the icon to access the Rich Text Editor. Click the icon again to close the Rich Text Editor. For more information about the editor, see Rich Text Editor[77].
20. Complete the other task-specific fields in Detail Information as needed. For more information about creating **variability**, see Variability[78].
21. Click the Steps tab. The Steps Editor opens. A task can have a series of steps that describe how to perform that task. With the Step Editor you can:
    - **Create a new step**
      - Click Add.
      - Give the step a name in the Name field.
      - Describe the step in the Description field.
    - **Remove a step**
      - Select the step to remove in the Steps field.
      - Click Delete.
    - **Move a step up the list**
      - Select the step that you want to move up.
      - Click Up.
    - **Move a step down the list**
      - Select the step that you want to move down.
      - Click Down.
22. Click the **Roles** tab. This part of the editor allows you to define the roles that perform the task. You should select a role as the **primary performer** for this task. You can also add one or more roles as additional performers. To add a role, click **Add**, select the role that you want to add, and click **OK**. To remove a role,

[76] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

[77] The rich text editor provides text formatting functions that alter the appearance of published content. The CSS style sheet in a Web site controls the appearance of text at a global level, but the rich text editor can override those controls for specific text elements.

[78] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

select it in the Primary performers or Additional performers boxes, and click **Remove**. When you select a role, the brief description of that role is displayed.

23. Click the **Work Products** tab. This part of the editor allows you to define the work products that are **inputs** and **outputs** for this task. You can select any number of work products as mandatory inputs, optional inputs, and outputs. To add a work product, click the appropriate **Add** button, select the work products that you want to add, and click **OK**. To remove a work product, select it in the appropriate box, and click the corresponding **Remove** button. When you select a work product, the brief description of the work product is displayed.
24. Click the **Guidance** tab. This part of the editor allows you to add and remove guidance for the task. To add guidance, click **Add**, select the guidance that you want to add, and click **OK**. To remove guidance, select it in the Guidance box, and click **Remove**. When you select a guidance element, the brief description of the guidance is displayed.
25. Click the **Categories** tab. A task can be included in one **discipline** and any number of custom categories. Open the Select window by clicking the appropriate **Add** button next to the category lists. Select a single discipline or any number of custom categories, and click **OK**. To remove category fro a task, select the category and click the appropriate Remove button.
26. Click the **Preview** tab to view the newly created task as it will appear in a published Web page, and save by closing the tab.

Related topics

Method Plug-in[79]
Configuration View[80]
Rich Text Editor[81]
Method Content Variability[82]
Method Content[83]
Create Method Content
Create a Role[84]
Create a Work Product[85]
Create Guidance Elements[86]

---

[79] All content is organized in method plug-ins. With method plug-ins and method packages, you can organize your content at a level of granularity that meets your needs for authoring and reusing content.

[80] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

[81] The rich text editor provides text formatting functions that alter the appearance of published content. The CSS style sheet in a Web site controls the appearance of text at a global level, but the rich text editor can override those controls for specific text elements.

[82] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

[83] Method content provides step-by-step explanations, describing how specific development goals are achieved, independent of the placement of these steps within a development lifecycle. Processes take these method elements and relate them into semi-ordered sequences that are customized to specific types of projects.

[84] A role defines a set of related skills, competencies, and responsibilities of an individual or individuals.

[85] A work product is a term that is used to describe task inputs and outputs.

[86] Guidance provides information about how to perform a role, how to create a work product, how to perform your task, and so on.

7.1.3. Create a Work Product

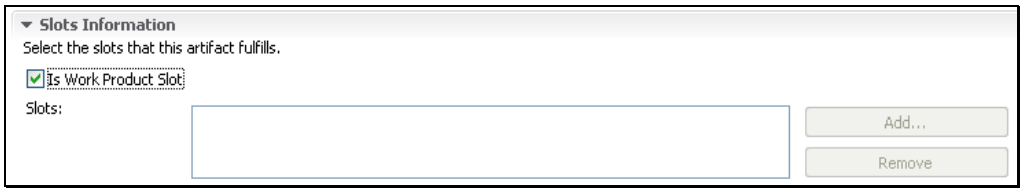
A work product is a term that is used to describe **task inputs and outputs**. There are three types of work products:

- **Artefacts**: An artefact is a tangible work product that is consumed, produced, or modified by one or more tasks. Artefacts may be composed of other artefacts.
- **Outcomes**: An outcome is an intangible work product that may be a result or state. It may also be used to describe work products that are not formally defined.
- **Deliverables**: A deliverable is a collection of work products, usually artefacts, used to define typical or recommended content in the form of work products packaged for delivery.

To create a work product:

1. Expand the **Content Package** in which you want to create the task.
2. Right-click Work Products and select one of the following work product types:
   - **New → Artefact**
   - **New → Outcome**
   - **New → Deliverable**

   The Work Product editor for the work product type that you chose opens and has the Description tab set as the default.
3. In the **Name** field, type a unique name for your work product. The work product name is the name that appears in the Library View[87].
4. In the **Presentation** name field, type a presentation name. The presentation name is the name that is shown in your published content and in the **Configuration** view. You can also make this name appear in the **Library** view by toggling the **Show Presentation Names** button in the Library view toolbar.
5. In the **Brief Description** field, type a short description of the new work product.
6. In the **Slots** Information section, select the **Is Work Product Slot** check box to make the work product a slot[88], or alternatively click the Add button and select one or more work product slots to fulfil your work product.



The feature is relevant for compatibility with the IBM Rational Method Composer. See the Help Files for more information. Leave the box unchecked.

[87] The Library view displays the method content that is available in the current method library, which is organized into sets of method plug-ins and configurations.

[88] A work product slot is an abstract work product that represents a placeholder for concrete work products. Concrete work products can fulfil one or more work product slots. Fulfilment is modelled in Rational Method Composer as a relationship from the concrete work product to the work product slot. Fulfilment is realised by Rational Method Composer when computing a method configuration for browsing or publishing. Then all slots are filled with the concrete work products that are available in that method configuration.

7. In the **Main Description** field, type a more detailed description of the new work product. The main description is a more detailed version of the description that you typed into the Brief Description field. There are three ways to create text for the description:
   - Directly, by typing it manually in the editor.
   - Copying from another similar work product and modifying using the editor.
   - Copying from an HTML source, such as one of the other Web sites.

   You can use the Rich Text Editor to edit or enter the text for any field that has the **Rich Text Editor** icon  . Click the icon to access the Rich Text Editor. ***Click the icon again to close*** the Rich Text Editor. For more information about the editor, see Rich Text Editor.
8. Under Notation, enter notations about the Work Product.

   Tip: Artefacts and Deliverables contain the Notation fields. Outcomes do not.
9. Under **Icon**, you can select a node icon to appear with the work product in the Library and Configuration views and in the tree browser in a published Web site. The node icon must be 16 x 16 pixels. You can also select a **Shape** icon to appear at the top of the published Web page for the work product.
10. Complete the other work product specific fields as needed. For information about adding variability to your work product, see Method Content Variability[89].
11. Click the Guidance tab. When you select a guidance element, the brief description of the guidance is displayed. Use this part of the editor to add and remove guidance for the work product.

    To add guidance:

    a. Click **Add**.

    b. Select the guidance element that you want to add, and click **OK**.

    To remove guidance:

    a. In the **Guidance** field, select a guidance element.

    b. Click **Remove**.
12. Click the **Categories** tab. Open the appropriate Select window by clicking **Select** or **Add** next to the category lists. Select a single domain or any number of work product kinds or custom categories and click **OK**. You can remove a work product from a category by selecting it and then clicking the appropriate **Remove** button.

    Note: A **work product** can be included in one and only one **Domain** (A domain is standard category), any number of **Work Product Kinds** (A Work Product Kind is a standard category), and any number of **custom categories**.
13. If the work product is a **deliverable**, you can go to the **Deliverable** Parts tab and define what the deliverable consists of. This part of the editor only appears for deliverables and it allows you to add **other work products as part of the deliverable**. When you select a work product, its brief description is displayed.

    a. Click the **Deliverable** Parts tab.

[89] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

b. To add work products, click **Add**, select the work products that you want to add, and click **OK**.

c. To remove a work product, select it in the work products as deliverable parts field, and then click **Remove**.

14. If the work product is an **Artefact**, you can define **sub-artefacts** for it. Artefacts can be composed of sub-artefacts expressing that the definition of this artefact type is composed into sub-artefact definitions, which are part of the overall artefact.

    For example, the definition of a Use-Case model is composed into the definition of the Use Case artefact and the Actor artefact.

    To create a sub-artefact, right-click an artefact in the Library view and select **New** → **Artefact**. A new artefact editor is opened and the artefacts are displayed in a hierarchy in the **Library** view.

    Note: When applying a decomposed artefact to a process, the child artefacts are also shown in the process' breakdown structure. You **can remove parts** for the process as well as removing the sub-artefact descriptors in the process editor.

15. Click the **Preview** tab to view the newly created work product. You can save it by closing the editor.

Related topics

Method Plug-in[90]
Configuration View[91]
Rich Text Editor[92]
Method Content Variability[93]
Method Content[94]
Create Method Content
Create a Task[95]
Create a Role[96]
Create a Work Product[97]
Create Guidance[98]

---

[90] All content is organized in method plug-ins. With method plug-ins and method packages, you can organize your content at a level of granularity that meets your needs for authoring and reusing content.

[91] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

[92] The rich text editor provides text formatting functions that alter the appearance of published content. The CSS style sheet in a Web site controls the appearance of text at a global level, but the rich text editor can override those controls for specific text elements.

[93] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

[94] Method content provides step-by-step explanations, describing how specific development goals are achieved, independent of the placement of these steps within a development lifecycle. Processes take these method elements and relate them into semi-ordered sequences that are customized to specific types of projects.

[95] A task is an assignable unit of work. Every task is assigned to a specific role. The granularity of a task is generally a few hours to a few days and usually affects one or a small number of work products.

[96] A role defines a set of related skills, competencies, and responsibilities of an individual or individuals.

[97] A work product is a term that is used to describe task inputs and outputs.

7.2. Create Guidance Elements

7.2.1. Common

Guidance is a general term for supplemental information that can be added to most method and process elements. Guidance elements can also be associated with other guidance elements.

Guidance elements can be attached to **method content elements**: Work products, tasks and roles and provide information about how to create a work product, how to perform a task, how to perform a role and so on.

Guidance can also be attached to **the standard method content categories**: Disciplines, domains, work product kinds, role sets and tools.

Guidance elements can also be attached to the following **process elements**:

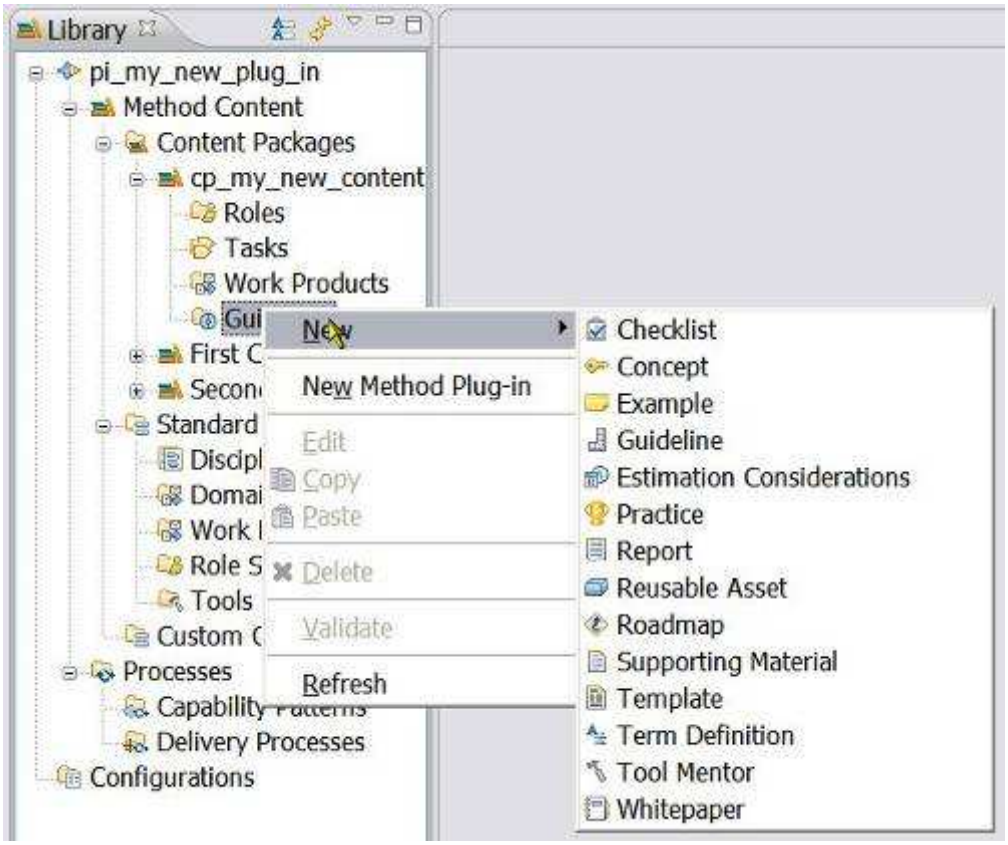
- Processes (Capability Patterns and Delivery Processes)
- Activity Descriptors (Iterations, Phases and Activities)
- Task Descriptors[99]

Since the guidance elements can applied to either method or process elements, they straddle both sides of the equation, aligning method content with their use in processes. On one hand they can provide additional free form information about how to deliver method content and on the other hand, guidance elements, such as checklists, examples, or roadmaps, can provide exemplary walkthroughs of a process.

[98] Guidance provides information about how to perform a role, how to create a work product, how to perform your task, and so on.

[99] The core content of activities is task descriptors and the steps of which they are composed. In addition, activities also contain role descriptors, work product descriptors, and milestones.

There are **fourteen types** of guidance elements:



The fourteen guidance element types are: Checklist, Concept, Example, Guideline, Estimation Consideration, Practice, Report, Reusable Asset, Roadmap, Supporting Material, Template, Term Definition, Tool Mentor and White Paper.

Types of guidance elements:

Table 4 - Guidance Element Types

Term	Description
Checklist	▪ Identifies a series of items that need to be completed or verified. Checklists are often used in reviews, such as walkthroughs or inspections.
Concept	▪ Outlines key ideas associated with basic principles underlying the referenced item. Concepts normally address more general topics than guidelines and span across several work product, tasks, or activities
Example	▪ Examples provide a model, a representative form or a typical example of a completed work product.
Guideline	▪ Provides additional detail about how to perform a particular task or grouping of tasks, or that provides additional details, rules, and recommendations about work products and their properties. ▪ Among others, a guideline can include details about best practices and different approaches for doing work, about how to use particular types of work products, information about different subtypes and variants of the work product and how they evolve throughout a lifecycle,

Term	Description
	discussions about skills the performing roles should acquire or improve upon, and measurements for progress and maturity.
Estimation Considerations	▪ Provides sizing measures or standards for sizing the work effort associated with performing a particular piece of work and instructions for their successful use. It may be comprised of estimation considerations and estimation metrics.
Practice	▪ Represents a proven way or strategy of doing work to achieve a goal that has a positive impact on the work product or process quality. ▪ Practices are defined orthogonally to methods and processes. They could summarise aspects that impact may different parts of a method or specific process.
Report	▪ A predefined template of a result that is generated on the basis of other work products as an output from some form of tool automation. An example for a report would be a use case model survey that is generated by extracting diagram information from a graphical model and textual information from documents and combines these two types of information into a report.
Reusable Asset	▪ Provides a solution to a problem in a given context. The asset may have a variability point, which is a location in the asset that might have a value provided or customised by the asset consumer. The asset has rules for usage that are the instructions describing how the asset should be used.
Roadmap	▪ Describes how a process is typically performed. Processes can often be much easier understood by providing a walkthrough of a typical instance of the process. ▪ In addition to making the process practitioner understand how the work in the process is being performed, a roadmap provides additional information about how activities and tasks relate to each other over time.
Supporting Material	▪ Used as a category for other types of guidance that are not specifically defined elsewhere. It can be related to all kinds of content elements, including other guidance elements.
Template	▪ Provides for a work product a predefined table of contents, sections, packages, headings, a standardised format, in addition to descriptions about how the sections and packages are supposed to be used and completed. Templates cannot only be provided for documents, but also for conceptual models or physical data stores.
Term Definition	▪ Terms define concepts used to enhance the Glossary. A term definition is not directly related to content elements, but its relationship is being derived when the term is used in the content elements description text.
Tool Mentor	▪ Shows how to use a specific tool to accomplish some piece of work, in the context of, or independently from, a task or activity.
White Paper	▪ A concept guidance that has been externally reviewed or published and can be read and understood in isolation of other content elements and guidance.

To create a guidance element:

1. Expand the **Content Package** in which you want to create your guidance.
2. Right-click Guidance and select **New**, then select a guidance type from the menu.
3. The following guidance types have the same editor including a **Description** tab, a **Guidance** tab, and a **Preview** tab: Concept, Example, Guideline, Estimating, Considerations, Report, Reusable Asset, Roadmap, Supporting Material, Template, Term Definition, Tool Mentor and Whitepaper (Not in the list: Checklist, Example and Practice).
4. Use the fields in the editor to specify the guidance details. Start by assigning a unique name to the guidance element along with a presentation name that is used as the external visible name when other elements refer to this element, or when the element is published. Every guidance type has specific content fields in the content editor that are distributed over two or more stacked tabs. Use the content fields to describe your guidance type.
5. Use the Rich Text Editor[100] to edit or enter the text for any field that has the Rich Text Editor icon  . Click the symbol to access the Rich Text Editor. Click the symbol again to close the Rich Text Editor.
6. In the **Icon** section, select a Node icon to be displayed with the guidance in the Library View[101] and Configuration View[102] and in the tree browser in a published Web site. The node icon should be 16 x 16 pixels. You can also select a shape icon to be displayed at the top of the published page for the guidance.
7. Click the **Guidance** tab. This part of the editor allows you to add and remove guidance for the element. To add guidance, click **Add**, select the guidance that you want to add, and click **OK**. To remove guidance, select it in the Guidance field and then click **Remove**. When you select a guidance element, the brief description of the guidance is displayed.
8. Click the **Preview** tab to view the newly created guidance and save by closing the tab.

It is possible to change the types of some of the guidance types to a limited number of other guidance types. To change a guidance type, in the Description tab under General Information section, click **Change Type** and select an available new type.

If the type of guidance is a **Practice**, you can add references to other elements in the References tab. If the guidance is a **Template**, you can attach multiple files or URLs to the guidance in the **Description** tab.

---

[100] The rich text editor provides text formatting functions that alter the appearance of published content. The CSS style sheet in a Web site controls the appearance of text at a global level, but the rich text editor can override those controls for specific text elements.

[101] The Library view displays the method content that is available in the current method library, which is organized into sets of method plug-ins and configurations.

[102] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

Related topics

Guidance Elements[103]
Create Method Content
Create a Role[104]
Create a Task[105]
Create a Work Product[106]

7.2.2. Practices

A practice is a documented approach to solving one or several commonly occurring problems, a proven way to achieve a goal.

A practice is published with a predefined tree-structure of elements that realise or relate to the practice. This allows practices to be displayed with a standard look and feel in the navigation views of the published site.

Practice navigation view:

A practice guidance element is published with the following ordered structure and hierarchy:

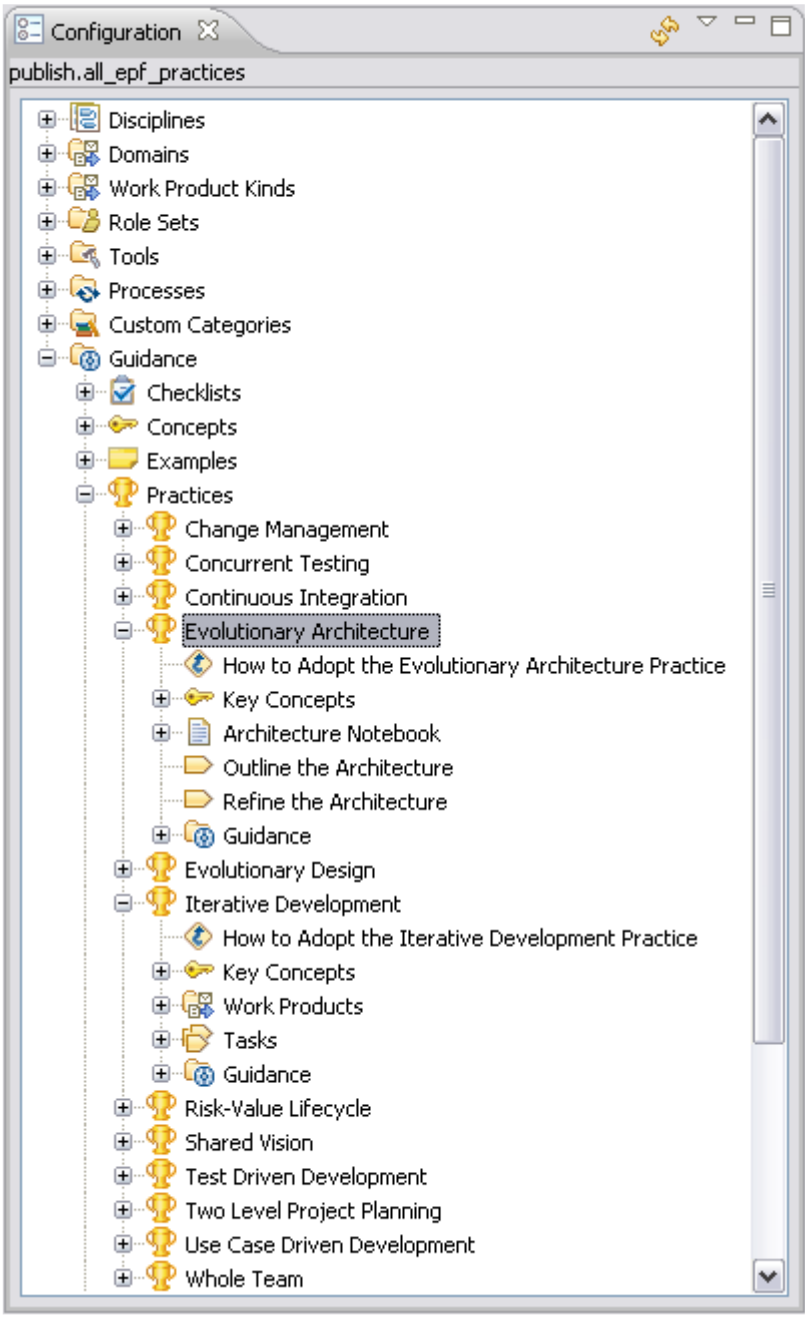
1. **List** of all **Roadmap** guidance elements associated to the practice.
2. **Folder** labelled **Key Concepts** with all associated concept guidance elements.
3. **Folder** with all **Work Product** elements associated to the practice.
4. **Folder** with all **Task** elements associated.
5. **Folder** labelled **Capability Patterns** with all associated capability pattern elements.
6. **Folder** with all **Role** elements associated.
7. **Folder** for remaining associated guidance elements:
   a. Sub-folder for every guidance type except concepts or roadmaps. For example, guideline, template, and so on.
   b. **Tool mentors** are organised in a Folder named **Tools**.
8. **List** of all **Categories** associated to the practice element.
   a. Sub-hierarchy of sub-categories and their categorised elements
9. ***Additional rules for the above topics:***
   a. If any of the folder elements listed above are empty, it will not be published.
   b. If any of the folder elements contain less than or three elements, no folder will be created, instead the elements will be published in a flat list in the same position as the folder.

[103] Guidance is a general term for supplemental information that can be added to most Method and Process elements. Guidance elements can also be associated with other guidance elements.

[104] A role defines a set of related skills, competencies, and responsibilities of an individual or individuals.

[105] A task is an assignable unit of work. Every task is assigned to a specific role. The granularity of a task is generally a few hours to a few days and usually affects one or a small number of work products.

[106] A work product is a term that is used to describe task inputs and outputs.



To create practice guidance elements:

1. Expand the **Content Package** in which you want to create your guidance.
2. Right-click **Guidance** and select **New** → **Practice** from the context menu.
3. Document your practice using the **Description** tab.
4. Select the **References** tab.
5. Click **Add** and select the method content elements and processes that you want to associate and present with your practice. The order in which the elements are listed is relevant in respect to the presentation rules defined above.
6. Click **OK** and **Save**. The new element you added to your practice is now displayed in the practice in the Configuration view.

Related topics

Creating Guidance Elements[107]
Guidance Relationships
Configuration View[108]
Method Configurations[109]

7.2.3. Glossary Entries

You can create a term definition in the same way as other types of guidance in a content package. Glossary entries are a type of guidance[110] called **term definition**.

If a content package is used in a configuration, all glossary elements contained in that package will be combined with terms contained in other packages of that configuration. The glossary in the published site will include these terms merged together from all packages and sorted alphabetically.

You can preview the glossary in the **Configuration** view by expanding **Guidance** → **Term Definitions**.

When a content package is removed from a configuration, all term definition elements in that package are removed from the configuration, and they are therefore no longer in the **Configuration** view.

When the *Publish glossary option* in the **Publish Method Configuration** wizard is selected, all term definitions in the configuration will be provided in the glossary used in the published site. If this option is cleared before publishing, the published site will not provide a glossary feature.

Create Glossary Entries

To create a glossary entry[111]:

1. In the Library view, right-click a Guidance element and select **New** → **Term Definition** . The **Guidance editor** panel opens.
2. The new term is named "*new_term_definition*" by default. Replace this text with the correct name of the term that you are adding, and enter the definition of this term in *Main description*.

[107] Guidance provides information about how to perform a role, how to create a work product, how to perform your task, and so on.

[108] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

[109] A method configuration is a selection of method plug-ins and method packages in a method library.

[110] Guidance is a general term for supplemental information that can be added to most Method and Process elements. Guidance elements can also be associated with other guidance elements.

[111] You can create a term definition in the same way as other types of guidance in a content package.

Edit Glossary Entries

To edit a glossary entry:

1. Right-click an existing term definition element and click **Edit**. (Double-clicking the icon for an element is a quicker way to launch the appropriate editor for that type of element.)
2. You can make changes to the term definition or attach additional guidance.
3. To save your changes and exit, click **File** → **Save**, or close the guidance editor window.

Delete Glossary Entries

Remember: Glossary entries must be deleted from the **Library** view. They cannot be deleted from the **Configuration** view.

To delete a glossary entry:

1. Right-click an existing term definition element and click **Delete**. The **Confirm Deletion** window opens.
2. Confirm your choice by clicking **OK**.

Related topics

Guidance Elements
Create Guidance Elements

7.2.4. Guidance Relationships

Guidance[112] is useful supplemental content of various types that can be attached to the following method elements: Roles, Tasks and Work products. The following table shows all possible relationships between work products, tasks, and roles, and the various types of guidance.

Essentially, all relationships listed are directed **from** one of the other **method elements to the guidance**. The one exception is the guidance type called practice, where the direction of the relationship is from practice to other types of method elements.

Table 5 - Guidance relationships

Guidance type	Work product	Task	Role
Checklists	X	X	X
Concepts	X	X	X
Estimation Considerations	X	X	
Examples	X	X	X
Guidelines	X	X	X
Practice	X	X	X

[112] Guidance is a general term for supplemental information that can be added to most Method and Process elements. Guidance elements can also be associated with other guidance elements.

Guidance type	Work product	Task	Role
Reports	X		
Reusable Asset	X	X	X
Supporting Materials	X	X	X
Templates	X		
Term Definitions			
Tool Mentors	X	X	
White papers	X	X	X

Note:

- **Practice** has a relationship to these elements, not from them.
- **Term Definitions** are a special type of guidance only used for Glossary items.

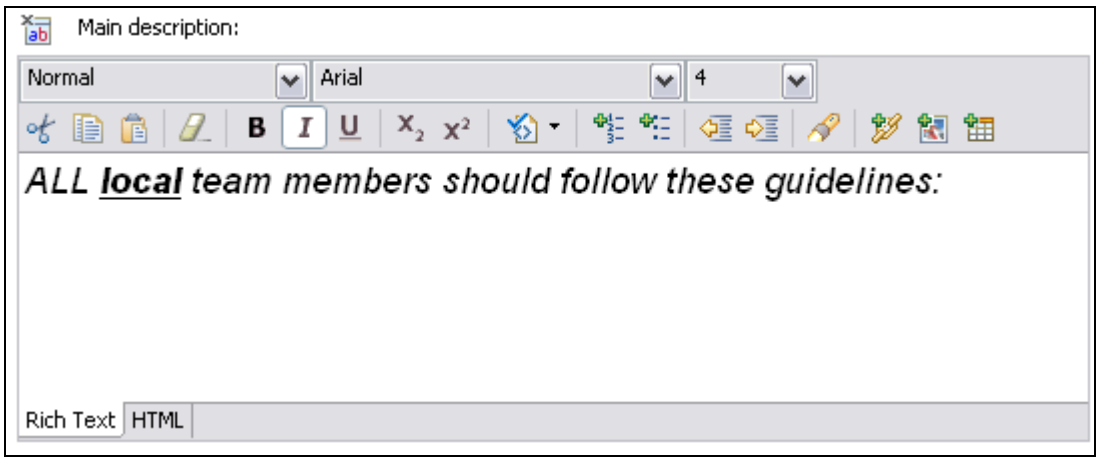
Related topics

Guidance Elements

7.3. Rich Text Editor

The rich text editor provides text-formatting functions that alter the appearance of published content. The CSS style sheet in a Web site controls the appearance of text at a global level, but the rich text editor can override those controls for specific text elements.

See the following example:



Button functions

You can use pull-down menus in the editor to apply specific styles and specify a font and a font size.

You can use the buttons in the editor to perform following functions:

- Cut text and save it to the clipboard, removing the original.
- Copy text to the clipboard, leaving the original text in place.

- Paste text from the clipboard. You can also right-click and select Paste as Plain Text.
- Remove all the contents in the text editor window.
- Make selected text bold.
- Make selected text italic.
- Make selected text underlined.
- Make selected text superscripted.
- Make selected text subscripted.
- Run cleaning utility (see below for more detail)
- Add an ordered list.
- Add an unordered list.
- Indent selected text.
- Outdent selected text.
- Find and replace text. The Find/Replace window opens. Type the text that you want to find in the Find field and, if you are replacing that text, type the replacement text in the Replace with field. You can search forward or backward and specify whether you want your search to be case-sensitive or search for parts of words. (This functionality may be limited on Linux systems.)
- Add or create hyperlinks. For more information about hyperlinks, see Add a Reference or Hyperlink.
- Add an image. The Add Image window opens. Browse for the image that you want to add and click OK.
- Add a table. The Add Table window opens. Specify the table parameters and click OK.

Clean HTML

There are three options when using the Clean HTML toolbar button:

- Default Clean
- Clean MS HTML
- Clean Word 2000 HTML

Click the button to use the first option (Default Clean). Use the drop-down arrow to select the other two options. Use Clean MS HTML for cleaning HTML that has been pasted from Microsoft Word. Use Clean Word 2000 HTML only for cleaning up HTML that has been pasted from Microsoft Word 2000.

Related topics

Add a Reference or Hyperlink

7.3.1. Add References or Hyperlinks

You can add three types of references or hyperlinks to your content text:

- References to other elements in the method library
- Reference to files imported into the method library

- References to external text resources that reside outside the method library

Text fields that support hyperlinks have a Rich Text Editor icon  next to the text field's label. Click the symbol to expand the field into a full text editor.

To insert a hyperlink to another element in the method library into a text field, locate the element that you want to create a link to in the Library or Configuration view, and drag it into the appropriate text field. You can also drag a method element into an unexpanded text field that supports hyperlinks with the same result.

You can use the **Add Link** function in the rich text editor to add links that are references to other method elements or links to files or Web pages. This gives you more options for how your links are displayed.

1. In the text editor, position the cursor where you want to insert the link and in the tool bar click **Add Link**  . The Add Link window opens.
2. Click **Browse** to locate and select the element you want to create a link to. There are several options for creating links to other content elements, with each option offering a different way for displaying the link's name:
   - Method element: The link displays the element name.
   - Method with type prefix: The link displays the element name and type.
   - Method element with custom text: The link displays user-defined text. To define this text, manipulate the text in between the **<a>**...**** tags in the URL text field after you select an element using Browse.
3. To insert a hyperlink to an external URL, from the Type drop down list, select URL and in the URL field type the URL details, and click OK.
4. To insert a hyperlink to a file, in the Type drop down list, select **File**, enter the file name or browse to find the file. The file that you select is **copied** from its source into the method library. It will be included in the Web site upon publication.
5. Each time you add a link, you can choose to have the target open in the ***same browser window or in a new window***. Click OK to close the window and insert the link.

Related topics

Rich Text Editor[113]

7.4. Method Content Variability

Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

You can use variability to customise configurations that use method content and processes that you do not own and cannot directly modify. When content packages are

[113] The rich text editor provides text formatting functions that alter the appearance of published content. The CSS style sheet in a Web site controls the appearance of text at a global level, but the rich text editor can override those controls for specific text elements.

upgraded, you can import them and reapply in a single step the customisation that you made earlier, without going through each element.

Variability generally affects two characteristics of a method element; its attributes and its relationships with other content elements. If an element supports variability, the specification is shown at the bottom of the element's Description view.

There are three factors to be considered when using variability:

- **Attributes**: Element data types such as Main Description.
- **Incoming associations**: Associations from other elements. The associated element may have one or more references to the subject element.
- **Outgoing associations**: Associations to other elements. The subject element may have one or more references to the associated element.

For a complete list of supported associations for each type of element, see Associations Impacted by Variability.

Variability Type

Variability type describes how one element affects another through variability associations. The five types of variability associations are:

Variability Type	Association Description
Not Applicable	▪ The element is a base element and does not affect another element through variability. This is the default value of an element's variability type.
Contributes	▪ A contributing element adds to the base element. Contributes provides a way for elements to contribute with their properties to their base element without directly changing any of its existing properties. ▪ The base appears in the published Web site but the contributing element does not. In and out relationships from the contributing element are added to the base. Text from the contributing element is appended to corresponding base sections. ▪ When the contribution is resolved during publication or by using the Browsing perspective, the base element is logically replaced with an element that combines the attributes and associations of the contributing element with the base element.
Replaces	▪ A replacing element replaces parts of the base element. It provides a mechanism for an element to replace a base element without directly changing any of the base element's existing properties. ▪ The replacer appears in the published Web site but the base element does not. “Out Relationships” in the replacer are left untouched, and those of the base element are ignored. “In Relationships” from the base are added to the replacer. Text in the replacer is left untouched, and the base element's text is ignored.
Extends	▪ An extending element inherits characteristics of the base element. Both the extender and the base element appear in the published Web site. ▪ Out relationships from the base are added to the extender. In relationships in the extender are left untouched, the base element's are ignored. Text is added from the base if the extender has no value defined for the given section.

Variability Type	Association Description
	▪ Provides a mechanism for method plug-ins to reuse elements from a base plug-in through a kind of inheritance. Attribute values and associations are inherited from the "based-on" element to the extending element. The result is that the extending element has the same properties as the "based-on" element, but might define its own additions.
Extends and Replaces	▪ This variability relationship combines the effects of extends and replace variability into one variability type. ▪ Whereas the replaces variability completely replaces all attributes and outgoing association instances of the base variability element with new values and instances, or removes all values or association instances if the replacing element does not define any, **extends and replaces variability only replaces the values that have been redefined and leaves all other values of the base element as is**.

You can use the graphical display to navigate to any of the variability elements, the complete hierarchy of variability associations for an element can be displayed graphically.

Related topics

Associations Impacted by Variability
Browsing Variability Relationships[114]
Contributes Variability[115]
Replaces Variability[116]
Extends Variability[117]
Extends and Replaces Variability

7.4.1. Contributes Variability

A contributing element adds to the base element. Contributes provides a way for elements to contribute their properties into their base element without directly changing any of its existing properties, such as in an additive fashion. When the contribution is resolved during publication or by using the Browsing perspective, the base element is logically replaced with an element that combines the attributes and associations of the contributing element with the base element.

[114] You can use the graphical display to navigate to any of the variability elements.

[115] A contributing element adds to the base element. Contributes provides a way for elements to contribute their properties into their base element without directly changing any of its existing properties, such as in an additive fashion. When the contribution is resolved during publication or by using the Browsing perspective, the base element is logically replaced with an element that combines the attributes and associations of the contributing element with the base element.

[116] The Replaces variability association provides a mechanism for an element to replace a base element without directly changing any of the base element's existing properties.

[117] Extends associations provide a mechanism for method plug-ins to reuse elements from a base plug-in through a kind of inheritance. Attribute values and associations are inherited from the "based-on" element to the extending element. The result is that the extending element has the same properties as the "based-on" element, but might define its own additions.

Relevant information in the contributing element is added to the base element. Text fields in the contributing element are appended at the end of the respective text fields in the base element. For example, text in a brief description is appended to the brief description in the base.

A contributing element only adds attributes and associations to the base element. It never overrides or replaces existing attributes or other association in the base. If the base element is defined with an existing association to exactly one other element, the existing association will not be replaced by a contributor's association.

Contribution rules

- Outgoing associations from the contributing element are added to the base element.
- Incoming associations from the contributing element are added to the base element.
- Attributes from the contributing element are appended to the base element, except for identifying or naming attributes and non-textual attributes such as Boolean or date.
- A base element can receive contributions from multiple elements.
- Contributions are transitive. This means a contributing element that has its own contributing elements will add them to the base.

Exceptions

All contributing associations are "many to many" except for the following:

- Work products can only be assigned to a single domain. When contributing to a work product, if the work product at the base is assigned to a domain, the relationship to domain from the contributor is ignored. There is no exception when contributing to domain elements.

A complete list of all possible associations is provided on the Associations Impacted by Variability page.

Add a contribution association

1. Use the Content Variability section on the Description tab to make an element contribute to another element.
2. Select Contributes as the Variability type and then select the base element to which this element will contribute. The base element must be the same type of element as the contributing element.

Related topics

Method Content Variability[118]
Associations Impacted by Variability
Browsing Variability Relationships[119]

[118] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

[119] You can use the graphical display to navigate to any of the variability elements.

7.4.2. Extends Variability

Extends associations provide a mechanism for method plug-ins to reuse elements from a base plug-in through a kind of inheritance. Attribute values and associations are inherited from the "based-on" element to the extending element. The result is that the extending element has the same properties as the "based-on" element, but might define its own additions.

Extends is not used to modify content of the base plug-in, but to provide the ability for the extending plug-in to define its own content, which is a variant of content already defined. An example of this is a special version of a generic Review Record for a specific type of review. The effect of this is that the base element and any number of extending elements can be used side-by-side, but refer to each other through the extends relationship.

Extends also provides the key mechanism for binding capability patterns to processes. A pattern is applied by defining an Extends relationship from an activity of the applying processes to the capability pattern. The activity inherits association instances from the pattern and the pattern seems to be part of the resulting process after publication or by using the Browsing perspective.

Relevant information in the extending element is added to the base element in creating an additional element. Text fields in the extending element are appended at the end of the respective text fields in the base element.

Extends only defines inheritance for the extending element. The base element remains untouched. If the extending element is allowed an association to only one other element and has such an element defined already, inheritance will not override this existing association.

Extends association rules

- Outgoing associations from the base element are inherited by the extending element.
- Incoming associations from the base element are not inherited by the extending element.
- Attribute values of the base element are inherited by the extending element if the extending element has not defined its own values.
- Extends relationships are transitive. If an extending element has its own extending associations, the second extension inherits attributes from its direct and indirect base elements.

[120] The Replaces variability association provides a mechanism for an element to replace a base element without directly changing any of the base element's existing properties.

[121] Extends associations provide a mechanism for method plug-ins to reuse elements from a base plug-in through a kind of inheritance. Attribute values and associations are inherited from the "based-on" element to the extending element. The result is that the extending element has the same properties as the "based-on" element, but might define its own additions.

- Contribution associations are resolved before Extends associations. Contribution is evaluated first and then extending elements inherit afterwards from the base element, including all of its contributions.
- Replace precedes Extends. If a base element has both replace and extends relationships, the extending element inherits from the replacing element.

Exceptions

All extending associations are "many to many" except for the following association:

- Work Products can only be assigned to a single Domain element. No additions are made from the base when a Domain is extended. There are no exceptions when extending a Work Product.

A complete list of supported associations for each type of element is provided on the Associations Impacted by Variability page.

Add and extend association

1. To make an element extend another element, on the **Description** page use the **Content Variability** section.
2. Select Extends as the Variability type, and then select the base element that this element extends. Note that the base element must be the same type of element as the extending element.

Related topics

Method Content Variability[122]
Associations Impacted by Variability
Browsing Variability Relationships[123]
Contributes Variability[124]
Replaces Variability[125]
Extends and Replaces Variability

7.4.3. Replaces Variability

The Replaces variability association provides a mechanism for an element to replace a base element without directly changing any of the base element's existing properties.

A replacing element replaces relevant attributes in the base element when the configuration is published or examined in the Browsing perspective. In most cases this is used for method plug-ins as a way to replace specific content elements such as roles, tasks, or activities with a completely new variant. Replacement can also be used to

[122] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

[123] You can use the graphical display to navigate to any of the variability elements.

[124] A contributing element adds to the base element. Contributes provides a way for elements to contribute their properties into their base element without directly changing any of its existing properties, such as in an additive fashion. When the contribution is resolved during publication or by using the Browsing perspective, the base element is logically replaced with an element that combines the attributes and associations of the contributing element with the base element.

[125] The Replaces variability association provides a mechanism for an element to replace a base element without directly changing any of the base element's existing properties.

change the fundamental relationships of these elements. The base content element will be logically replaced with the new replacing element to which all incoming associations still point as before, but which has potentially new attribute values and outgoing associations.

When you replace an element, all description attributes are replaced by corresponding attribute values from the replacing element. If a description attribute in the replacing element is blank but the base element has content, the attribute will be blank in the resulting element.

Replace always replaces attributes and associations of the base element with the replacing element's attributes and associations, except for incoming associations, which are only added to the base, but do not replace the base's incoming associations.

Replace association rules

The following rules are for replacing associations:

- All outgoing associations from the base element are replaced with associations of the replacing element. If the replacing element does not define any, then the resulting element will also not have any.
- Incoming associations from the base element are added to the replacing element.
- Attributes of the base element are replaced with attributes of the replacing element including the base element's identifier.
- A base element of a replacement can have only one replacing element per configuration. If more than one replacing element is present no replacement takes place.
- Replacement is transitive. If a replacing element is replaced itself, the final replacer prevails.
- Contribution precedes Replacement. Contribution associations are resolved first and then replacement is performed afterward. The evaluation of contribution and replacement is performed in serial order in the specialisation hierarchy.

Exceptions

All replacing associations are "many to many" except for the following association:

- Work Products can only be assigned to a single Domain. If the replacing Work Product is assigned to a Domain, the relationship from the base is ignored. There is no exception when replacing a Domain element.

For more information about the associations for each element type, see Associations Impacted by Variability.

Add a replace association

1. On the Description tab, use the Content Variability section to make an element replace another element.
2. Select Replaces as the Variability type and select the base element that this element will replace. Note that the base element must be the same type of element as the replacing element.

Related topics

Method Content Variability[126]
Associations Impacted by Variability
Browsing Variability Relationships[127]
Contributes Variability[128]
Extends Variability[129]
Extends and Replaces Variability

7.4.4. Extends and Replaces Variability

The Extends and Replaces variability relationship combines the effects of Extends[130] and Replaces[131] variabilities into one variability type. Whereas Replaces variability completely replaces all attributes and outgoing associations of the base element with new values and instances, or removes all values or associations if the replacing element does not define any, **Extends and Replaces** variability only replaces values that have been redefined. All other values of the base element are unaffected.

In other words, Extends and Replaces allows users to selectively replace specific attributes and associations of the base elements. This type of variability can be used to generate method plug-ins that rename elements, or replace some descriptions of method elements with new ones, without completely remodelling all other relationships and attributes needed by the base plug-in.

Extends and replaces association rules

- Extends and Replace variability combines the effects of the Extends and the Replaces variabilities. The evaluation first performs the effects of the Extends and then the effects of the Replaces variability. This implies that:
  - **First**, the new element will inherit all attributes and associations from the base element.
  - **Second**, the new elements might override inherited attributes or associations.
  - **Third**, the base element will be replaced with new element using the overridden values and if no override was specified, keep the inherited values.

---

[126] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

[127] You can use the graphical display to navigate to any of the variability elements.

[128] A contributing element adds to the base element. Contributes provides a way for elements to contribute their properties into their base element without directly changing any of its existing properties, such as in an additive fashion. When the contribution is resolved during publication or by using the Browsing perspective, the base element is logically replaced with an element that combines the attributes and associations of the contributing element with the base element.

[129] Extends associations provide a mechanism for method plug-ins to reuse elements from a base plug-in through a kind of inheritance. Attribute values and associations are inherited from the "based-on" element to the extending element. The result is that the extending element has the same properties as the "based-on" element, but might define its own additions.

[130] Extends associations provide a mechanism for method plug-ins to reuse elements from a base plug-in through a kind of inheritance. Attribute values and associations are inherited from the "based-on" element to the extending element. The result is that the extending element has the same properties as the "based-on" element, but might define its own additions.

[131] The Replaces variability association provides a mechanism for an element to replace a base element without directly changing any of the base element's existing properties.

- If the extends and replaces element defines outgoing associations, they replace all outgoing associations of the base elements. If the extends and replaces element does not define any new associations, the resulting element retains the associations of the base element.
- Incoming associations from the base element are added to the replacing element.
- If the extends and replaces element defines attributes, these attributes are replaced in the resulting element including the base element's identifier. Undefined attributes retain values used in the base element.
- The base element of a replaces relationship or an extends and replaces relationship can have only one replaces or extends and replaces element per configuration. If more than one element is present, no replacement takes place.
- The extends and replaces relationship is transitive and evaluated top-down relative to the direction of the replacement. If a replacing element is also replaced, the final replacing element prevails.
- Contributes variability relationships are resolved before replaces and extends and replaces relationships. Extends relationships are resolved last. Variability is always resolved top-down from the base to the variability elements. Within the same level, contributes relationships are resolved first. Replaces or extends and replaces are resolved afterwards.

How to add an extends and replaces association

1. Use the Content Variability section on the Description tab to make an element extend and replace another element.
2. Select Extends and Replaces as the Variability type and select the base element that this element will replace. The base element must be the same type of element as the replacing element.

Related topics

Method Content Variability[132]
Associations Impacted by Variability
Browsing Variability Relationships[133]
Extends Variability[134]
Replaces Variability[135]
Contributes Variability[136]

---

[132] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

[133] You can use the graphical display to navigate to any of the variability elements.

[134] Extends associations provide a mechanism for method plug-ins to reuse elements from a base plug-in through a kind of inheritance. Attribute values and associations are inherited from the "based-on" element to the extending element. The result is that the extending element has the same properties as the "based-on" element, but might define its own additions.

[135] The Replaces variability association provides a mechanism for an element to replace a base element without directly changing any of the base element's existing properties.

[136] A contributing element adds to the base element. Contributes provides a way for elements to contribute their properties into their base element without directly changing any of its existing properties, such as in an additive fashion. When the contribution is resolved during publication or by using the Browsing perspective, the base element is logically replaced with an element that combines the attributes and associations of the contributing element with the base element.

7.4.5. Associations Impacted by Variability

The following table shows all possible associations between method content elements that can be impacted by variability. **Outgoing** associations are shown with a right-arrow →. **Incoming** associations are shown with a left-arrow ←.

Unless noted in the table, all associations are "many to many." For example, there can be many associations from a Guidance element to another Guidance element.

The few exceptions are noted with a "1" next to the arrow indicating that only one association is allowed in that direction. For example, there can be only one incoming association from a Task primary performer to a Role. Similarly, there can be only one outgoing association from a Task to a Role primary performer.

Table 6 - Variability associations

Element	Association direction	Associated element
Guidance	→	Guidance
	→	Section
	←	Any method content element
	←	Any standard category
	←	Custom category
	←	Guidance
	←	Guidance - Practice
	←	Tool
	←	Tool Mentor
Guidance - Checklist	→	Check Item
Guidance - Practice	→	Any category
	→	Any method content element
Role	→	Guidance
	→	Section
	→	Work product: responsible for
	←	Custom Category
	←	Guidance - Practice
	←	Role Set
	←	Task: additional performer
	←	Task: primary performer
Task	→	Guidance
	→	Role: additional performer
	→	Role: primary performer
	→	Section
	→	Step
	→	Work Product: mandatory input

Element	Association direction	Associated element
	→	Work Product: optional input
	→	Work Product: output
	←	Custom category
	←	Discipline
	←	Guidance - Practice
Tool Mentor	→	Guidance
	←	Tool
Work Product	→	Guidance
	→	Section
	←	Custom category
	← 1	Domain
	←	Guidance - Practice
	←	Role
	←	Task: mandatory input
	←	Task: optional input
	←	Task: output
	←	Work Product - deliverable
	←	Work Product kind
Work Product - Artefact	→	Work Product - Artefact
	←	Work Product - Artefact
Work Product - Deliverable	→	Work Product
Activity	→	Activity
	←	Activity
Custom Category	→	Any method content element
	→	Any standard category
	→	Custom category
	→	Discipline Grouping
	→	Role set grouping
	→	Tool
	←	Custom Category
	←	Guidance - Practice
Discipline	→	Discipline
	→	Guidance
	→	Reference Workflow
	→	Task

Element	Association direction	Associated element
	←	Custom category
	←	Discipline
	←	Discipline grouping
	←	Guidance - Practice
Discipline Grouping	→	Discipline
	←	Custom category
Domain	→	Domain
	→	Guidance
	1 →	Work product
	←	Custom category
	←	Domain
	←	Guidance - Practice
Role Set	→	Guidance
	→	Role
	←	Custom category
	←	Guidance - Practice
	←	Guidance - Practice
Role Set Grouping	→	Role Set
	←	Custom Category
Tool	→	Guidance
	→	Tool mentor
	←	Custom category
	←	Tools
	→	Tools
Work Product Kind	→	Guidance
	→	Work Product
	→	Work Product Kind
	←	Custom category
	←	Guidance - Practice
	←	Work Product kind

Related topics

Method Content Variability[137]

[137] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

7.4.6. Browsing Variability Relationships

You can use the graphical display to navigate to any of the variability elements. The complete hierarchy of variability[142] associations for an element can be displayed graphically.

To browse variability relationships:

1. Switch to the **Authoring** perspective if you are not already using it 

.
2. Right-click any task icon  in the Library, then click **Open via variability elements**. A window will be displayed which shows the hierarchy of content packages with variability elements related to the selected element. For each element you can see an icon indicating the element's type, name, and variability associations (shown in parentheses).

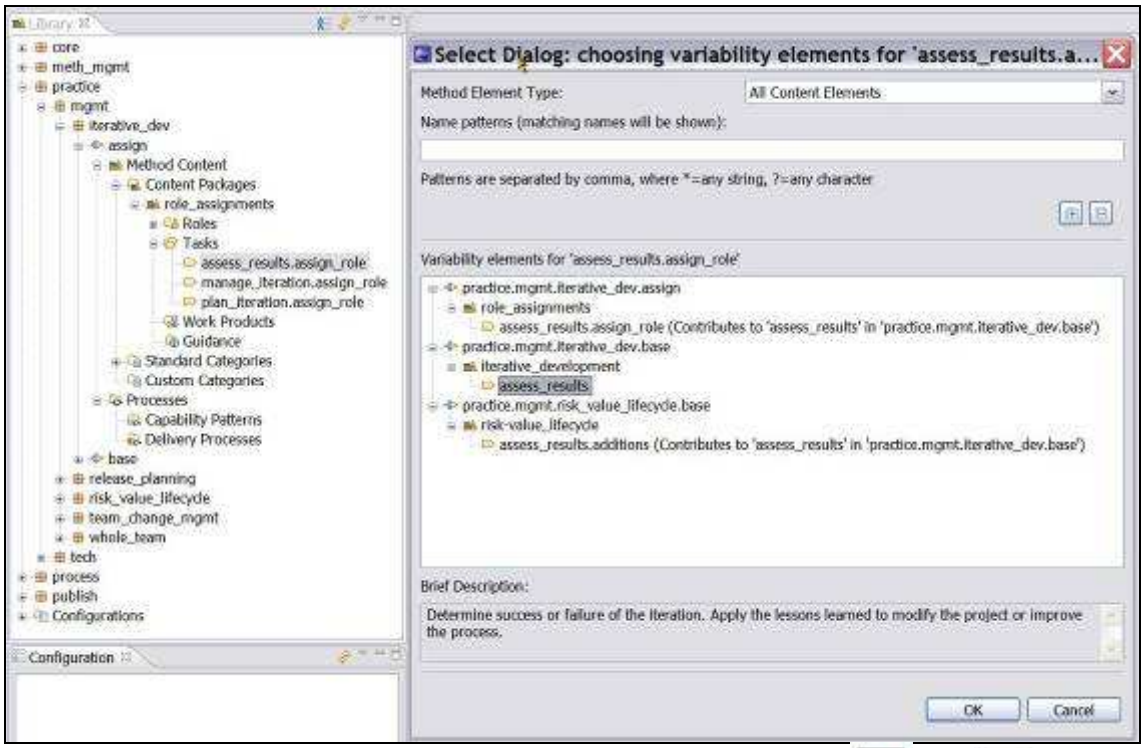
[138] You can use the graphical display to navigate to any of the variability elements.

[139] A contributing element adds to the base element. Contributes provides a way for elements to contribute their properties into their base element without directly changing any of its existing properties, such as in an additive fashion. When the contribution is resolved during publication or by using the Browsing perspective, the base element is logically replaced with an element that combines the attributes and associations of the contributing element with the base element.

[140] Extends associations provide a mechanism for method plug-ins to reuse elements from a base plug-in through a kind of inheritance. Attribute values and associations are inherited from the "based-on" element to the extending element. The result is that the extending element has the same properties as the "based-on" element, but might define its own additions.

[141] The Replaces variability association provides a mechanism for an element to replace a base element without directly changing any of the base element's existing properties.

[142] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.



3. You can expand all or collapse all associations by using the  and  buttons above the tree browser.
4. You can either click once and click **OK** or double-click any element to open the appropriate editor for that type of element.

Related topics

Method Content Variability[143]
Associations Impacted by Variability
Contributes Variability[144]
Extends Variability[145]
Replaces Variability[146]
Extends and Replaces Variability

[143] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

[144] A contributing element adds to the base element. Contributes provides a way for elements to contribute their properties into their base element without directly changing any of its existing properties, such as in an additive fashion. When the contribution is resolved during publication or by using the Browsing perspective, the base element is logically replaced with an element that combines the attributes and associations of the contributing element with the base element.

[145] Extends associations provide a mechanism for method plug-ins to reuse elements from a base plug-in through a kind of inheritance. Attribute values and associations are inherited from the "based-on" element to the extending element. The result is that the extending element has the same properties as the "based-on" element, but might define its own additions.

[146] The Replaces variability association provides a mechanism for an element to replace a base element without directly changing any of the base element's existing properties.

7.5. Copyright Notices

You can specify the default copyright for all elements in a particular method plug-in by using the method plug-in editor.

Copyright notices are stored as supporting material elements. For more information about using the method plug-in editor to change the default copyright, see Change Default Copyright Notice.

Sometimes elements in your method plug-in may require a different copyright than the default one. For this procedure, see Override Default Copyright Notice[147].

Frequently, a method plug-in contains contributing elements with a different copyright than the base elements that receive the contribution. When there are multiple contributing elements, each with its own copyright notice, the individual copyrights are appended to each other in the generated page on the published site.

When you create a new method plug-in, you should also create a supporting materials guidance element with the appropriate copyright information. For this procedure, see Create Copyright Notices.

Related topics

Create Copyright Notices
Change Default Copyright Notice
Override Default Copyright Notice[148]
Method Content Variability[149]

7.5.1. Create Copyright Notice

Copyright notices are stored as supporting material elements. When you create a new method plug-in, you should also create a supporting materials guidance element with the appropriate copyright information. You can then reference this element from the plug-in's Copyright field.

To create copyright notice in new method plug-ins:

1. In the Library view panel, use the tree browser to locate the method plug-in that you want to work on and double-click to open the editor.
2. Under Version Information section, click Select to the right of the Copyright field.
3. Select the appropriate copyright element and click OK.
4. Close the method plug-in editor and save your changes.

Related topics

Copyright Notices[150]

[147] Sometimes elements in your method plug-in may require a different copyright than the default for the plug-in. You can override the default copyright.

[148] Sometimes elements in your method plug-in may require a different copyright than the default for the plug-in. You can override the default copyright.

[149] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

[150] You can specify the default copyright for all elements in a particular method plug-in by using the method plug-in editor.

Changing the Default Copyright Notice
Overriding the Default Copyright Notice
Method Content Variability

7.5.2. Change Default Copyright Notice

Remember: The default copyright notice can only be changed in editable method plug-ins.

To change the default copyright notice:

1. In the Library view panel, use the tree browser to locate the method plug-in  that you want to work on.
2. Expand the method plug-in node until you see the icon for the guidance element.
3. Expand the Guidance node and double-click the supporting material icon. The guidance editor opens.
4. Under Version Information, select the copyright notice from the Copyright field and click Deselect.
5. Click Select and select a different copyright notice from the list.
6. Click the Preview tab to view your copyright notice.
7. Close the window and save your changes.

Related topics

Copyright Notices[151]
Create Copyright Notice
Change Default Copyright Notice
Override Default Copyright Notice
Method Content Variability

7.5.3. Override Default Copyright Notice

Sometimes elements in your method plug-in may require a different copyright than the default for the plug-in. You can override the default copyright.

To override the copyright notice in a method plug-in:

1. Create a new supporting material element for the new copyright information.
2. Associate this new element with the Copyright field for the specific element.

Related topics

Copyright Notices
Create Copyright Notice
Change Default Copyright Notice
Method Content Variability

[151] You can specify the default copyright for all elements in a particular method plug-in by using the method plug-in editor.

7.6. Method Content for Publishing

7.6.1. Create Index Entries

Index entries are created by adding a special HTML anchor tag next to the target for the index entry. The anchor tag should start with the characters **XE_** followed by a **keyword**. Keywords are collected into a file and are displayed in alphabetical order.

Keyword syntax

The syntax of a keyword is **[***prefix***][***level1***][***levelseparator***][***level2***]**.

Each keyword starts with a prefix. The prefix is specified in the input file to the application. A keyword can consist of one or more levels. A keyword with more than one level is useful when you would like to group different keywords. After the prefix, the keyword at level one follows. If a keyword has more than one level a separator between the levels is needed. The separator is specified in the input file to the application. Then the next level follows.

Remember: Space characters inside keywords must be replaced with "_" characters. In the generated index, "_" characters are converted back to space characters.

The following table shows examples of valid keyword definitions. The centre column shows how the keywords are presented in the generated index.

Table 7 - Table 1. Syntax for index keywords

XE_Copyright	**Copyright**	****
XE_KeyWordIndex__Syntax_description XE_KeyWordIndex__Input_file_syntax	KeyWordIndex Input file syntax Syntax description	** **
XE_Users_guide__Chapter_1 XE_Users_guide__Chapter_2	Users guide Chapter 1 Chapter 2	** **

Keywords must be manually entered into the file that contains the index target. If you are using the rich text editor (RTE), switch to **HTML** editing mode to make these targets. Index targets are created by adding an HTML anchor tag in front of the index target. The general pattern is **. The previous table shows specific examples of this syntax.

Internal references

Index entries can point to other index entries rather than a page in the published site. These are internal references within the index itself. In the following example, the entry "software architect" under "roles" links to the second entry for "software architect," not to content page in the published site.

roles software architect [...] software architect indexentry1 indexentry2

Internal references are collected in a separate file and follow a special syntax. Each line in the file consists of two keywords with a "tab character (\t)" separator. The first keyword refers to the second keyword in the resulting index. A keyword can only occur once, as a referring keyword, in the file. The second keyword is only allowed to be a one level keyword. The following table shows this relationship:

XE_KeyWord__Author	XE_Author

XE_KeyWord__Copyright	*XE_Copyright*

If the referring (second) keyword does exist in an HTML file, the index entry is named "See also ...". If the referring (second) keyword does not exist in an HTML file index entry will be named "See ...".

The previous example results in the following index.

Author Copyright KeyWord	*Author, See Author*	*Copyright, See Copyright*

Suppressing index generation

By default, published Web sites include an index, however, this feature can be cleared before publishing. If the check box **Include index page** is cleared, the published site will not include this feature. This check box is one of the publishing options in **Publish Method Configuration**.

Configuration file

The KeyWordMap application takes a number of parameters as input. These parameters are specified in a configuration file.

Each row in the configuration file defines one parameter. The order of the rows does not matter. Each part of a row is separated with a "tab character (\t)". The name of the parameters is not case-sensitive. Space characters to the left or right of an item will be removed.

Parameter	Setting	Description
wwwroot	d:/work/rup	This is the top directory of the directory structure to traverse. Normally this is the top directory of the site. This parameter is mandatory.
relativepath	..	Defines the relative path from where the result of this application is written to the files it refers to. If the result is written to a directory one level below the top of the site, this parameter has the value "..". This parameter is mandatory.
mainresultfile	/index/index.htm	If this parameter and the parameter "indexresultfile" is se,t a frameset is generated into this file. This is useful when the site contains many keywords. A faster navigation among the keywords is possible. The value of this parameter together with the value of the parameter "wwwroot" shall be a valid path for a file. This parameter is optional.
indexresultfile	/index/navig.htm	If this parameter and the parameter "mainresultfile" shortcuts to the first keyword of each letter will be generated into this file. A faster navigation among the keywords possible. The value of this parameter together with the value of the parameter "wwwroot" ise a valid path for a file. This parameter is optional.
keywordresultfile	/index/contents.htm	The actual keyword index is written into this file. The value of this parameter together with the value of the parameter "wwwroot" is a valid path for a file. This parameter is mandatory.
maintitle	Index - Rational Unified Process	The title of the main index page. This parameter is optional. The default value is "Index - Rational Unified Process".
target	ory_doc	The name of the target in which the html documents containing the actual keywords is to be shown. This parameter is mandatory.

Parameter	Setting	Description
index target	index	The name of the target in which the file defined by the parameter "indexresultfile" is written. This parameter is mandatory if the parameter "indexresultfile" is set.
keywordtarget	keyword_t	The name of the target in which the file defined by the parameter "keywordresultfile" is written. This parameter is mandatory if the parameter "keywordresultfile" is set.
indexheight	25	When the parameter "indexresultfile" is set, a frameset is created. This parameter defines the height (in pixels) of the frame in which the "indexresultfile" is shown. This parameter is mandatory if the parameter "indexresultfile" is set.
headerfile	keywordpreamble.txt	This parameter makes it possible for the user to customise this application as much as possible. This parameter is the name of a file that serves as the header for the file defined by the parameter "keywordresultfile". The value of this parameter is either a relative path from where this program is executed or an absolute path. This parameter is mandatory. If you use a cascading-styles-sheet file to control the layout of the index, then the headerfile must contain a link to that cascading-style-sheet file.
footerfile	keywordpostamble.txt	This parameter makes it possible for the user to customise this application as much as possible. This parameter is the name of a file that serves as the footer for the file defined by the parameter "keywordresultfile". The value of this parameter is either a relative path from where this program is executed or an absolute path. This parameter is mandatory.
keywordfile	keywords.txt	This parameter defines the internal references within the index. For more information, see the Internal references. This parameter is optional.
leavedir	_borders	This parameter can occur in the definition file more than once. It is used to define which directories that will not be searched for keywords. The path is relative to the directory defined by the parameter "wwwroot". This parameter is optional.
mulitdocumentkeyword	false	This parameter defines if a keyword is allowed to be in more than one document. The default value is false. This parameter is optional.
showdocumenttitle	false	This parameter defines if the text of the link to a document containing this keyword will be the name of the keyword or the title of the document. Default is false. True is only applicable if the value of the parameter "mulitdocumentkeyword" is set to true. This parameter is optional.
prefix	XE_	With this parameter you can define the prefix of the keywords this application will search for. This parameter is optional. The default prefix is "XE_".
levelseparator	__	With this parameter you can define the separator between different levels in a keyword this application will search for. This parameter is optional. The default levelseparator is "__".
see	, See	With this parameter you can define the text before a keyword when it is referring to another keyword. This parameter is optional. The default levelseparator is ", See".
seealso	, See also	With this parameter you can define the text inserted before a keyword when it is referring to another keyword. This parameter is optional. The default levelseparator is ", See also".
levelstyle levelstyle	defaultstyle indexlevel 2 1 indexlevel1	This parameter defines which style in a cascading style sheet (CSS) will be used to for the different keyword levels. The syntax to use is [keywordlevel/reservedname][stylename]. - [keywordlevel/reservedname] is either the level of the

Parameter	Setting	Description
		keyword (1, or 2, or 3, ...). Alternately you can use one of the two reserved names, ("defaultstyle" or "headlinestyle"). - [stylename] is the name of the style. The KeyWordIndex application adds a prefix "p." to the stylename. This means if the style sheet defines define styles for the classes "p.indexlevel1", and "p.indexlevel2", then [stylename] is "indexlevel1" or "indexlevel2". The attribute is optional but if you intend to use cascading style sheets you must specify at least a "defaultstyle". Use the file defined by the parameter "headerfile" to import the styles. You can choose not to use cascading style sheets (CSS). In that case you use the parameter "levelfont" to control format. See the parameter "levelfont".
levelfont levelfont	defaultfont Arial Bold +2 1 Times Italic -1	This parameter defines which font that will be used to generate the different keywords. If you have defined any stylesheets, this parameter will be left unnoticed. If you have not specified any stylesheets, this parameter must to be used. The syntax [keywordlevel/reservedname][fontname][fontstyle][fontsize]. - [keywordlevel/reservedname] is either the level of the keyword or one of the reserved names, ("defaultfont" or "headlinefont"). - [fontname] is mapped to the "FACE" attribute of the HTML FONT attribute. - [fontstyle] is mapped to the "<B>" or "<I>" tags. Possible values are Bold, Italic and Plain. - [fontsize] is mapped to the "SIZE" attribute of the HTML FONT attribute. The attribute is optional but if you intend to use Font definition you have to specify at least a "defaultfont".

Related topics

Publish Configuration

7.6.2. Change Feedback Addresses

The e-mail address used for feedback in a published configuration can be changed by editing the Feedback URL field in the Select publishing options window. This window is part of the Publish Method Configuration wizard.

The default setting is:

http://www.published_website.com/feedback

This address must be changed before publishing.

A simple e-mail address uses syntax such as:

mailto:xyz@eclipse.org

Additional syntax can be used to embellish the feedback e-mail. Those features are shown in the following table:

Table 8 - Table 1. Feedback Syntax

Feature	Syntax
Send message to multiple recipients	, (comma separating email addresses)
Specify the "Subject" field	?subject= Subject Text
Send message using the "Copy To" or "CC"	?&cc=id@internet.node

Feature	Syntax
field	
Send message using the "Blind Copy To" or "BCC" field	?&bcc=id@internet.node

Example:

Feedback URL: mailto:feedback@your_company.org?subject=Process Configuration&cc=process_eng@your_company.org

Note:

- Use only **one ?** (Question mark) when creating any entry beyond e-mail address. See example above.
- If the subject parameter is not included, it will be automatically generated based on the page with the feedback button.

Related topics

Publish Configuration

8. Categorising Method Content

Contents

8.1. Navigation Views

Categories[152] are used to create navigation views for the final, published Web site. They are created to organise method content and process elements into logical categories that categories can appear in the final, published Web site as navigation views. There are two types of categories: standard and custom. Custom categories is the core element for creating navigation views.

Related topics

Standard Method Categories
Custom Categories
Publish Configurations as Web Sites

8.2. Standard Method Categories

Content categorisation can be based on a set of predefined categories, called standard method categories. Standard method categories are, by definition, linked to specific types of method content. There is a standard category for grouping tasks, called **disciplines** and there is a standard for categorising roles, called **role sets**. There are even two standard categories for grouping work products: domains and work product kinds. Each standard category can only contain the specified type of method content; disciplines can only contain tasks for example.

Table 9 - The five standard categories:

Method Content	Standard Categories	Description
Tasks	**Disciplines**	▪ A discipline is a collection of tasks that are related to a major area of concern within the overall IT environment. ▪ For example, on a software development project, it is common to perform certain requirements tasks in close coordination with analysis and design tasks. Separating these tasks into different disciplines makes the tasks easier to understand. ▪ You can organise disciplines by using discipline groupings.
Work Products	**Domains**	▪ A domain is a logical hierarchy of related work products that are grouped together based on timing, resources, or relationship.

[152] Method content and process elements are organised into logical categories. The categories can appear in your final, published Web site as navigation views. There are two types of categories: standard and custom.

Method Content	Standard Categories	Description
		▪ A domain categorises many work products, but a work product can only belong to one domain. You can divide domains into sub-domains.
	Work Product Kinds	▪ A work product can have many work product kinds. For instance, you might want to have a series of work product kinds that correspond to the overall intent of work products, such as specification, plan, or model.
Roles	**Role Sets**	▪ A role set is used to group roles with certain commonalities. ▪ For example, you can set up a role set named "Analyst" to group together roles such as Business Process Analyst, System Analyst, and Requirements Specifier. Each of these roles work with similar techniques and have overlapping skills, but might be responsible for performing certain tasks and creating certain work products. ▪ Role sets can be organised by using role set groupings.
Tool Mentor (Guidance)	**Tools**	▪ The tools type is a container for tool mentors. A tool mentor is a type of guidance that shows how to use a specific software application to accomplish a piece of work. ▪ Tools can also provide general descriptions of a tool and its general capabilities.

Standard categories provide a means to categorise content according to best practices for creating structured methods.

To use a standard method category in a navigation view:

You will rarely, if ever, use a standard category as the top-level navigation view since a standard category only contains one type of method content. Standard categories can however be used as part of a navigation view based upon a custom category.

1. In the **Library** view of the Authoring Perspective, double-click the configuration in the configurations folder to which you want to add a view.
2. Click the **Views** tab and then click **Add View**. The Select Categories window opens. ***The Select Categories window opens all the standard and custom categories in the configuration***.
3. Select one or more categories to include as a view in your configuration and click **OK**. The view display provides a preview of how the view is displayed in the published tree browser.
4. Select a view to display as the default start-up view in the published Web site, and click **Make Default**. You will normally not make a standard category your default view since it can contain only one type of method content. You will typically select a custom category for your default view.
5. To save the configuration, click **File** → **Save** or close the configuration editor and click **Yes** when prompted to save the changes.

Related topics

Method Content[153]
Method Content Package[154]
Create Method Content
Custom Categories[155]

8.3. Custom Categories

Custom categories are used to compose Navigation Views, thereby providing a means to organise method content for publishing. A Navigation View is a custom category that is designed for publication.

You can categorise content according to any scheme using custom categories. Required content packages and content elements are assigned to a custom category. The custom category can then be added as a view to a method configuration, showing the required content packages and content elements assigned to that custom category.

Custom categories can also be displayed with the elements that they are categorising. For example, you could create a custom category that logically organises content that is related to your development organisation department, such as a "testing" category that groups together all roles, work products, tasks, and guidance elements related to testing.

You can organise custom categories in a hierarchy, which means that you can create a category as a child of another category. Child categories can be referenced by more than one parent category. In the hierarchy, you are building nested custom categories using operations such as Assign or Reassign to influence the structure of that hierarchy.

You can copy a custom category with all its children and assignments. Deep copy is the mechanism used to clone a hierarchy of custom categories.

Custom categories can contain any type of element and can be used to organise content according to any scheme. You can then use custom categories to compose publishable navigation views, which provide a means to organise define the way method content should be presented and read.

Related topics

Authoring Perspective
Configuration View[156]
Method Content Categories[157]
Method Configurations Overview

---

[153] Method content provides step-by-step explanations, describing how specific development goals are achieved, independent of the placement of these steps within a development lifecycle. Processes take these method elements and relate them into semi-ordered sequences that are customized to specific types of projects.

[154] A method content package is a container for method elements. Elements are organized in method packages to structure a large scale of method content and processes and to define a mechanism for reuse.

[155] Method content and process elements are organized into logical categories. The categories can appear in your final, published Web site as navigation views. There are two types of categories: standard and custom.

[156] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

[157] Method content and process elements are organised into logical categories. The categories can appear in your final, published Web site as navigation views. There are two types of category: standard and custom.

8.3.1. Create Custom Categories

Custom categories can be used to compose publishable navigation views and to provide additional information for method and process elements.

To create a custom category:

1. In the Library view of the Authoring Perspective, open the Method Content folder in the method plug-in where you want to create the custom category.
2. To create a top-level category, right-click the Custom Categories folder and click **New → Custom Category**.
3. Enter the name, presentation name, brief description and other descriptive information about the custom category in the **Description** tab.
4. Check the "**Publish this category with the categorised method elements**" box if you want to see the category presentation name to be listed on each page of the categorised element.
5. Under the **Assign** tab, select the items to include in the custom category. Select individual method content elements, process elements, standard categories, or other custom categories. In some editors such as the role editor, the custom category is also available to add to individual elements on the Categories tab.
6. After you assign content to the custom category, you can modify the sequence of the items in the category by setting the order manually, in alphabetical order, reverse alphabetical order, or by element type. This affects the sequence of the items in the **Library** view, the Configuration View[158], and in a published Web site if the custom category is included as a view in a configuration.
   a. To manually sort the elements in a category, select **Manual** from the **Sort Type** box. Click **Order** and specify the order by using the **Up** and **Down** buttons.
   b. To automatically sort the elements in a category choose an automatic order from the Sort Type combo box.

      Tip: You can also select an automatic order from the Sort Type combo box when you click Order.
7. To save your new custom category, click **File → Save** or close the editor and click **Yes** when prompted to save the changes.

You can now use the custom category as a view in a published configuration.

To use the custom category as a navigation view:

8. In the **Library** view of the Authoring Perspective, double-click the configuration in the configurations folder to which you want to add a view.
9. Click the **Views** tab and then click **Add View**. The Select Categories window opens. ***The Select Categories window opens all the standard and custom categories in the configuration***.

[158] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

10. Select one or more categories to include as a view in your configuration and click **OK**. The view display provides a preview of how the view is displayed in the published tree browser.
11. Select a view to display as the default start-up view in the published Web site, and click **Make Default**.
12. To save the configuration, click **File** → **Save** or close the configuration editor and click **Yes** when prompted to save the changes.

Related topics

Authoring Perspective
Configuration View[159]
Method Content Categories
Custom Categories[160]
Method Configurations Overview

8.3.2. Modify Custom Categories

To edit a custom category:

1. In the **Library** view of the Authoring Perspective, right-click the custom category that you want to edit and select **Edit**. The element opens in the method content editor.

   Tip: You can also access the method element editor by double-clicking the custom category you want to edit.
2. Edit the content of the custom category, When finished, you can modify the sequence of the items in the category. This affects the sequence of the items in the **Library** View, Configuration View, and in a published Web site if the custom category is included as a view in a configuration. Use the **Sort Type Box** to specify the sequence.
3. To save your custom category, click **File** → **Save** or close the editor and click **Yes** when prompted to save the changes.

To delete a custom category:

1. Right-click the custom category that you want to delete and select **Delete** and click **OK**.
2. Click **OK** to confirm removal of the customer category.

To rename a custom category:

1. Right-click the custom category that you want to rename and select **Rename**. The Rename dialog is displayed.

[159] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

[160] Method content and process elements are organized into logical categories. The categories can appear in your final, published Web site as navigation views. There are two types of categories: standard and custom.

2. In the **New name** text field, enter a name for the custom category and click OK. The custom category is displayed in the Library view with the new name.

   Tip: You can also rename a custom category using the method element editor; double-click the custom category and enter the new name in the Name text field.

Related topics

Authoring Perspective
Configuration View[161]
Method Content Categories
Custom Categories[162]
Method Configurations Overview

8.3.3. Nested Custom Categories

You can organise custom categories in a hierarchy, which means that you can create a category as a child of another category. Child categories can be referenced by more than one parent category. In the hierarchy, you are building nested custom categories that can use operations such as **Assign** or **Reassign** to influence the structure of that hierarchy.

To change the assignment of custom categories in a hierarchy of categorised elements (For example, to reassign a sub-category to a new or additional parent category), you can use either the **Library** view or the method element editor.

To change the assignment of custom categories:

1. To change assignments by using the **Library view**:
   a. To **assign** a category to an **additional parent**:
      i. Right-click the custom category and select **Assign**. The assign dialog opens.
      ii. From the tree view, choose the categorised element destination and click **OK**.
   b. To **reassign** a category to a **new parent category** or the top-level categories folder:
      i. Right-click the custom category and select **Reassign**. The Reassign dialog opens.
      ii. From the tree view, choose the categorised element destination and click **OK**.
   c. To **unassign** a category that is a sub-category of more than one parent category:
      i. Right-click the custom category and select **Unassign**. If this category is its last occurrence in the method plug-in, the **Remove** dialog will open and warn you that the entire category will be deleted.

[161] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

[162] Method content and process elements are organised into logical categories. The categories can appear in your final, published Web site as navigation views. There are two types of categories: standard and custom.

ii. Select **Yes** to confirm. If other parent categories still reference the category then the category is removed without confirmation.

2. (Optional) To change assignments by using the **method element editor**:
   a. Open the custom category element to which you want to assign new children to in the Method element editor and select the **Assign** tab.
   b. Click **Assign**. The **Select** window is displayed.
   c. From the tree view, select the method element that you want to assign to the custom category and click **OK**.
   d. Click **Unassign** to remove elements from the category.
   e. To save your changes, click **File** → **Save** or close the editor and click **Yes** when prompted to save the changes.

Related topics

Custom Categories
Method Content Categories

8.3.4. Deep Copy Custom Categories

Deep copy is the mechanism that you can use to copy a custom category with all its children and assignments. It is used to clone a hierarchy of custom categories.

To deep copy a custom category:

1. In the Library View[163] of the Authoring Perspective, right-click the custom category that you want to clone and select **Deep Copy**. The Deep copy dialog is displayed.
2. From the tree view, select a destination for the selected elements and click **OK**. The custom category, all of its children, and its assignment are copied and now displayed in the Library view.

Related topics

Custom Categories
Method Content Categories

8.4. Assign Categories to Content Elements

There are no help files for this.

Standard and custom categories are assigned to method content elements. You can assign categories, either from the category point of view or from the method content point of view. In other words, you can edit the category and add the relevant content elements or you can edit the method content elements and add the relevant categories.

Related topics

Method Content Categories

[163] The Library view displays the method content that is available in the current method library, which is organized into sets of method plug-ins and configurations.

8.4.1. Assign Category

To assign a custom category:

3. Make sure that you are in the **Authoring** perspective.
4. Expand the folder (node) in the **Library** view panel for relevant method plug-in and expand the **Method Content** and **Custom Categories** folder.
5. Double-click the custom category that you want to assign to method content elements or right-click the same category and select Edit. The custom category editor opens.
6. Click the **Assign** tab and click the **Assign** button. The **Select Dialog: Most common** window opens.
7. If you have previously created content, add your **Roles**, **Artefacts**, **Tasks**, and **Guidance** into the custom category.

   Remember: You can select multiple items by using Shift or Control key.
8. Click **OK**.

To assign a standard category:

1. Make sure that you are in the **Authoring** perspective.
2. Expand the folder (node) in the **Library** view panel for relevant method plug-in and expand the **Method Content**, **Standard Categories** and the relevant standard category type folder.
3. Double-click the standard category that you want to assign to method content elements or right-click the same category and select Edit. The custom category editor opens.
4. Click the tab related to the type of method content that the standard category can contain, i.e. Task tab for the Discipline category, Work Products tab for the Domains category and for the Work Product Kinds category and so forth.
5. Click the **Assign** button to the right. The **Select Dialog: Method content (tasks, work products etc)** window opens.
6. If you have previously created content, add your **Roles**, **Work Products (Artefacts, Deliverables, Outcomes)**, **Tasks**, and **Guidance** into the standard category.

   Remember: You can select multiple items by using Shift or Control key.
7. Click **OK**.

Related topics

8.4.2. Modify Category Assignment

The editor of each method content element has a Categories tab from which you can Assign, Unassign or Order the relevant categories. The correct standard category section(s) is presented in the tab, together with the custom category section.

To modify a category assignment:

8. Make sure that you are in the **Authoring** perspective.
9. Expand the folder (node) in the **Library** view panel for relevant method plug-in and expand the **Method Content** and **Content Packages** folder.
10. Expand the content package that contains the method content element you want to edit and double-click the element or right click it and select Edit. The content element editor opens.
11. Click the **Categories** tab and click the **Add** button. The **Select Dialog** with the relevant Category Type opens.
12. If you have previously created content, add your relevant categories.

    Remember: You can select multiple items by using Shift or Control key.
13. Click **OK**.

Related topics

Assign Category
Category Variability

8.5. Category Variability

The help files are strangely silent about this.

Both standard and custom categories have variability types assigned to them. Since all tutorials and help files are mum about this, I will have to experiment to verify the effect. I suspect that variability affects the method content elements that the categories are assigned, not the categories themselves. Since categories affect configurations and variability is used to customise configurations, the link is there, somehow.

Variability Type

Variability type describes how one element affects another through variability associations. The five types of variability associations are:

Variability Type	Association Description
Not Applicable	▪ The element is a base element and does not affect another element through variability. This is the default value of an element's variability type.
Contributes	▪ A contributing element adds to the base element. Contributes provides a way for elements to contribute with their properties to their base element without directly changing any of its existing properties. ▪ The base appears in the published Web site but the contributing element does not. In and out relationships from the contributing element are added to the base. Text from the contributing element is appended to corresponding base

Variability Type	Association Description
	sections. ▪ When the contribution is resolved during publication or by using the Browsing perspective, the base element is logically replaced with an element that combines the attributes and associations of the contributing element with the base element.
Replaces	▪ A replacing element replaces parts of the base element. It provides a mechanism for an element to replace a base element without directly changing any of the base element's existing properties. ▪ The replacer appears in the published Web site but the base element does not. “Out Relationships” in the replacer are left untouched, and those of the base element are ignored. “In Relationships” from the base are added to the replacer. Text in the replacer is left untouched, and the base element's text is ignored.
Extends	▪ An extending element inherits characteristics of the base element. Both the extender and the base element appear in the published Web site. ▪ Out relationships from the base are added to the extender. In relationships in the extender are left untouched, the base element's are ignored. Text is added from the base if the extender has no value defined for the given section. ▪ Provides a mechanism for method plug-ins to reuse elements from a base plug-in through a kind of inheritance. Attribute values and associations are inherited from the "based-on" element to the extending element. The result is that the extending element has the same properties as the "based-on" element, but might define its own additions.
Extends and Replaces	▪ This variability relationship combines the effects of extends and replace variability into one variability type. ▪ Whereas the replaces variability completely replaces all attributes and outgoing association instances of the base variability element with new values and instances, or removes all values or association instances if the replacing element does not define any, **extends and replaces variability only replaces the values that have been redefined and leaves all other values of the base element as is**.

Related topics

Method Content Variability
Associations Impacted by Variability
Method Content Categories
Custom Categories
Assign Category
Modify Category Assignment
Activity Variability
Create Capability Patterns
Capability Patterns Reuse

9. Create Processes

Contents

9.1. Create Capability Patterns

A Capability Pattern is a special process that describes a reusable cluster of activities in a general and common process area that provides a consistent development approach to common problems. Capability patterns articulate process knowledge for a key area of interest, such as a discipline, and can be used by process practitioners, either directly or as a guide.

A capability pattern does not relate to any specific phase or iteration of a development lifecycle, and should not imply any. In other words, a pattern should be designed in a way that it is applicable anywhere in a delivery process, thereby enabling its activities to be flexibly assigned to whatever phases there are in the delivery process to which it is being applied.

Capability patterns can be used as building blocks to assemble delivery processes or larger capability patterns. You do not need to develop your delivery process from scratch; you can reuse existing capability patterns or even capability pattern parts. The reuse can consist of copying or extending already existing capability patterns.

Typically, but not necessarily, capability patterns have the scope of one discipline, providing a breakdown of reusable complex activities, their relationships to the roles that perform the tasks within these activities, in addition to the work products that are used and produced. Examples of capability patterns include use-case based requirements management, use case analysis, or unit testing.

Before creating a capability pattern, do the following:

- Select a Method Plug-in[164] to hold your process.
- Find or create a Process Package[165].

A capability pattern does not relate to any specific phase or iteration of a development lifecycle, and should not imply any. In other words, a capability pattern should be designed in a way so that it is applicable anywhere in a delivery process. This enables the pattern's activities to be flexibly assigned to whatever phases there are in the delivery process to which it is being applied.

There are several ways to populate a process with method elements:

- By including already defined capability patterns.
- By including parts of already defined capability patterns.
- By incorporating individual method elements by dragging the elements onto an activity in the process.
- By creating descriptors directly in the process, which are either unrelated to any method content or related to method content at a later point in time.

To create a capability pattern:

1. Make sure that you are in the **Authoring** perspective.
2. Expand the folder (node) in the **Library** view panel for **Processes** and right-click **Capability Patterns**, select **New → Process Package** and specify a name for the Process Package.

3. After you have selected or created a Process Package, right-click Process Package and click **New → Capability Pattern**. The **New Process Component** window opens.
4. In the **Name** field, type a name for the capability pattern, and select a default configuration from the drop-down list.

   Your process can contain content from many different method plug-ins, therefore, you need to define a configuration that defines the visible set of elements and relationships when the process is authored. This process authoring configuration is referred to as the default configuration for the process and should define the set

[164] All content is organised in method plug-ins. With method plug-ins and method packages, you can organise your content at a level of granularity that meets your needs for authoring and reusing content.

[165] Process Packages provide visual groupings of processes.

of method plug-ins, content packages, and other processes from the method library that will be referred to by the process at some point.

5. Click **OK**. The process is created and the process editor is opened.
6. In the process editor, under the **Description** tab, document your process using the available text fields.

   Note: You can use the Rich Text Editor[166] to edit or enter the text for any field that has the rich text editor icon  . Click the icon to access the rich text editor. Click the icon again to close the editor.
7. Decide on your **primary process authoring view**. You develop a process using any of three different views:
   - **Work Breakdown Structure**: Define a work breakdown structure in your process. Create iterations and activities first, and populate your activities by applying tasks from your method content. For more information about defining a work breakdown structure, see Develop Work Breakdown Structures[167].
   - **Team Allocation**: Define which teams and roles will participate in activities and find responsible work products and tasks from there. For more information about teams and roles, see Develop Team Allocation Structures[168].
   - **Work Product Usage**: Define which work products should be created in activities and find tasks and roles from there. For more information about work products, see Develop Work Product Usage Structures[169].

Related topics

Method Plug-in[170]
Capability Pattern Reuse
Copy Capability Patterns
Deep Copy Capability Patterns
Extend Capability Patterns
Develop Work Breakdown Structures
Develop Team Allocation Structures
Develop Work Product Usage Structures

---

[166] The rich text editor provides text formatting functions that alter the appearance of published content. The CSS style sheet in a Web site controls the appearance of text at a global level, but the rich text editor can override those controls for specific text elements.

[167] A work breakdown structure is a hierarchical breakdown of work, such as activities, tasks, and steps, defining a process.

[168] In the Team Allocation view, you can create a process by defining which roles participate in activities and find responsible work products and tasks from there. You can also review the roles in a process that has been created by adding tasks or work products to the process.

[169] In the Work Product Usage view, you can create a process by defining which work products will be created and used in the process, and then finding responsible roles and tasks from there. You can also review the work products in a process that has been created by adding tasks or roles to the process.

[170] All content is organised in method plug-ins. With method plug-ins and method packages, you can organise your content at a level of granularity that meets your needs for authoring and reusing content.

9.2. Create Delivery Processes

A delivery process describes a complete and integrated approach for performing a specific type of project and reflects the project's specific planning and project management needs.

A delivery process describes what is produced, how it is produced and the staffing required for the entire project lifecycle.

Before creating a delivery process:

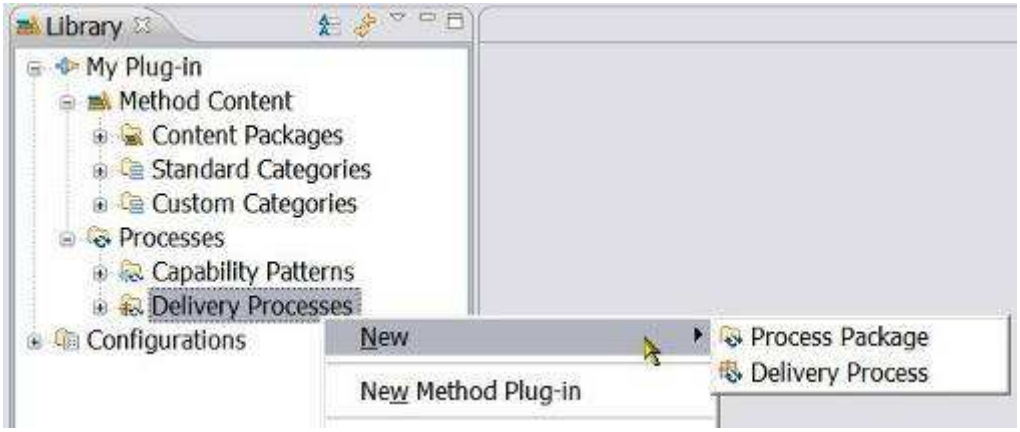
- Select a Method Plug-in[171] to hold your process.
- Find or create a Process Package[172].

There are several ways to populate a process with method elements:

- By including already defined capability patterns.
- By including parts of already defined capability patterns.
- By incorporating individual method elements by dragging the elements onto an activity in the process.
- By creating descriptors directly in the process, which are either unrelated to any method content or related to method content at a later point in time.

To create a delivery process:

8. Make sure that you are in the **Authoring** perspective.
9. Expand the folder (node) in the **Library** view panel for **Processes** and right-click **Capability Patterns**, select **New → Process Package** and specify a name for the Process Package.



10. After you have selected or created a process package, right-click it and then Click **New → Delivery Process**.
11. In the Name field, type a name for the delivery process and then select a default configuration from the drop-down list.

[171] All content is organised in method plug-ins. With method plug-ins and method packages, you can organise your content at a level of granularity that meets your needs for authoring and reusing content.

[172] Process Packages provide visual groupings of processes.

Your process can contain content from many different method plug-ins, therefore, you need to define a default configuration that defines the visible set of elements and relationships when the process is authored. This process authoring configuration is referred to as the default configuration for the process and should define the set of method plug-ins, content packages, and other processes from the method library that will be referred to by the process at some point.

12. Click **OK**. The process is created and the process editor opens.
13. In the process editor under the **Description** tab, document your process using the available text fields.

    Tip: You can use the Rich Text Editor[173] to edit or enter the text for any field that has the rich text editor icon  . Click the symbol to access the rich text editor. Click the icon again to close the rich text editor.
14. Decide on your **primary process authoring view**. You develop a process using three different views:
    - **Work Breakdown Structure**: Define a work breakdown structure in your process. Create iterations and activities first, and populate your activities by applying tasks from your method content. For more information about defining a work breakdown structure, see Develop a Work Breakdown Structure[174].
    - **Team Allocation**: Define which teams and roles will participate in activities and find responsible work products and tasks from there. For more information about teams and roles, see Develop a Team Allocation Structure[175].
    - **Work Product Usage**: Define which work products should be created in activities and find tasks and roles from there. For more information about work products, see Develop a Work Product Usage Structure[176].
15. Apply a Capability Pattern or capability pattern parts to the delivery process.

Process Packages can be created both under Capability Patterns and under Delivery Processes. New Capability Patterns can be created both directly under Capability Patterns and under any Process Package. New Delivery Processes can be created both directly under Delivery Processes and under any Process Package. This indicates that Process Packages only operate as a visual grouping tool.

[173] The rich text editor provides text-formatting functions that alter the appearance of published content. The CSS style sheet in a Web site controls the appearance of text at a global level, but the rich text editor can override those controls for specific text elements.

[174] A work breakdown structure is a hierarchical breakdown of work, such as activities, tasks, and steps, defining a process.

[175] In the Team Allocation view, you can create a process by defining which roles participate in activities and find responsible work products and tasks from there. You can also review the roles in a process that has been created by adding tasks or work products to the process.

[176] In the Work Product Usage view, you can create a process by defining which work products will be created and used in the process, and then finding responsible roles and tasks from there. You can also review the work products in a process that has been created by adding tasks or roles to the process.

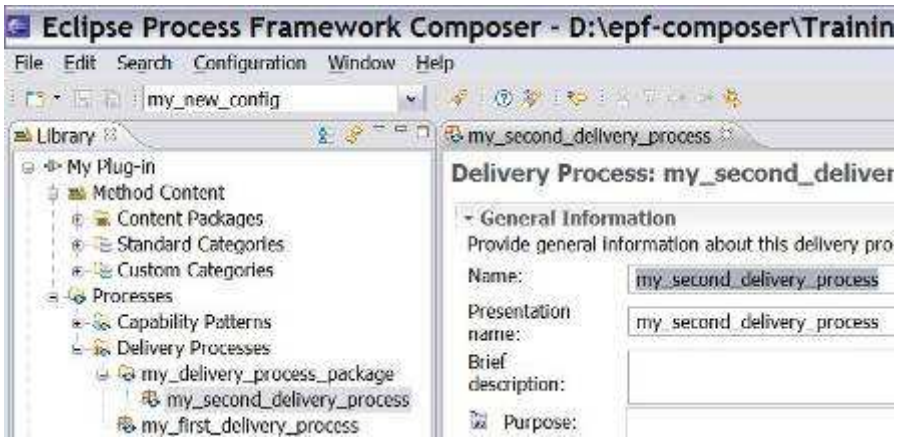
Related topics

Method Plug-in[177]
Process Management[178]
Create Capability Patterns
Capability Patterns Reuse
Create Delivery Processes
Develop Work Breakdown Structures
Develop Team Allocation Structures
Develop Work Product Usage Structures
Process Element Properties View[179]
Activity Variability Types
Rich Text Editor[180]

9.3. Develop Work Breakdown Structures

A work breakdown structure is a hierarchical breakdown of work, such as activities, tasks, and steps, defining a process. Iterations, Phases and Activities are Activity Types and each type can be transformed into another type using the Properties View.

Before creating a work breakdown structure, be sure that the configuration selected in the tool bar is the same as the configuration selected as the default configuration for your process.



[177] All content is organised in method plug-ins. With method plug-ins and method packages, you can organise your content at a level of granularity that meets your needs for authoring and reusing content.

[178] A process describes how a particular piece of work should be done. The work may have a relatively small scope, in which case it can be described as a capability pattern, or may address a full project lifecycle, in which case it can be described as a delivery process. A process can reuse method elements and combines them into a structure and sequence for carrying out work.

[179] While editing a process, you can use the Process Element Properties View to edit all details for a single element in the process. If you click in any row in a process display, you will see the full details of the process element in the row in the Properties View.

[180] The rich text editor provides text formatting functions that alter the appearance of published content. The CSS style sheet in a Web site controls the appearance of text at a global level, but the rich text editor can override those controls for specific text elements.

To develop a work breakdown structure:

1. Access the 'work breakdown structure' editor by clicking the **Work Breakdown Structure** tab in a process editor.

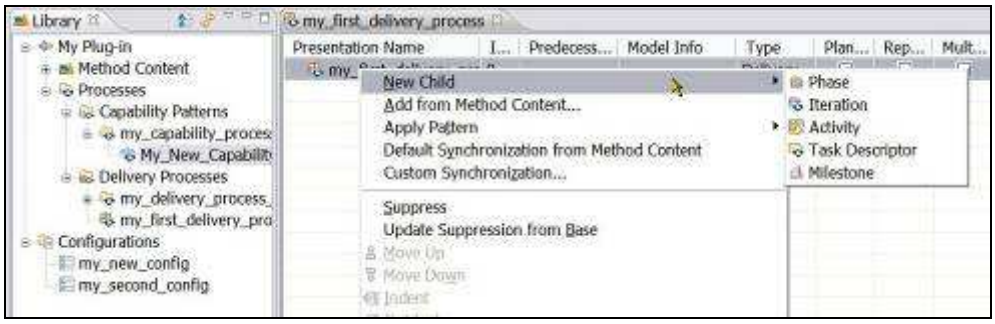
   Note: A breakdown structure is created with breakdown elements. Examples of breakdown elements are Iteration, Phase, Activity, or Task Descriptor.

   Iteration, Phase, Activity are activity types and each type can be transformed into another type. Typically, a process is created by defining its iterations and phases which are then further broken down into levels of activities. Finally, you can populate a work breakdown structure's activity with task descriptors.

   The structure is not verified or imposed by application; it is up the process designer to choose the appropriate process element type (Phase, Iteration, Activity, Task Descriptor or Milestone) and to insert it at the appropriate place in the process hierarchy.

2. Right-click the element (either a Capability Pattern or Delivery Process) to which you want to add structure, and click **New Child → Activity** to create the new activity. If needed, create more activities to set up your breakdown structure.

   Activities (in addition to phases and iterations) **can be nested inside each other** according to how each relates in the hierarchy.



3. Right-click the activity and click **Show Properties View**. Complete the information under the **Documentation** tab for the activity. This information appears in the published Web page for the activity.
4. Review the list of **tasks** in the **Configuration view** to see which tasks are available.
5. Select a task to add to the breakdown structure and then ***drag it on top*** of the activity to which you want it to belong. The task is added as a task descriptor to that activity.
6. If the **Properties** view for the task is not displayed, select the task in the work breakdown structure editor, right-click, and click **Show Properties View**. Click the **Documentation** tab and complete the information required under this tab. This information appears in the published Web page for the activity.

   Use the **tabs** on the side of the Properties view to review different aspects of the task descriptor. In the Properties view, you can perform individual modifications of the task descriptor, such as change the presentation name, add textual descriptions, and change performing roles, among others.

   ;When changing the task descriptor's relationships under the Roles or Work Products tabs, you can add new elements from your method content by clicking

Add, or connect your task descriptor with tasks already present in this activity. For more information, see Process Element Properties View[181].

7. Continue adding tasks to the activity or activities.

You can preview what your process will look like in a published Web site at any time by switching to the Browsing Perspective and then selecting the process, or an activity in the process, in the Configuration View[182]. You can use the links on the page to navigate through the process. Switch back to the Authoring Perspective to continue editing your process.

Related topics

Process Management[183]
Create Capability Patterns
Capability Patterns Reuse
Create Delivery Processes
Develop Team Allocation Structures
Develop Work Product Usage Structures
Process Element Properties View[184]
Activity Variability Types
Rich Text Editor[185]

9.4. Develop Team Allocation Structures

In the Team Allocation view, you can create a process by defining which roles participate in activities and find responsible work products and tasks from there. You can also review the roles in a process that has been created by adding tasks or work products to the process.

Before you create a team allocation structure, ensure the configuration selected in the tool bar is the same as the configuration that you selected as the default configuration for your process.

To develop a team allocation structure:

1. In the process editor, click the **Team Allocation** tab.

[181] While editing a process, you can use the Process Element Properties View to edit all details for a single element in the process. If you click in any row in a process display, you will see the full details of the process element in the row in the Properties View.

[182] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

[183] A process describes how a particular piece of work should be done. The work may have a relatively small scope, in which case it can be described as a capability pattern, or may address a full project lifecycle, in which case it can be described as a delivery process. A process can reuse method elements and combines them into a structure and sequence for carrying out work.

[184] While editing a process, you can use the Process Element Properties View to edit all details for a single element in the process. If you click in any row in a process display, you will see the full details of the process element in the row in the Properties View.

[185] The rich text editor provides text formatting functions that alter the appearance of published content. The CSS style sheet in a Web site controls the appearance of text at a global level, but the rich text editor can override those controls for specific text elements.

Remember: For the purposes of this topic, the breakdown element example used is Activity. Other breakdown elements include (but are not limited to) phase and iteration. The element that you use depends on the scope of the process that you create.

2. Right-click the element (either a Capability Pattern or Delivery Process) to which you want to add structure, and click **New Child → Activity** to create a new activity. If needed, create more activities to set up your breakdown structure.

   Activities (in addition to phases and iterations) can be nested inside each other according to how each relates in the hierarchy. You can **add roles directly** to your activities.

3. In **the Configuration** view, review the list of roles. In this view, tasks are **sorted by role sets.**
4. Drill into the role sets hierarchy to see which roles are available in this configuration.
5. Select a role and drag it on top of the appropriate activity. The role is added as a role descriptor. If the role is responsible for work products in the configuration's method content, a wizard prompts you to add work products.
6. Select one or more work products and click **OK**. For each selected work product, the next wizard prompts you to select tasks that produce these work products. Again, select one or more tasks and then click **OK** to add these elements to your process.
7. Review the role descriptor's details in its **Properties** view. If the Properties view is not displayed, in the work breakdown structure editor, right-click the role and then select Show Properties View. Use the tabs on the side of the Properties view to review different aspects of the role descriptor. In the Properties view, you can also perform individual modifications of the role descriptor, such as change the presentation name; add textual descriptions, change work products the role is responsible for, and more.
8. Continue adding roles to your activities.

You can preview what your process will look like in a published Web site at any time by switching to the Browsing Perspective and then selecting the process, or an activity in the process, in the Configuration View[186]. You can use the links on the page to navigate through the process. Switch back to the Authoring Perspective to continue editing your process.

Related topics

Process Management[187]
Create Capability Patterns
Capability Patterns Reuse

[186] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

[187] A process describes how a particular piece of work should be done. The work may have a relatively small scope, in which case it can be described as a capability pattern, or may address a full project lifecycle, in which case it can be described as a delivery process. A process can reuse method elements and combines them into a structure and sequence for carrying out work.

Create Delivery Processes
Develop Work Breakdown Structures
Develop Work Product Usage Structures
Process Element Properties View[188]
Activity Variability Types
Rich Text Editor[189]

9.5. Develop Work Product Usage Structures

In the Work Product Usage view, you can create a process by defining which work products will be created and used in the process, and then finding responsible roles and tasks from there. You can also review the work products in a process that has been created by adding tasks or roles to the process.

Before you create a work product usage structure, ensure that the configuration selected in the tool bar is the same as the configuration that you selected as the default configuration for your process.

To develop a work product usage structure:

1. To access the work product usage editor, in the process editor, click the **Work Product Usage** tab.

   Note: For the purposes of this topic, the breakdown element example used is Activity. Other breakdown elements include (but are not limited to) Phase and Iteration. The element that you use depends on the scope of the process that you create.

2. Right-click the element (either a Capability Pattern or Delivery Process) to which you want to add structure, and click **New Child** → **Activity** to create a new activity. If needed, create more activities to set up your breakdown structure. Activities (in addition to phases and iterations) can be nested inside each other according to how each relates in the hierarchy.
3. Review the list of work products in the Configuration View[190]. In this view, work products are sorted by domain and work product kinds. Drill into either of these hierarchies to see which work products are available in this configuration.
4. Select a work product to add to the activity and then drag it on top of the activity to which you want it to belong. The work product is added as a work product descriptor to that activity. If the work product is an output to one or more tasks in the configuration, a wizard opens prompting you to add the task.

   Remember: It is not required to add a task. A valid process can contain just roles and work products.

---

[188] While editing a process, you can use the Process Element Properties View to edit all details for a single element in the process. If you click in any row in a process display, you will see the full details of the process element in the row in the Properties View.

[189] The rich text editor provides text formatting functions that alter the appearance of published content. The CSS style sheet in a Web site controls the appearance of text at a global level, but the rich text editor can override those controls for specific text elements.

[190] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

5. Specify the entry and exit states of the work product for the activity to which it was added.
6. Review the new work product descriptor details in the Properties view. If the Properties view is not displayed, right-click the work product descriptor in the process editor, and then click Show Properties View. Use the tabs on the side of the Properties view to review different aspects of the work product descriptor. In the Properties view, you can also perform individual modifications of the work product descriptor, such as change the presentation name, add textual descriptions, add entry and exit states, and more.
7. Continue adding work products to your activities.

Related topics

Process Management[191]
Create Capability Patterns
Capability Patterns Reuse
Create Delivery Processes
Develop Work Breakdown Structures
Develop Team Allocation Structures
Process Element Properties View[192]
Activity Variability Types
Rich Text Editor[193]

9.6. Activity Variability

Activity variability is a mechanism for defining a new process, iteration, phase or activity (Iteration, phase and activity are three types of activity and a process is a sequence of these activities) by referencing an existing base activity and then specifying the differences between them. Process authors can use this mechanism to reuse and customise previously created activities in a new context. It also provides a way to make changes to activities in a locked method plug-in which otherwise could not be modified. Activity variability is available for processes, iterations, phases or activities.

Activity variability is similar to Method Content Variability[194]. There are three ways to use activity variability to customise an existing activity:

- Contributes Variability[195]

[191] A process describes how a particular piece of work should be done. The work may have a relatively small scope, in which case it can be described as a capability pattern, or may address a full project lifecycle, in which case it can be described as a delivery process. A process can reuse method elements and combines them into a structure and sequence for carrying out work.

[192] While editing a process, you can use the Process Element Properties View to edit all details for a single element in the process. If you click in any row in a process display, you will see the full details of the process element in the row in the Properties View.

[193] The rich text editor provides text formatting functions that alter the appearance of published content. The CSS style sheet in a Web site controls the appearance of text at a global level, but the rich text editor can override those controls for specific text elements.

[194] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

- Extends Variability[196]
- Replaces Variability[197]

With a **Contributes** variability relationship type, the customisation can introduce changes relevant to the new target activity, without affecting the base activity. The base activity is logically replaced with an activity that combines the contributing activity with the base activity.

With an **Extends** variability relationship type, the customisation can reuse a base activity through a kind of inheritance to which it adds its own activity elements. The result is that the extending activity has the same properties as the "based-on" activity, but might define its own additions.

To modify or enhance an extended activity, its status will have to be changed, either to “Local Contribution” to add supplementary local tasks, to “Local Replacement” to replace the tasks or to “Local Replacement and Deep Copy” to create a copy of all linked elements.

With a **Replaces** variability relationship type, the association provides a mechanism for an activity element to replace a base activity element, without directly changing any of the base element's existing properties.

To apply activity variability:

1. Make sure that you are in the **Authoring** perspective.
2. Expand the folder (node) in the **Library** view for **Processes** and the **Delivery Processes** or the **Capability Pattern** folder (node) and select the process that will be linked to other activities through a variability relationship[198].
3. Open the process editor by double-clicking the process or by right-clicking it and selecting **Edit** from the pop-up menu.
4. Select your process authoring view, by clicking the **Work Breakdown Structure**, **Team Allocation** or the **Work Product Usage** tab.
5. Select the activity that you want to reference another already existing activity, using the variability mechanism.
6. In the Properties view, under the **General** tab, select the Variability drop-down list and select **Contributes**, **Extends** or **Replaces** or leave the default ‘**Not applicable**’.

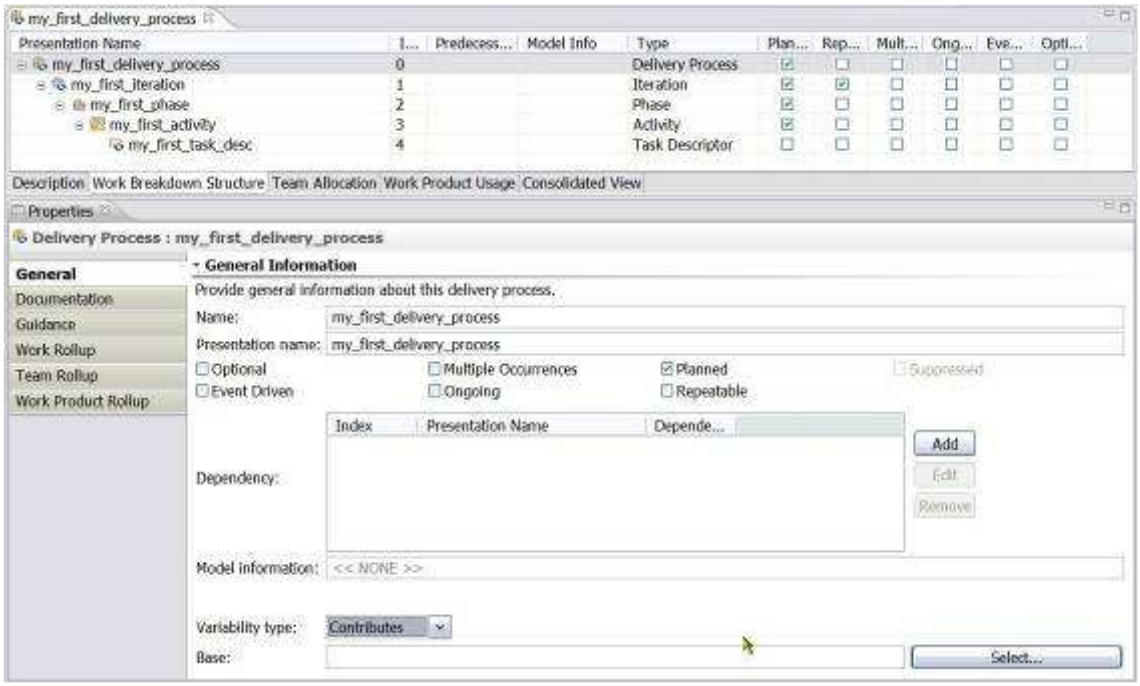
---

[195] A contributing element adds to the base element. Contributes provides a way for elements to contribute their properties into their base element without directly changing any of its existing properties, such as in an additive fashion. When the contribution is resolved during publication or by using the Browsing perspective, the base element is logically replaced with an element that combines the attributes and associations of the contributing element with the base element.

[196] Extends associations provide a mechanism for method plug-ins to reuse elements from a base plug-in through a kind of inheritance. Attribute values and associations are inherited from the "based-on" element to the extending element. The result is that the extending element has the same properties as the "based-on" element, but might define its own additions.

[197] The Replaces variability association provides a mechanism for an element to replace a base element without directly changing any of the base element's existing properties.

[198] You can select the process and open the process editor from both the Library view and the Configuration view when the default configuration for the process has been selected.



7. Select the referenced base activity by clicking **Select**. The **Select Dialog: Activities** pops up and you can select the base activity with which you want to create a variability relationship.

Your new process, iteration, phase or activity will now reuse the previously created activities in a new context.

Related topics

Process Management[199]
Method Content Variability
Category Variability
Create Capability Patterns
Capability Patterns Reuse
Develop Work Breakdown Structures
Develop Team Allocation Structures
Develop Work Product Usage Structures
Process Element Properties View[200]

9.7. Capability Patterns Reuse

Capability patterns are a special type of process that describes a reusable cluster of activities in common process areas. A capability pattern is applied against processes and the three activity types (iteration, phase and activity), either through copying or extending already existing capability patterns.

[199] A process describes how a particular piece of work should be done. The work may have a relatively small scope, in which case it can be described as a capability pattern, or may address a full project lifecycle, in which case it can be described as a delivery process. A process can reuse method elements and combines them into a structure and sequence for carrying out work.

[200] While editing a process, you can use the Process Element Properties View to edit all details for a single element in the process. If you click in any row in a process display, you will see the full details of the process element in the row in the Properties View.

With the mechanism of capability patterns, it is not necessary to develop a process from scratch, adding descriptors one by one. You can use existing capability patterns as building blocks to assemble a delivery process or larger capability patterns and customise the pattern's content to the particular situation to which it is applied. This ensures optimal reuse and application of the key practices they express.

With capability patterns you can decompose complex set activities into patterns with the added benefit of improved complexity management. Change is more localised, consistency is improved and productivity is increased.

The following chapters describe how processes, either delivery processes or capability patterns, can achieve reuse, either through copying or extending already existing capability patterns or parts of patterns.

Related topics

Process Management[201]
Method Content Variability
Category Variability
Activity Variability
Create Capability Patterns
Capability Patterns Reuse
Copy Capability Patterns
Deep Copy Capability Patterns
Extend Capability Patterns
Local Contribution
Local Replacement
Local Replacement and Deep Copy

9.7.1. Copy Capability Patterns

When an activity or Capability Pattern is copied into a target process or activity, its method content elements are disconnected from the original capability pattern. The copied capability pattern elements can be altered, deleted or changed, as you want, in the process editor without affecting the original method content elements. Method elements in a copied capability pattern appear in black.

Iteration, phase and activity are three types of activity and a process is a sequence of these activities and the copying capability pattern procedure can be used to **copy any activity**, not only capability patterns.

Copy creates a **Not Applicable** variability relationship type to the original activity.

To copy a capability pattern:

1. Make sure that you are in the **Authoring** perspective.
2. Expand the folder (node) in the **Library** view for **Processes** and the **Delivery Processes** or the **Capability Pattern** folder (node) and select the process that will be linked to other activities through a variability relationship[202].

[201] A process describes how a particular piece of work should be done. The work may have a relatively small scope, in which case it can be described as a capability pattern, or may address a full project lifecycle, in which case it can be described as a delivery process. A process can reuse method elements and combines them into a structure and sequence for carrying out work.

3. Open the process editor by double-clicking the process or by right-clicking it and selecting **Edit** from the pop-up menu.
4. Select your process authoring view, by clicking the **Work Breakdown Structure**, **Team Allocation** or the **Work Product Usage** tab.
5. In the Configuration View[203], expand the **Processes** and **Capability Patterns** folders (node) and select a capability pattern or its parts that you want to copy into an activity in a target process. *Note that you select and drag the parts you want to copy from the **Configuration** view and not the Library view.*
6. With the mouse, drag the capability pattern or parts into the activity (process, iteration, phase or activity) where you want the copy to go. To select multiple activities in a capability pattern, press the **Ctrl** or **Shift** keys while you select.
7. In the menu that opens, click **Copy**. The capability pattern is copied and its name is displayed in black and it is available for you to edit.

Alternative procedure:

1. In one of the process views, right-click an activity, and click **Apply Pattern** → **Copy**. The **Select Dialog: Processes** window opens.
2. Drill down into the process tree until you locate the capability pattern that you want to copy.
3. Select the pattern, and click **OK**. The Select Dialog: Processes window closes and the capability pattern is applied.

Suppress

If you do not want a specific activity or task in the capability pattern to be included in your process, you could "suppress" it. The feature is not very useful when copying a capability pattern, since the activity or the task can be deleted from the copy, which is more natural than suppressing it. See Extending Capability Patterns for more details.

Related topics

Activity Variability
Create Capability Patterns
Capability Patterns Reuse
Deep Copy Capability Patterns
Extend Capability Patterns
Develop Work Breakdown Structures
Develop Team Allocation Structures
Develop Work Product Usage Structures
Process Element Properties View[204]

---

[202] You can select the process and open the process editor from both the Library view and the Configuration view when the default configuration for the process has been selected.

[203] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

[204] While editing a process, you can use the Process Element Properties View to edit all details for a single element in the process. If you click in any row in a process display, you will see the full details of the process element in the row in the Properties View.

9.7.2. Deep Copy Capability Patterns

Deep copy is the mechanism that you can use to copy an activity or a Capability Pattern to a target process or activity with all its descriptors and assignments. The resulting copy and its method content elements are disconnected from the original capability pattern.

The copied capability pattern elements can be altered, deleted or changed without affecting the original method content elements. Method elements in a deep-copied capability pattern appear in black.

The difference between copying and deep copying is – is what?

Iteration, phase and activity are three types of activity and a process is a sequence of these activities and the copying capability pattern procedure can be used to **deep copy any activity**, not only capability patterns.

Deep Copy creates a **Not Applicable** variability relationship type to the original activity.

To apply a capability pattern using deep copy:

1. Make sure that you are in the **Authoring** perspective.
2. Expand the folder (node) in the **Library** view for **Processes** and the **Delivery Processes** or the **Capability Pattern** folder (node) and select the process that will be linked to other activities through a variability relationship[205].
3. Open the process editor by double-clicking the process or by right-clicking it and selecting **Edit** from the pop-up menu.
4. Select your process authoring view, by clicking the **Work Breakdown Structure**, **Team Allocation** or the **Work Product Usage** tab.
5. In the Configuration View[206], expand the **Processes** and **Capability Patterns** folders (node) and select a capability pattern or its parts that you want to copy into an activity in a target process. *Note that you select and drag the parts you want to copy from the **Configuration** view and not the Library view.*
6. With the mouse, drag the capability pattern or parts into the activity (process, iteration, phase or activity) where you want the copy to go. To select multiple activities in a capability pattern, press the **Ctrl** or **Shift** keys while you select.
7. In the menu that opens, click **Deep Copy**.
8. A second prompt asks, "*Do you want to copy all descriptors?*"

   Attention: This second prompt only appears when a capability pattern (or activity) contains descriptors that are outside of the target delivery process' default configuration.

   - When selecting '**No**' the descriptors that are not part of the target delivery process' default configuration are not copied.

[205] You can select the process and open the process editor from both the Library view and the Configuration view when the default configuration for the process has been selected.

[206] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

- When selecting '**Yes**' the descriptors that are not part of the target delivery process' default configuration will be copied over.

Related topics

Activity Variability
Create Capability Patterns
Capability Patterns Reuse
Copy Capability Patterns
Extend Capability Patterns
Develop Work Breakdown Structures
Develop Team Allocation Structures
Develop Work Product Usage Structures
Process Element Properties View[207]

9.7.3. Extend Capability Patterns

If an activity or a Capability Pattern is extended into a target process or activity, its method content elements retain a connection to the original capability pattern. Extended capability pattern elements cannot be deleted or modified and appear as read-only in green italic text.

Iteration, phase and activity are three types of activity and a process is a sequence of these activities and the copying capability pattern procedure can be used to **extend any activity**, not only capability patterns.

Extend creates an **Extends** variability relationship type to the original activity.

To extend a capability pattern:

1. Make sure that you are in the **Authoring** perspective.
2. Expand the folder (node) in the **Library** view for **Processes** and the **Delivery Processes** or the **Capability Pattern** folder (node) and select the process that will be linked to other activities through a variability relationship[208].
3. Open the process editor by double-clicking the process or by right-clicking it and selecting **Edit** from the pop-up menu.
4. Select your process authoring view, by clicking the **Work Breakdown Structure**, **Team Allocation** or the **Work Product Usage** tab.
5. In the Configuration View[209], expand the **Processes** and **Capability Patterns** folders (node) and select a capability pattern or its parts that you want to copy into an activity in a target process. *Note that you select and drag the parts you want to copy from the **Configuration** view and not the Library view.*

[207] While editing a process, you can use the Process Element Properties View to edit all details for a single element in the process. If you click in any row in a process display, you will see the full details of the process element in the row in the Properties View.

[208] You can select the process and open the process editor from both the Library view and the Configuration view when the default configuration for the process has been selected.

[209] The Configuration view displays content elements in a library that is filtered by a method configuration. A method configuration is a subset of the method library content. The view displays a preview of the content elements that will be published or exported for this configuration and allows you to preview the published site's navigation views.

6. With the mouse, drag the capability pattern or parts into the activity (process, iteration, phase or activity) where you want the copy to go. To select multiple activities in a capability pattern, press the **Ctrl** or **Shift** keys while you select.
7. In the menu that opens, click **Extend**. The capability pattern is extended and its name is displayed in green, which indicates it is part of a process defined elsewhere.

Alternative procedure:

1. Or, in the process view, right-click an activity and click **Apply Pattern** → **Extend**. The **Select Dialog: Processes** window opens.
2. Drill down into the process tree until you locate the capability pattern that you want to extend.
3. Select the pattern and click **OK**. The Select Dialog: Processes window closes and the capability pattern is applied.

Suppress

If you do not want a specific activity or task in the capability pattern to be included in your process, you can "suppress" it. The activity or task will not appear in the published process nor will it be exported to a project-planning tool.

1. To suppress a process element, right-click the element and click **Suppress** in the pop-up menu. The element is disabled in the process view.
2. If you later want to reverse the action, you can right-click the element and click **Reveal** in the pop-up menu.

Related topics

Activity Variability
Create Capability Patterns
Capability Patterns Reuse
Copy Capability Patterns
Deep Copy Capability Patterns
Local Contribution
Local Replacement
Local Replacement and Deep Copy
Develop Work Breakdown Structures
Develop Team Allocation Structures
Develop Work Product Usage Structures
Process Element Properties View[210]

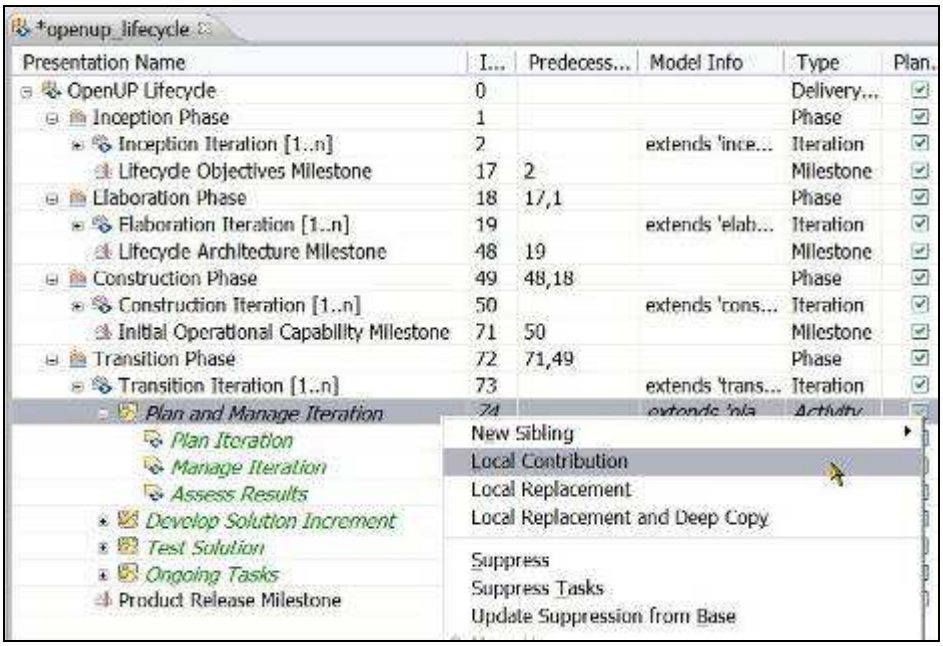
9.7.3.1. Local Contribution

You cannot modify, add or delete elements to an extended capability pattern. If you need to add supplementary local tasks to an extended activity, you first have to change the status of the extended activity to "Local Contribution".

[210] While editing a process, you can use the Process Element Properties View to edit all details for a single element in the process. If you click in any row in a process display, you will see the full details of the process element in the row in the Properties View.

To add elements to an activity in an extended capability pattern:

1. Locate the activity's parent element or parent's parent and so forth until you reach the top level in green, meaning the top parent of the extended activity.



2. Right-click the top level of the extended activity and click **Local Contribution**. The activity becomes local and appears in black. Do the same with all parent elements, from the top to the bottom, until you reach the activity you want to contribute to; the activity becomes local and appears in black.
3. Right-click the activity that has had its status changed to "locally contributes" and add new elements as required.

It is not clear why EPF changes semantics. It would seem that there is here another case of Variability. The help files do not explain the Replace feature, strangely enough.

Related topics

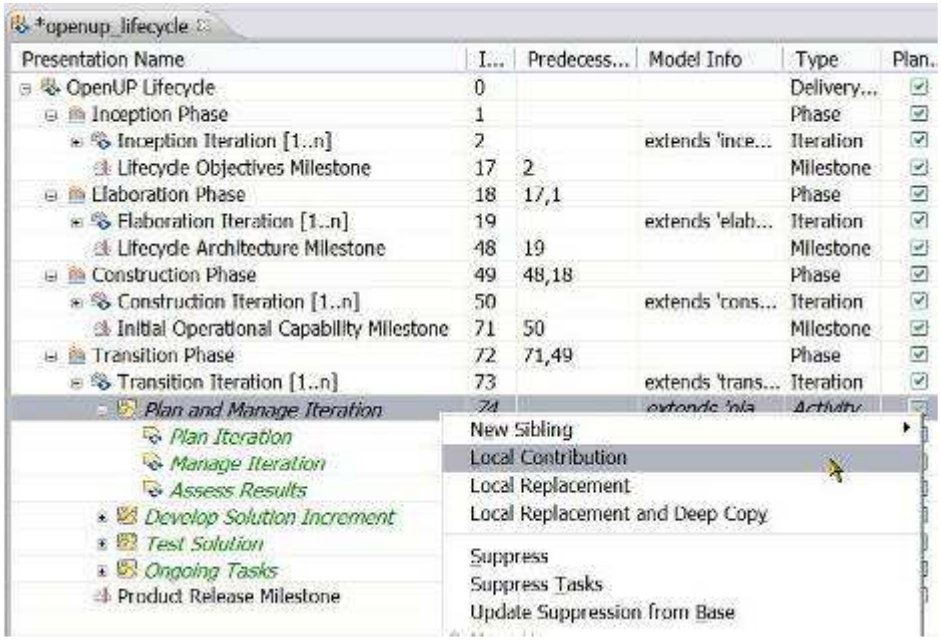
Activity Variability
Create Capability Patterns
Capability Patterns Reuse
Copy Capability Patterns
Deep Copy Capability Patterns
Local Replacement
Local Replacement and Deep Copy

9.7.3.2. Local Replacement

You cannot modify, add or delete elements to an extended capability pattern. If you need to replace the local tasks in an extended activity, you first have to apply "Local Replacement".

To replace elements of an activity in an extended capability pattern:

1. Locate the activity's parent element or parent's parent and so forth until you reach the top level in green, meaning the top parent of the extended activity.



2. To replace the extended activity, right-click the top level of the extended activity and click **Local Replacement**. The tasks in the activity that is being replaced are deleted from the local target delivery process, without affecting the original capability pattern.
3. You can now add new "Child" elements to the activity in your delivery process, in replacement of those that have been removed.

Related topics

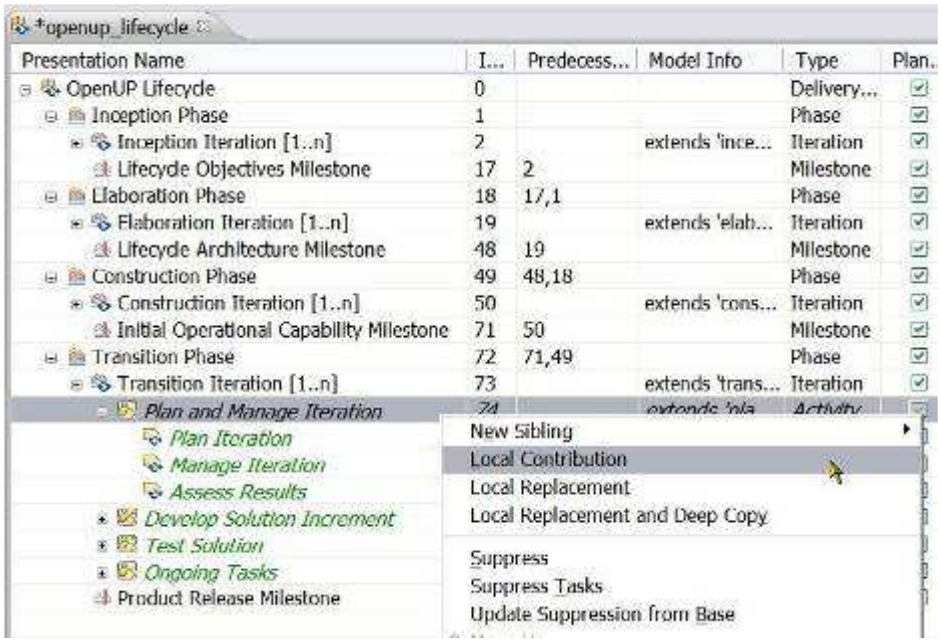
Activity Variability
Create Capability Patterns
Capability Patterns Reuse
Copy Capability Patterns
Deep Copy Capability Patterns
Local Contribution
Local Replacement and Deep Copy

9.7.3.3. Local Replacement and Deep Copy

You cannot modify, add or delete elements to an extended capability pattern. If you need to modify the local elements of an extended activity, you can apply "Local Replacement and Deep Copy". This creates a copy of all dynamically linked elements that you need to main separately.

To copy and replace elements of an activity in an extended capability pattern:

1. Locate the activity's parent element or parent's parent and so forth until you reach the top level in green, meaning the top parent of the extended activity.



2. To create a local copy of the extended activity, right-click the top level of the extended activity and click **Local Replacement and Deep Copy**. All the elements become local copies and appear in black.
3. You can now modify the activity locally to your heart's extent without affect the original capability pattern.

Related topics

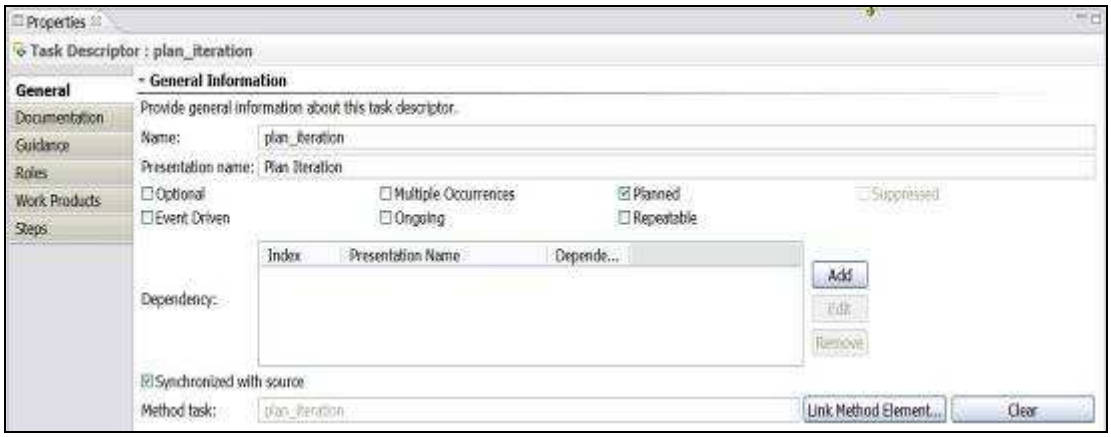
Activity Variability
Create Capability Patterns
Capability Patterns Reuse
Copy Capability Patterns
Deep Copy Capability Patterns
Local Contribution
Local Replacement

9.8. Process Element Properties View

While editing a process, you can use the process element **Properties** view to edit all details for a single element in the process. If you click in any row in a process display, you will see the full details of the process element in the row in the Properties view.

If the Properties view is not open, right-click any element in a process and click **Show Properties View**.

The Properties view is the editor for activities, task descriptors, roles descriptors, and work product descriptors. The information for a descriptor is similar to the corresponding method element editor, but it also has information related to the element in the process. The process information is also displayed in the corresponding row in the process display.



General tab:

The Name and Presentation name default to the corresponding names in the base method element. These names can be changed in a descriptor.

The following attributes are used to specify certain aspects of the element in the process and can be set for all types of process elements:

- ***Optional***: If this is checked (true), the element is considered optional in the process. This means that it is safe to remove the element. If this is not checked (false), the element is considered mandatory and should not be removed from the process.
- ***Multiple Occurrences***: If this is checked (true), when the process is instantiated as a project or other process, it is expected that there will be multiple occurrences of this element. If this is not checked (false), there should only be a single occurrence of this element when the process is instantiated.
- ***Planned***: If this is checked (true), the element will be included in an export to a project management tool. If this is not checked (false), it will **not** be included in an export.
- ***Suppressed***: If this is checked (true), the element will not appear in a published process. This is usually used to modify a capability pattern included in a process. This is not checked (false) as default.

The following elements can be set for task descriptors and activities:

- ***Event Driven***: If this is checked (true), the task or activity will be initiated when a particular event occurs. If this is not checked (false), the task or activity will be initiated based on other tasks in the process.
- ***Ongoing***: If this is checked (true), the task or activity is continuous. If this is not checked (false), the task or activity has a clear start and finish in the process.
- ***Repeatable***: If this is checked (true), the task or activity can iterate when the process is instantiated. If this is not checked (false), the task or activity will only occur once in the process.
- **Dependency Field**: You can add one or more dependencies for a task or activity. Click Add and enter the index number for the task or activity on which the current element is dependent. The dependency (type) defaults to **Finish-to-Start**. If you want to set another dependency type, click **Edit** and select the required dependency type in the dependency drop-down box (Finish-to-Start, Finish-to-Finish, Start-to-Start, Start-to-Finish). You can enter a different index number in the **Edit** window.

You can also remove a dependency by selecting the dependency and clicking **Remove**.

Method task: The base method element for the descriptor is displayed on the page. You can select a different element by clicking Link Method Element and selecting the required element.

Documentation tab:

The Documentation tab contains descriptive attributes for the descriptor or activity.

"Rollup" tabs:

The Properties view for an activity includes rollup tabs that provide lists of **tasks**, **roles**, or **work products** "rolled up," or summarised, for the activity.

Association tabs:

The remaining tabs (*Funny, there is only one remaining tab which is the Guidance tab)* are used to add or remove associations between the activity or the descriptor to method elements.

Related topics

Process Management[211]
Create Capability Patterns
Create Delivery Processes
Capability Patterns Reuse
Use of Descriptors in Processes
Create Method Content Elements
Create Guidance Elements

9.9. Apply Process to Method Synchronisation

Changes to information in method elements can be updated in related process elements through a mechanism called Synchronisation.

The *Synchronised with source option* is selected by default. To change this behaviour:

Change the synchronisation option

1. In the Process editor, click the **Work Breakdown Structure** tab.
2. Right-click a task and click **Show Properties View**.
3. In the **Properties** view panel, click the **General** tab.
4. Near the bottom of this panel there is a checkbox labelled **Synchronised with source**. You can check or clear this box to enable or disable automatic synchronisation. Note that you can only change the behaviour of elements that are not marked read-only.

[211] A process describes how a particular piece of work should be done. The work may have a relatively small scope, in which case it can be described as a capability pattern, or may address a full project lifecycle, in which case it can be described as a delivery process. A process can reuse method elements and combines them into a structure and sequence for carrying out work.

Custom synchronisation

Custom synchronisation at the activity level synchronises the activity descriptors by bringing in the task descriptor's associations.

1. Right-click any activity that has descriptors and click **Custom Synchronisation**.
2. The Synchronization wizard opens up. Use the wizard to select the properties of the method content that you want to be considered for synchronisation. Synchronisation is based on the options that you select. For example, if you select only mandatory output work product, then in task descriptor's properties view, you should see only output work product and not input and optional work products.
3. Click **Finish**.

Default synchronisation

Default synchronisation at the activity level synchronise activities by bringing in the task descriptor's associations.

1. Right-click any activity that has descriptors and click **Default Synchronisation** from Method Content. This will bring in all default associations of the task descriptors.
2. Click **OK**.

Related topics

Process Management[212]
Use of Descriptors in Processes
Create Method Content Elements
Create Guidance Elements

9.10. Working with Process Diagrams

You can create process diagrams to illustrate the relationships between processes elements. When you publish a method Web site, you can choose whether to include these diagrams.

EPF Composer provides three types of process diagrams:

- **Activity diagrams:** These diagrams show the subordinate activities in a higher-level activity. They also show the sequence relationships between those activities.
- **Activity detail diagrams:** These diagrams show tasks in an activity with their performing roles along with input and output work products. Activity detail diagrams are similar to workflow detail diagrams.
- **Work product dependency diagrams:** These diagrams illustrate work product dependencies on other work products.

[212] A process describes how a particular piece of work should be done. The work may have a relatively small scope, in which case it can be described as a capability pattern, or may address a full project lifecycle, in which case it can be described as a delivery process. A process can reuse method elements and combines them into a structure and sequence for carrying out work.

All three types of diagrams are generated and synchronised with the associated work breakdown structure. Changes to the process structure using the diagram editor will be automatically reflected in the work breakdown structure.

Related topics

Process Authoring Overview
Process Authoring Overview
Method Configurations Overview
Create Capability Patterns
Create Delivery Processes
Working with Process Diagrams[213]
Working with Activity Diagrams
Working with Activity Detail Diagrams[214]
Working with Work Product Dependency Diagrams
Publish Diagrams

9.10.1. Working with Activity Diagrams

Activity diagrams show the workflow of child process elements of a process, activity, phase, or iteration. The diagrams show the subordinate activities as part of a higher-level activity and the sequence relationships between those activities. The diagram can also be used to show the sequence of tasks for an activity that consists of a set of tasks.

When you create the diagram, the sequence of activities, or control flow depicted in your diagram is reflected in the precedence for these activities in your work breakdown structure.

You can create activity diagrams internally by using the EPF Composer diagram editor or you can choose to include diagrams created externally by another application.

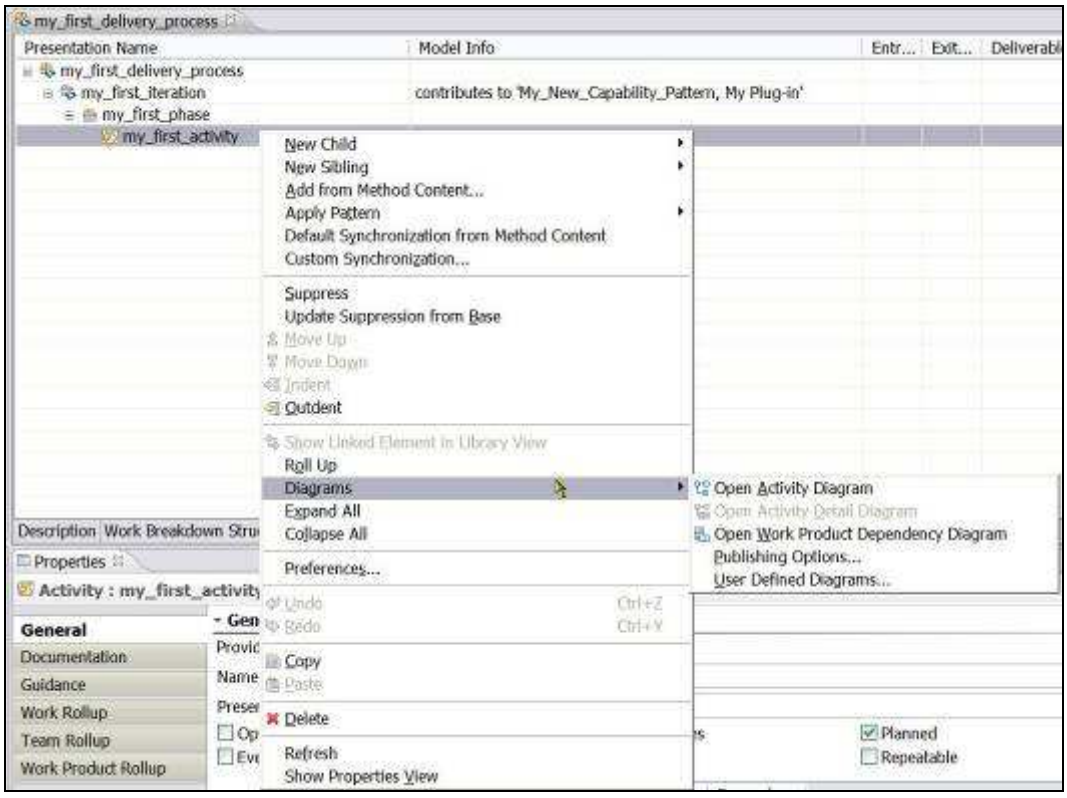
To create activity diagrams:

1. Make sure that you are in the **Authoring** perspective.
2. Expand the folder (node) in the **Library** view for Processes and the Capability Patterns or Delivery Processes node and select the process that contains the activity that the diagram will model[215].
3. Open the process editor by double-clicking the process or by right-clicking it and selecting **Edit** from the pop-up menu.
4. Click the **Work Breakdown Structure** tab and select the activity to be modelled by the diagram by right-clicking the item.
5. Select **Diagrams → Open Activity Diagram** from the pop-up menu. You can also specify your own diagram image to use for this diagram by selecting **Diagrams → User Defined Diagrams**.

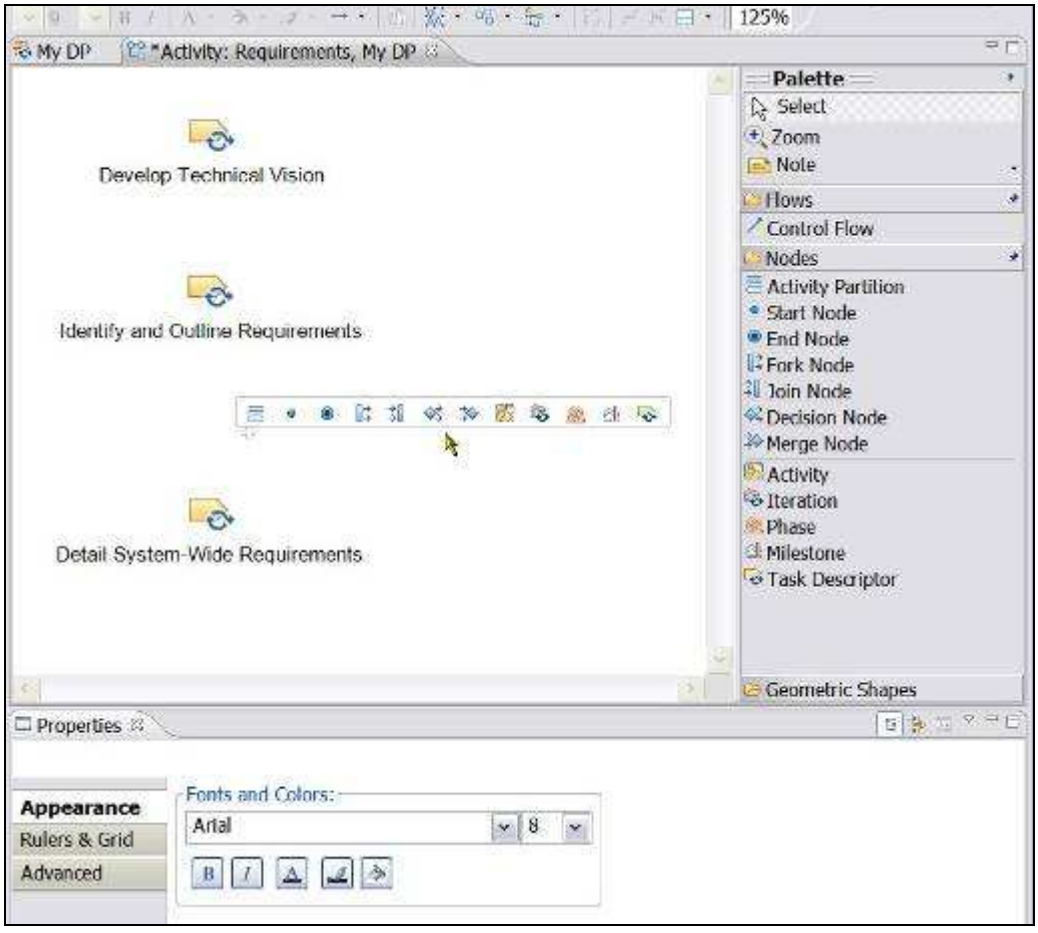
[213] You can create process diagrams to illustrate the relationships between processes. When you publish a method Web site, you can choose whether or not to include these diagrams.

[214] Activity detail diagrams show the tasks to be performed as part of an activity that is associated with a particular role. These diagrams also show mandatory input and output work products for each task. Activity detail diagrams are suitable for activities that consist of only child tasks. You can create activity detail diagrams by using the Diagram editor, and you can include diagrams that are created by other applications.

[215] You can select the process and open the process editor from both the Library view and the Configuration view when the default configuration for the process has been selected.

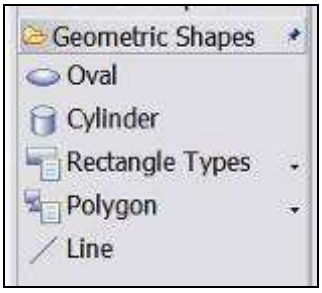


6. The activity diagram editor opens:

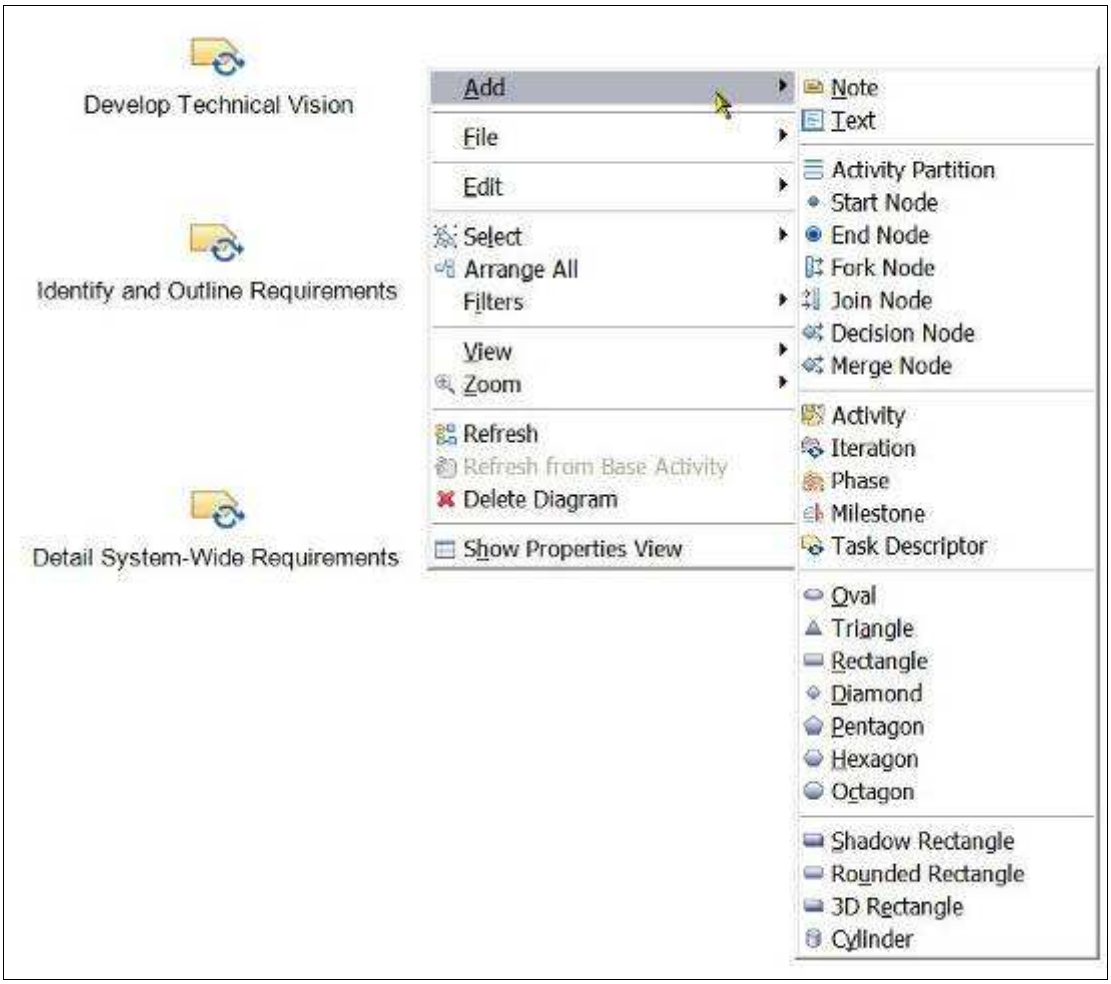


7. There are multiple ways to access to the diagram editing options:

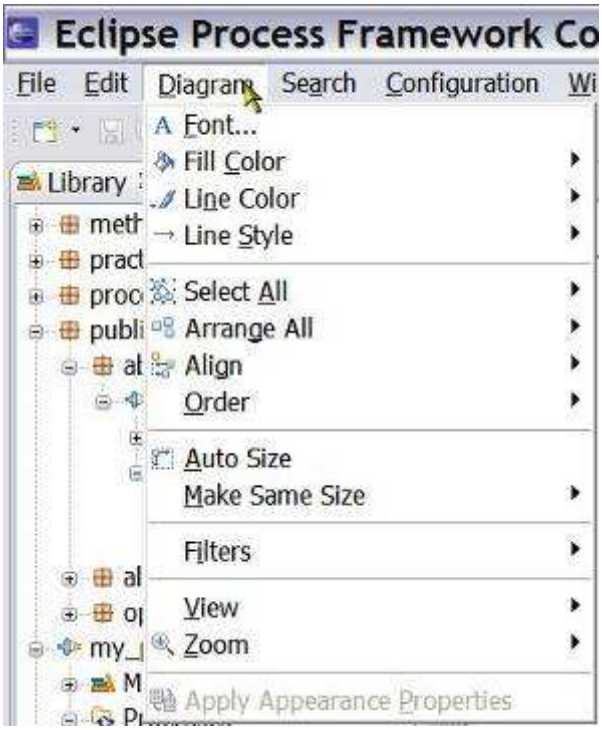
a. Draw or modify the activity diagram using the editor **Palette**. You can set different options for the palette by right-clicking in the palette area to access a pop-up window with different options.
b. There is a button bar on the top for editing. You select the objects and click on the edit buttons to change the appearance.
c. The activity diagram **Properties** view contains three tabs: **Appearance**, **Rulers & Grid** and **Advance** tabs with options for editing the diagram.
d. By letting the mouse hover over the surface of the diagram itself will produce a special fleeting horizontal "window" which let you select and add different types of nodes  . Each node type is explained below
e. By clicking the "Geometric Shapes" in the palette, several shapes can be added to the diagram:



f. By right-clicking the surface of the diagram, a window pops up which lets you add nodes and activities, objects, notes and text.



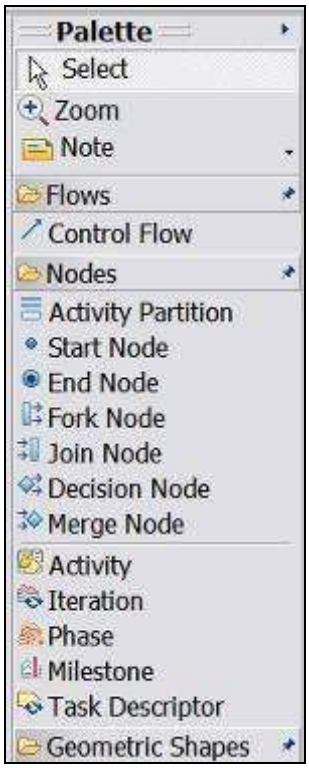
g. In the menu toolbar, click **Diagram** to have access to multiple sets of functionality:



8. When you modify a diagram, a ***** symbol is displayed left to the name of the diagram tab, indicating that it needs to be saved. There are four ways to save it:
   - Close the diagram tool and click **Yes** in response to the question about saving the changes
   - Click one of the disk icons (all or current) in the toolbar
   - Use the shortcut "***Ctrl+s***"
   - Click **File** → **Save**

Diagram editor tools

The diagram editor palette has several tools for creating diagrams.



- **Select**: Use this tool to select elements or portions of your diagram for manipulation using the drawing aids.
- **Zoom**: Use this tool to zoom in and zoom out your diagram. Left-click to zoom in, Shift + Left-click to zoom out, and drag to zoom to a selection.
- **Note and Note Attachment**: Use this tool to add notes to the diagram and link the notes to other diagram elements.
- **Text**: Use this tool to add your own text to the diagram. After you add the text to the diagram, you can open the Properties view via the pop-up menu to set display and font properties.
- **Control Flow**: Use this tool to draw directional arrows from one node on your diagram to another.
- **Activity Partition**: Use this tool to create 'swim-lanes' in your activity diagrams. After you create a partition, you can drag other elements inside to add them to the partition. You can alternatively add or remove elements to a partition via the Advanced Properties view.
- **Start Node and End Node**: Use these tools to identify the start and end of the workflow.
- **Fork Node**: Use this tool to depict the beginning of concurrent threads of control.
- **Join Node**: Use this tool to depict the end of concurrent threads of control. The workflow cannot continue past the **Join Node** until all of the flows that it joins have completed.
- **Decision Node**: Use this tool to show the control flow coming out of a decision.
- **Merge Node**: Use this tool to merge two or more distinct flows back to one common flow.
- You can also add new process elements to your diagram using Iteration, Phase, Activity, Milestone or Task Descriptor.

Drawing aids

The activity diagram editor provides several aids to manipulate and improve the appearance of your diagram.

- To move a selected node in one direction, pixel-by-pixel: Select the node, hold the **Control key** and press the cursor keys in the direction that you want to move the selected node.
- To move a selected node or nodes in a vertical or horizontal plane, using the mouse: Select the node or nodes to move, hold the **Shift** key and hold the primary mouse button on the selected node. Drag the node or nodes with the mouse in the direction that you want to move them.
- To align nodes with one another: Select the nodes of interest, open the pop-up menu and choose **Format** → **Align** followed by your choice of alignment.
- To bend a link line: Select the link, grab the centre dot and drag it with the mouse to the preferred point.

Activity diagrams in extended capability patterns

You can reuse an existing capability pattern or parts of a capability pattern, such as an activity, in another process. When you reuse another pattern by either copying or extending it, any diagrams that you have created are also included in your process. However when you extend a capability pattern, the diagrams from the base pattern are not updated when the diagram for the base pattern is changed.

To retrieve updates to a diagram of the base element, Right-click in the open diagram of the extending element and select "**Refresh from Base Activity**".

Related topics

Process Authoring Overview
Method Configurations Overview
Create Capability Patterns
Create Delivery Processes
Working with Process Diagrams[216]
Working with Activity Detail Diagrams[217]
Working with Work Product Dependency Diagrams
Publish Diagrams

9.10.2. Working with Activity Detail Diagrams

The activity detail diagrams show the tasks within an activity, arranged by the responsible role that performs them and the mandatory input and output work products for each task.

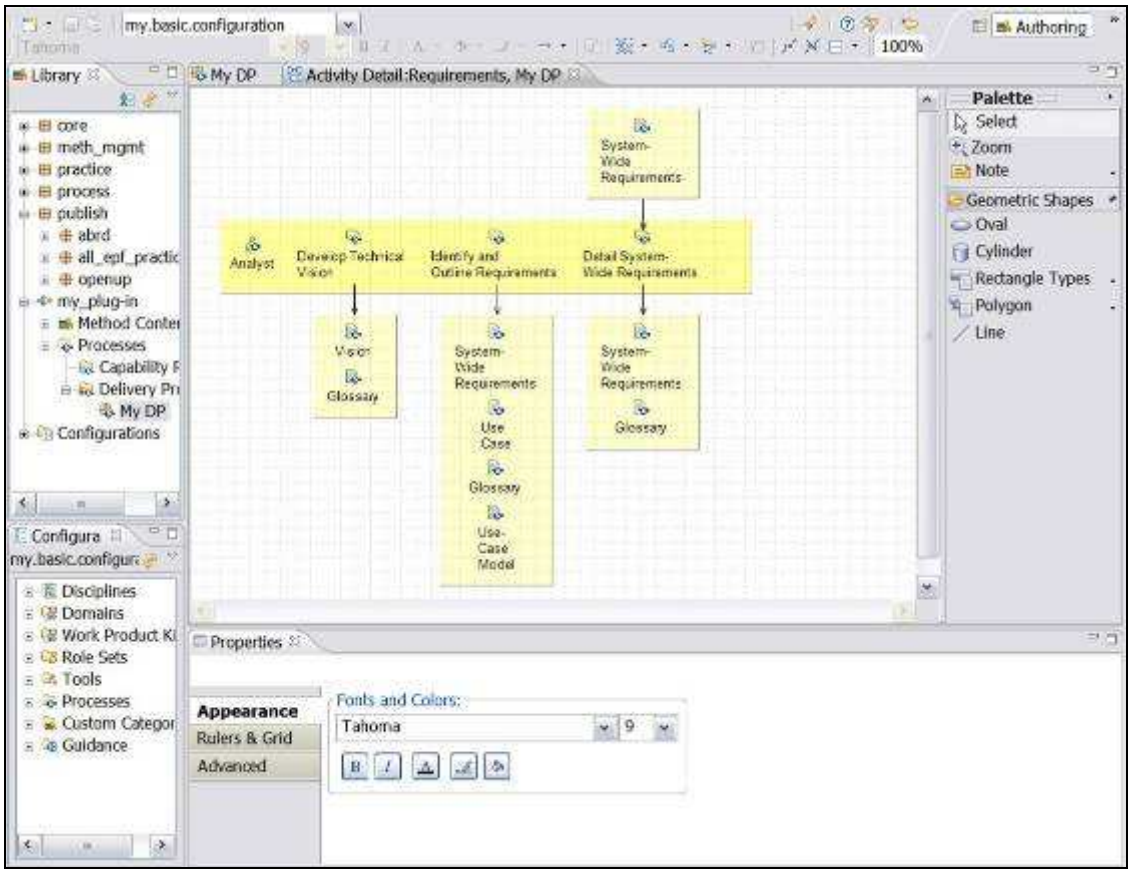
[216] You can create process diagrams to illustrate the relationships between processes. When you publish a method Web site, you can choose whether or not to include these diagrams.

[217] Activity detail diagrams show the tasks to be performed as part of an activity that is associated with a particular role. These diagrams also show mandatory input and output work products for each task. Activity detail diagrams are suitable for activities that consist of only child tasks. You can create activity detail diagrams by using the Diagram editor, and you can include diagrams that are created by other applications.

The automatically generated diagrams show all the tasks associated with a particular role, with their mandatory inputs and outputs. You can also create or modify activity detail diagrams by using the diagram editor and the palette tools. You can also include diagrams created by other applications.

To create activity detail diagrams:

1. Make sure that you are in the **Authoring** perspective.
2. Expand the folder (node) in the **Library** view for Processes and the Capability Patterns or Delivery Processes node and select the process that contains the activity that the diagram will model[218].
3. Open the process editor by double-clicking the process or by right-clicking it and selecting **Edit** from the pop-up menu.
4. Click the **Work Breakdown Structure** tab and select the activity to be modelled by the diagram by right-clicking the item.
5. Select **Diagrams → Open Activity Detail Diagram** from the pop-up menu. The editor is enabled only if the activity contains tasks descriptors. You can also specify your own diagram image to use for this diagram by selecting **Diagrams → User Defined Diagrams**.

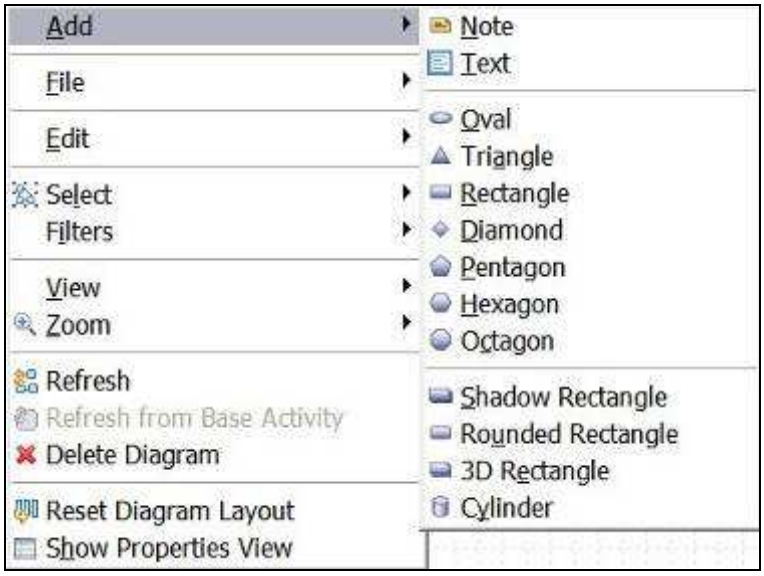


6. The diagram is generated automatically and it shows the tasks within the selected activity.[219] The tasks are arranged by the responsible role that performs them.

[218] You can select the process and open the process editor from both the Library view and the Configuration view when the default configuration for the process has been selected.

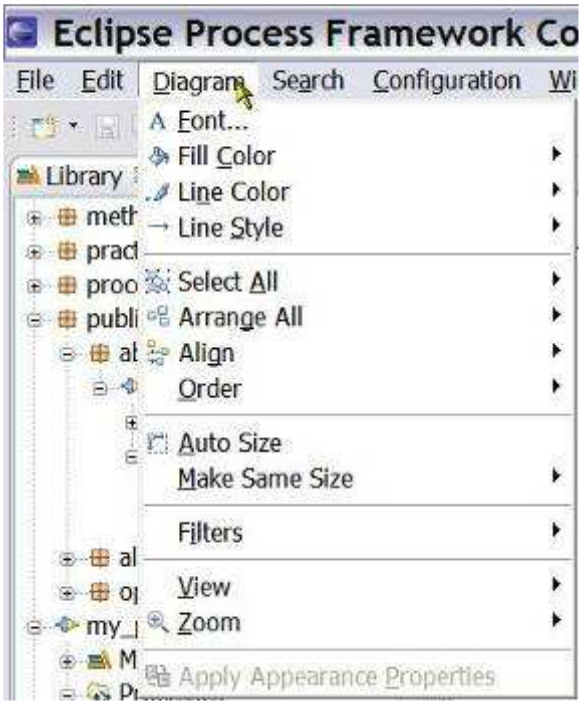
Activity detail diagrams also show the mandatory input and output work products for each task.

7. If you use generated diagrams, make sure that you do not have out-of-date activity detail diagrams. To remove old diagrams, right-click in the diagram editor to open the menu and select **Delete Diagram**.
8. You can modify the activity detail diagram by editing it in the diagram editor. There are multiple ways to access to the diagram editing options:
   a. Draw or modify the activity detail diagram using the editor **Palette**. You can set different options for the palette by right-clicking in the palette area to access a pop-up window with different options.
   b. There is a button bar on the top for editing. You select the objects and click on the edit buttons to change the appearance.
   c. The activity diagram **Properties** view contains three tabs: **Appearance**, **Rulers & Grid** and **Advance** tabs with options for editing the diagram.
   d. By clicking the "Geometric Shapes" in the palette, shapes become visible and they can be added to the diagram.
   e. By right-clicking the surface of the diagram, a window pops up which lets you add nodes and activities, objects, notes and text.



   f. In the menu toolbar, click **Diagram** to have access to multiple sets of functionality:

[219] The help files say: "Activity detail diagrams are **generated** if you select the Publish activity detail diagrams that have not been created in process editor option when you publish your configuration." But it seems that they are generated automatically in any case and that the option only affects the publishing of the diagram.



9. When you modify a diagram, a ***** symbol is displayed left to the name of the diagram tab, indicating that it needs to be saved. There are four ways to save it:
   - Close the diagram tool and click **Yes** in response to the question about saving the changes
   - Click one of the disk icons (all or current) in the toolbar
   - Use the shortcut "***Ctrl+s***"
   - Click **File** → **Save**

Related topics

Process Authoring Overview
Method Configurations Overview
Create Capability Patterns
Create Delivery Processes
Working with Process Diagrams[220]
Working with Activity Diagrams
Working with Work Product Dependency Diagrams
Publish Diagrams

9.10.3. Working with Work Product Dependency Diagrams

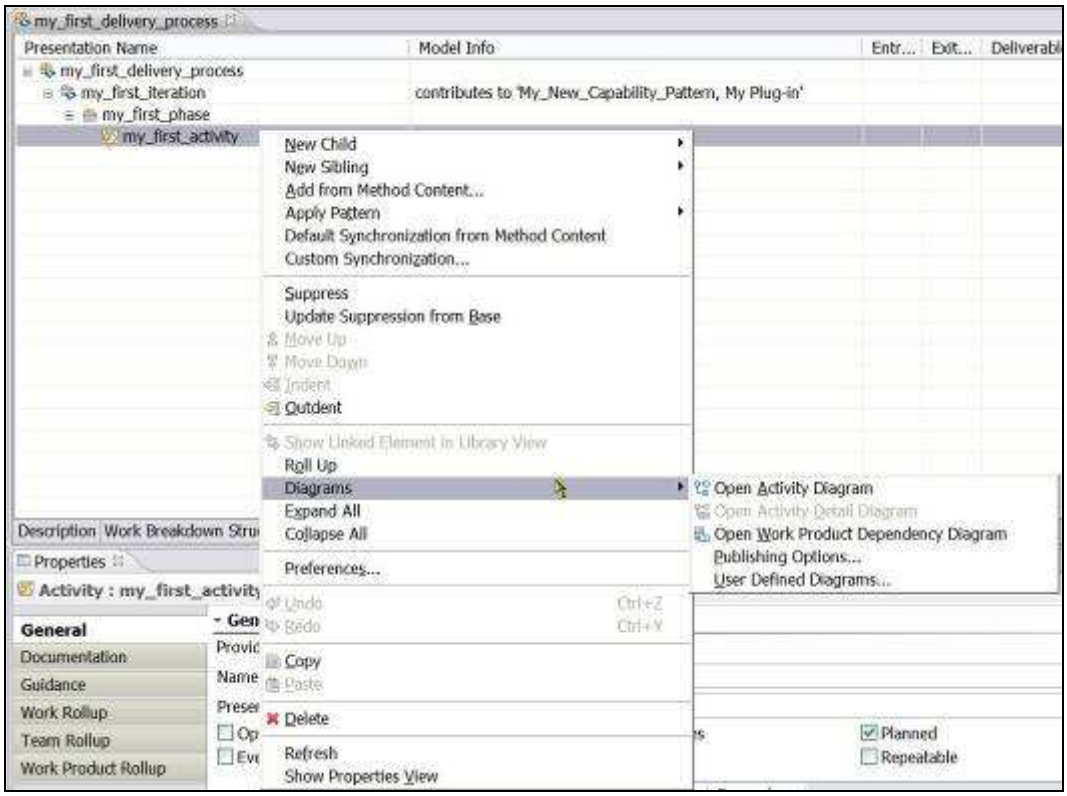
A 'Work Product Dependency Diagram' illustrates the dependencies of a work product on other work products. The diagrams show how one or more work products are used in the creation of another work product within a particular activity in a process and provide traceability of the work products.

The diagrams must be created manually. Relationships depicted in these diagrams are not reflected elsewhere in your process.

[220] You can create process diagrams to illustrate the relationships between processes. When you publish a method Web site, you can choose whether or not to include these diagrams.

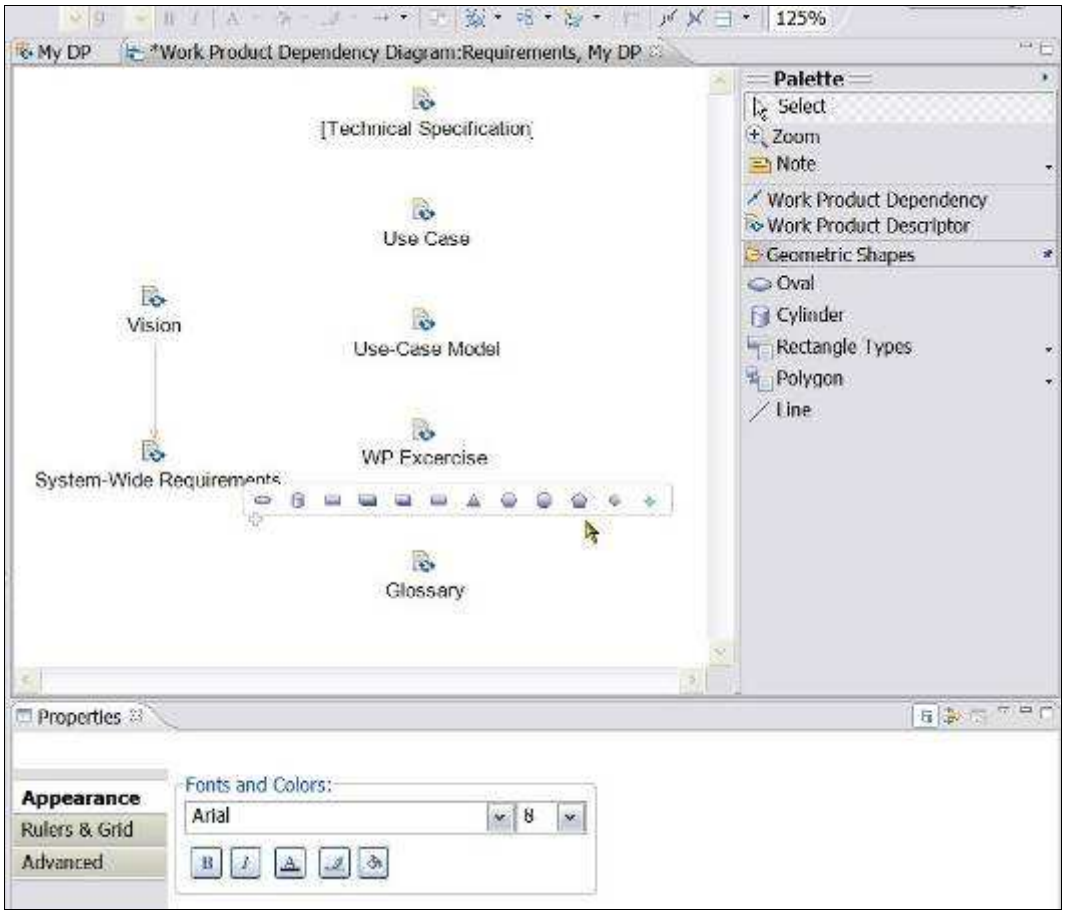
To create work product dependency diagrams:

1. Make sure that you are in the **Authoring** perspective.
2. Expand the folder (node) in the **Library** view for Processes and the Capability Patterns or Delivery Processes node and select the process that contains the activity that the diagram will illustrate[221].
3. Open the process editor by double-clicking the process or by right-clicking it and selecting **Edit** from the pop-up menu.
4. Click the **Work Product Usage** tab and select the activity to be rendered by the diagram by right-clicking it.
5. Select **Diagrams → Open Work Product Dependency Diagram** from the pop-up menu. You can also specify your own diagram image to use for this diagram by selecting **Diagrams → User Defined Diagrams**.

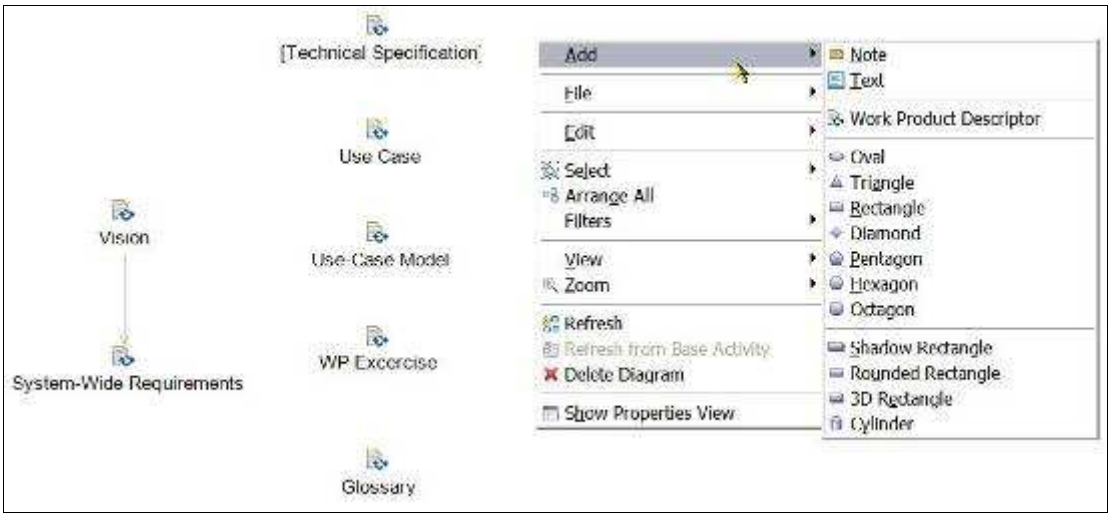


6. The Work Product Dependency Diagram editor opens:

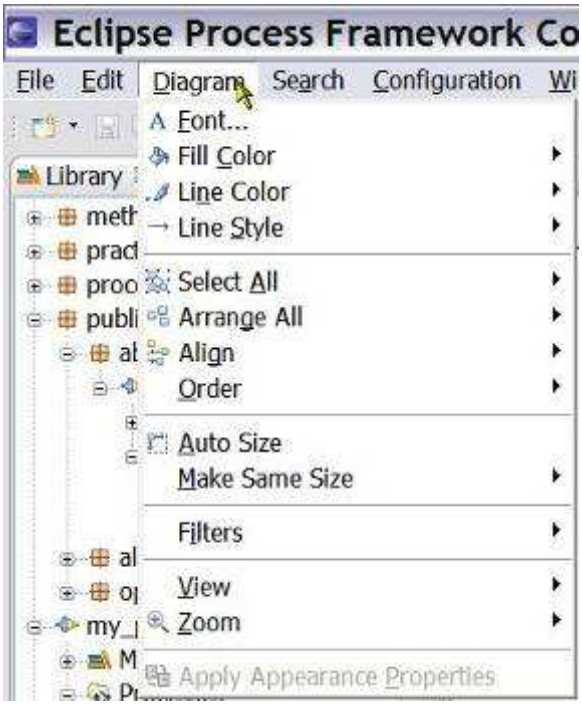
[221] You can select the process and open the process editor from both the Library view and the Configuration view when the default configuration for the process has been selected.



7. There are multiple ways to access to the diagram editing options:
   a. Draw or modify the dependency diagram using the editor **Palette**. You can set different options for the palette by right-clicking in the palette area to access a pop-up window with different options.
   b. There is a button bar on the top for editing. You select the objects and click on the edit buttons to change the appearance.
   c. The activity diagram **Properties** view contains three tabs: **Appearance**, **Rulers & Grid** and **Advance** tabs with options for editing the diagram.
   d. By letting the mouse hover over the surface of the diagram itself will produce a special fleeting horizontal "window" which let you select and add different types of objects  .
   e. By right-clicking the surface of the diagram, a window pops up which lets you add nodes and activities, objects, notes and text.



f. In the menu toolbar, click **Diagram** to have access to multiple sets of functionality:



8. When you modify a diagram, a ***** symbol is displayed left to the name of the diagram tab, indicating that it needs to be saved. There are four ways to save it:
   - Close the diagram tool and click **Yes** in response to the question about saving the changes
   - Click one of the disk icons (all or current) in the toolbar
   - Use the shortcut "***Ctrl+s***"
   - Click **File** → **Save**

Related topics

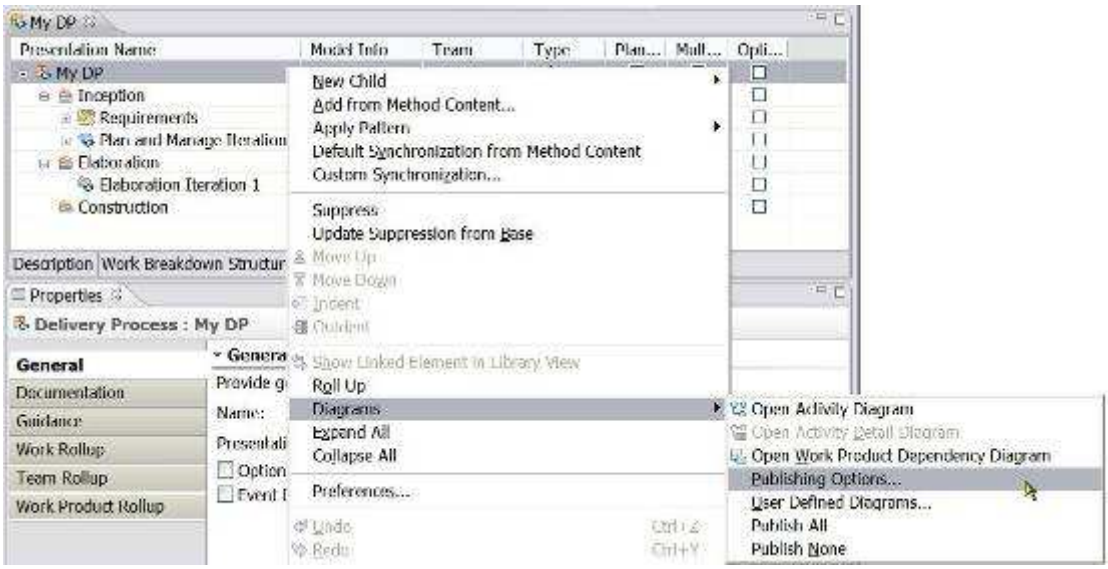
Process Authoring Overview
Method Configurations Overview
Create Capability Patterns

9.10.4. Publish Diagrams

When you publish your configuration, you may want to include some or all of your process diagrams. By default, every diagram that you create is published but you can choose not to publish a particular diagram. You can control which diagrams are published or not.

To select diagrams not to be published:

1. Make sure that you are in the **Authoring** perspective.
2. Expand the folder (node) in the **Library** view for Processes and the Capability Patterns or Delivery Processes node and select the process that contains the activity with the diagram for which you want to change the publishing options[224].
3. Open the process editor by double-clicking the process or by right-clicking it and selecting **Edit** from the pop-up menu.
4. Select your process authoring view, by clicking the **Work Breakdown Structure**, **Team Allocation** or the **Work Product Usage** tab.
5. Select the activity element that has a diagram that you do not want to publish and selecting **Diagrams** → **Publishing Options**.

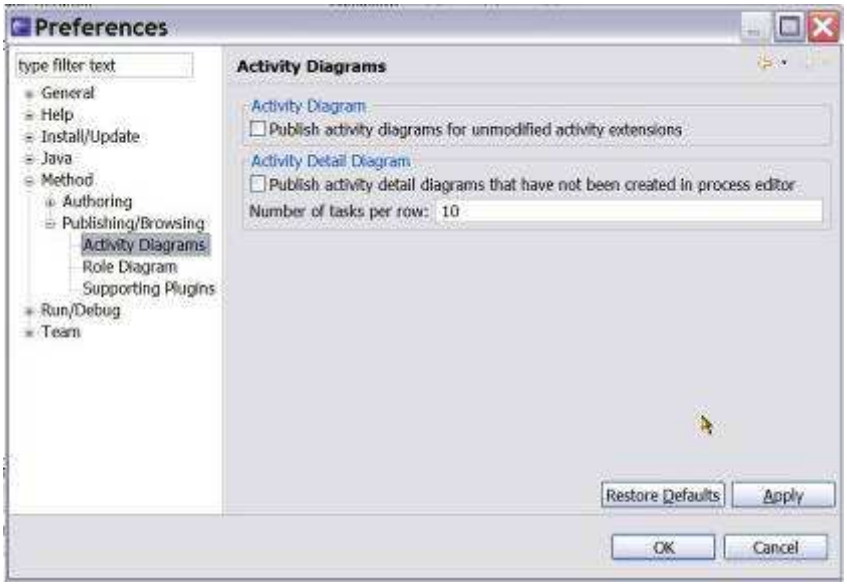


[222] You can create process diagrams to illustrate the relationships between processes. When you publish a method Web site, you can choose whether or not to include these diagrams.

[223] Activity detail diagrams show the tasks to be performed as part of an activity that is associated with a particular role. These diagrams also show mandatory input and output work products for each task. Activity detail diagrams are suitable for activities that consist of only child tasks. You can create activity detail diagrams by using the Diagram editor, and you can include diagrams that are created by other applications.

[224] You can select the process and open the process editor from both the Library view and the Configuration view when the default configuration for the process has been selected.

6. A window opens, listing each diagram that you created for this activity element. Clear the diagrams that you do not want to be published.
7. If there are multiple diagrams in the process, than you can select to have all diagrams in the process published by selecting **Publish All** or have none of them published by selecting **Publish None**.
8. To select which diagrams are published, you can set default options for activity and activity detail diagrams. To set these options, navigate to **Window** → **Preferences** → **Method** → **Publishing/Browsing** → **Activity Diagrams**.



a. **Activity Diagram**: By default, activities that are extensions of other activities are not published with their activity diagrams, only their Breakdown Structure. If the option "*Publish activity diagrams for unmodified activity extensions*" is selected, however, extending activities are published with the diagram of their base activities.

   They will however only be published if the extending activities do not define their own sub-elements, that is, if they are pure capability pattern applications without providing its own modifications.

   If these activities define their own elements, you should create a local activity diagram for the extending activity that will then be published.

b. **Activity Detail Diagram**: You can choose to generate activity detail diagrams automatically. To set this option as the default whenever you open the publishing wizard, select "*Publish activity detail diagrams that have not been created in process editor*". This setting also generates activity detail diagrams when you browse your process in the tool.

   When this option is selected, EPF Composer publishes an Activity Detail Diagram for every activity. It creates these diagrams and publishes them with a default auto-layout. If the option is cleared, then only the Activity Detail Diagrams that a user has manually created in the Process editor and saved with a custom layout are published.

Related topics

Process Authoring Overview
Method Configurations Overview
Create Capability Patterns
Create Delivery Processes
Working with Process Diagrams[225]
Working with Activity Diagrams
Working with Activity Detail Diagrams[226]
Working with Work Product Dependency Diagrams

[225] You can create process diagrams to illustrate the relationships between processes. When you publish a method Web site, you can choose whether or not to include these diagrams.

[226] Activity detail diagrams show the tasks to be performed as part of an activity that is associated with a particular role. These diagrams also show mandatory input and output work products for each task. Activity detail diagrams are suitable for activities that consist of only child tasks. You can create activity detail diagrams by using the Diagram editor, and you can include diagrams that are created by other applications.

10. Publish and Exporting

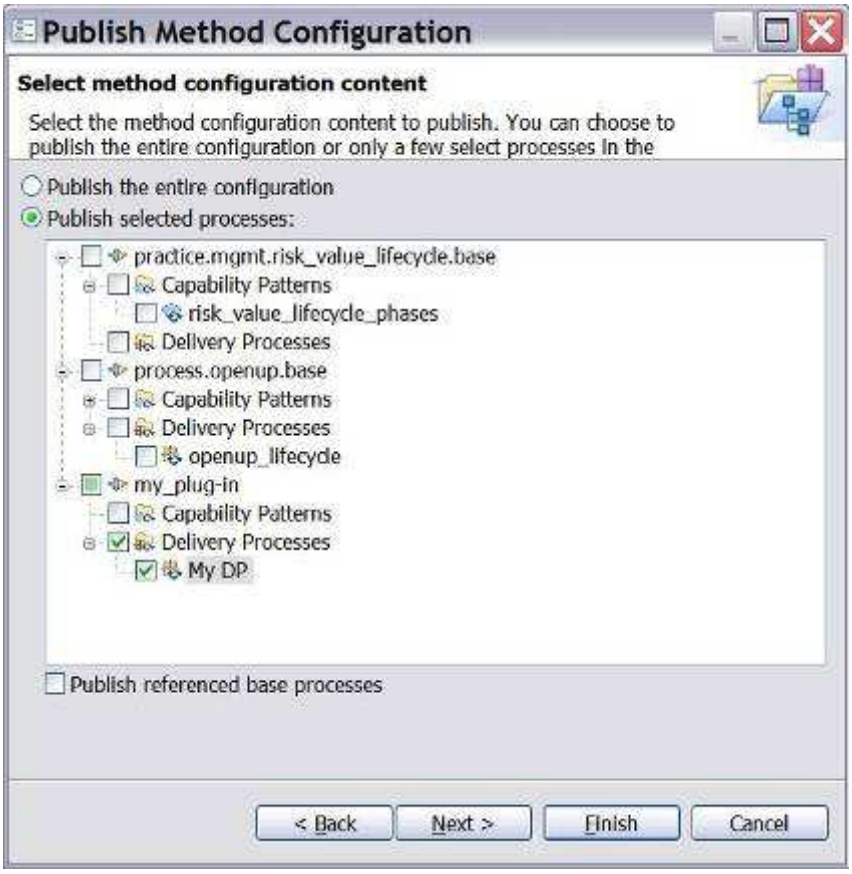
Contents

10.1. Publish Configurations as Web Sites

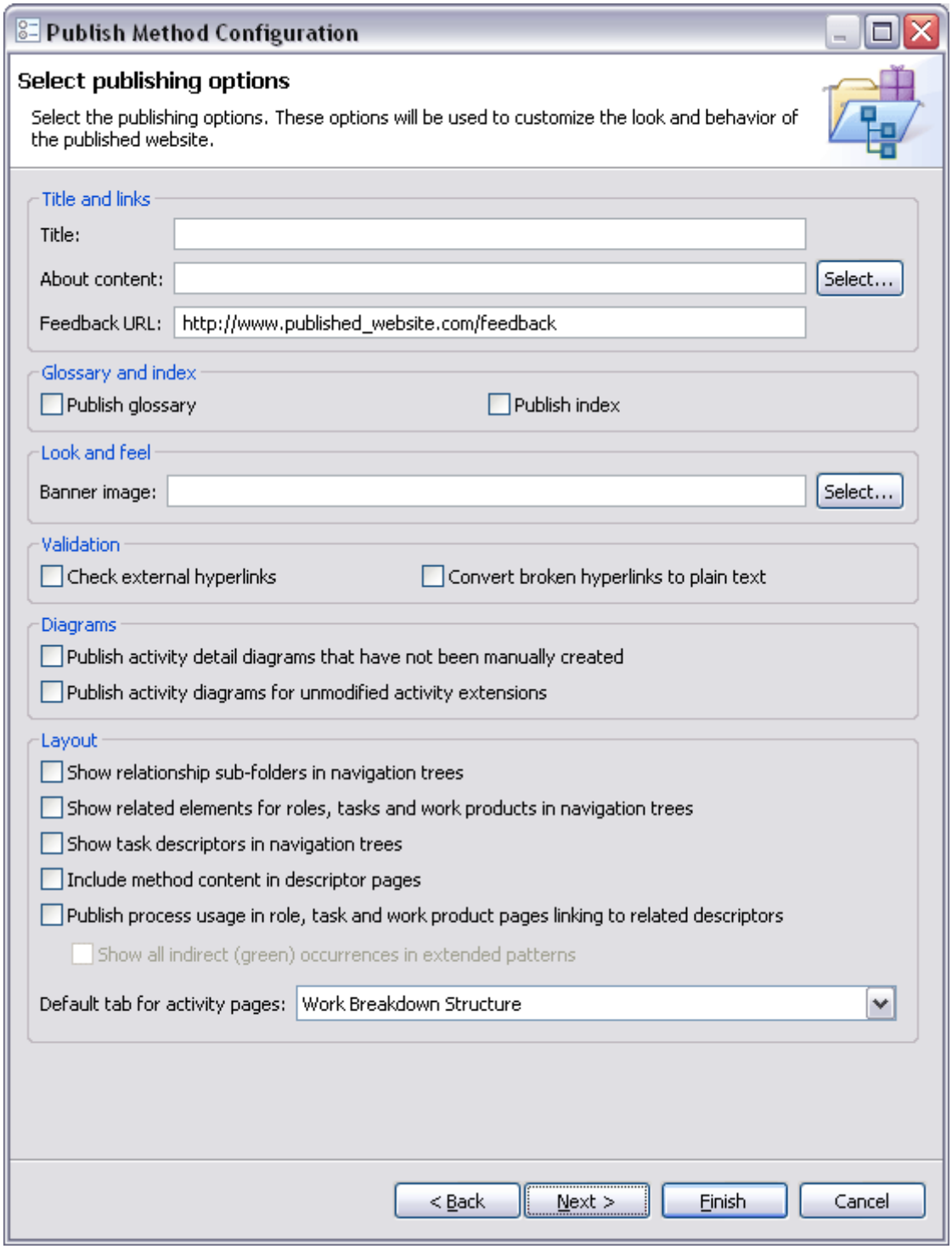
You can use the **Publish Method Configuration** wizard to generate a Web site based on a method configuration.

To publish a configuration as a Web site:

1. In the menu toolbar, click the **Configuration** menu and select **Publish** (**Configuration** → **Publish**). The Publish Method Configuration wizard opens.
2. From the list of available configurations, select the method configuration that you want to publish. Click **Next**.
3. The **Publish Method Configuration** wizard opens. You can choose to publish the entire configuration or select only a few processes in the configuration.



4. The selection of "*Publish referenced base processes*" most likely will have as a result that when activity variability relationships are established, the referenced base process will be published, in addition to the extending activity. To verify.
5. Click **Next**. Another Publish Method Configuration wizard page opens.
6. Select options to customise the look and behaviour of the published site:



7. Select options for Title and Links.

- **Title**: The title is displayed in the title bar of the browser showing the published site.
- **About content**: Select a text or HTML file that contains the text that will be displayed when the user presses About in the published site.
- Feedback URL: For more information about the feedback URL, see page Changing Feedback Addresses.

8. Select options for Glossary and Index.

- Check the option boxes for glossary and/ or index to generate a Glossary and Index section.

9. **Select options for Look and Feel.**
   - **Publish banner**: Check this box to create a banner on top of the published content. Use the Banner image field to select a graphic file that contains your banner. If you do not enter anything here, EPF Composer will use its default banner.
10. **Select options for Validation.**
   - **Check external hyperlinks**: When this option is selected, EPF Composer checks all hyperlinks before publishing. Clear this option if you are not connected to the Internet or need to use a Proxy server (currently not supported).
   - **Convert broken hyperlinks to plain text**: If the Check external hyperlinks finds broken links and the corresponding option is not selected, EPF Composer prefixes each link with a broken link symbol. The same is true for links to content elements that are not in the published method configuration. If this option is selected, all links are converted to plain text instead and no broken link symbol is used.
11. **Select options for Diagrams.**
   - **Publish activity detail diagrams that have not been manually created**: When this option is selected, EPF Composer publishes an Activity Detail Diagram for every activity. It creates these diagrams and publishes them with a default auto-layout. If the option is cleared, then only the Activity Detail Diagrams are published that a user has manually created in the Process editor and saved with a custom layout.
   - **Publish activity diagrams for unmodified activity extensions**: By default, activities that are extensions of other activities are not published with an Activity Diagram, but only the Breakdown Structure. If this option is selected, however, extending activities are published with the diagram of their base activities if these extending activities do not define their own sub-elements, that is, are pure pattern applications without providing its own modifications. If these activities define their own elements, you should create a local activity diagram for the extending activity that will then be published.
12. **Select options for Layout.**
   - **Show relationship sub-folders in navigation trees**: If this option is selected, it will publish elements, for instance tasks with its related elements (such as input and output work products), in separate ***sub-folders***. If cleared, it will publish related elements as ***sub-elements***. It will also list only a subset of these related elements in the tree; for example, only output work products for tasks and not the inputs.
   - **Show related elements for roles, tasks and work products in navigation trees**: If this option is selected, it will publish the roles, tasks, and work products together with their related elements.
   - **Show task descriptors in navigation trees**: If this option is cleared, EPF Composer creates a breakdown for all the activities of the published processes in the tree-browser. If selected, it also **includes the descriptor level** into the tree-browser.

Selecting this option has an impact on publication time and performance of the published site.

- **Include method content in descriptor pages**: If this option is selected, it hides the distinction between method content elements and descriptors from the users of the published site. All descriptors are published without using the word "Descriptor" as their type and all textual content and guidance relationships of the related method content elements.
- **Publish process usage in role, task and work product pages linking to related descriptors**: When this option is selected, all method content pages include an additional section called "**Process Usage**" that lists all occurrences of the method content elements in all processes of the published configuration.

  This provides a quick overview to see in which processes and where in these processes method content elements have been used.

  - When this option is selected, you have also the option "**Show all indirect (green) occurrences in extended patterns**" which determines how many process occurrences are shown for a method content element.
    - If the option is **not** checked, then **only direct usage** of the elements is shown (recognisable by using a black font for these elements in the process editor's breakdown structure).
    - If the option **is** checked, then also **all indirect occurrences** are shown. Indirect usage means that the element has not been applied directly to the process, but indirectly by applying a capability pattern that had this element applied (recognisable by using a green font for these elements in the process editor's breakdown structure).
- **Default tab for activity pages**: Published activity pages comprise four tabs. This option allows selecting which of these tabs is presented first by default.

13. To get to the next set of publishing options, click **Next**.
14. Enter destination **Directory**: Decide in which directory the published site will be created.
15. Select options for Website format.
    - **Static web site**: This option creates static HTML pages in the directory selected above. If this option is selected, search is supported requiring a Java runtime environment installed on the Web site user's machine because it will use a Java Applet.
    - **Java EE Web application**: This option creates one WAR or EAR file in the location selected above that can be installed on a Java EE Web server with Servlet 2.3 capabilities. Search will be realised as a Servlet running on such servers.
16. Click **Finish**. Publishing is initiated.

Related topics

Method Configurations Overview

10.2. Export to Microsoft Project

You can export a capability pattern or delivery process as an XML file that can be opened in Microsoft® Project. This XML file contains links to specific pages in the original method library.

Dependencies in the work breakdown structure are carried over to Microsoft Project. If there are circular dependencies in work breakdown structure, Microsoft Project will issue a warning but will accept the circular dependencies.

Roles are exported as Microsoft Project **resources**. They are connected to tasks or summary tasks as follows:

- If a task is not suppressed, the primary and additional performer roles are associated with it.
- If an activity has all of its children suppressed, the primary and additional performers of all of its suppressed children are assigned to it. This activity exports as a Microsoft Project task because it has no children.

The following elements are not exported:

- Work products
- Suppressed activities
- Tasks

Microsoft Project sets the start and end dates for the project. You need to adjust them after the export has completed by using Microsoft Project.

To export to Microsoft Project:

1. Click **File** and click **Export**.
2. Click Microsoft Project and click **Next**.
3. Select a capability pattern or delivery process to export. Click **Next**.
4. Select the method configuration and export options that will be used to export the selected process.
   - Selected process
   - Method configuration
   - Publish the process Web site
     - *Publish the entire method configuration*
     - *Publish content that is only relevant to the selected process*
   - Export only breakdown elements that are planned
5. Select the publishing options that will be used to publish the Web site for the exported process. Click **Next**.
6. Enter a Name and Directory for the exported file. Use Browse to navigate to the exported folder.
7. Click **Finish** to begin the export process.

Related topics

Export XML[227]

[227] Method library content can be exported in XML format. You can choose to export the entire method library or select individual method plug-ins for export.

10.3. Export a Library Configuration

You can export a library using a method configuration and all its method plug-ins or just export one or more method configuration specifications.

To create a method configuration, see Creating New Method Configurations[228] and then use the procedure described below to export this configuration as a method library. When exporting method configurations, all referenced method plug-ins in the selected configuration will be included in the export. The export will scan all elements of the method configuration and will also export all resource files, including images, documents, and templates.

Tip: You could use this export to "clean" your library from unused resource files, by creating a method configuration that includes all your plug-ins and then exporting. As a result, all unused resource files will not be exported.

In addition to exporting the method configuration, you could simply export a method configuration specification file that you want to share with other users. In other words, you could export a file that defines a configuration, but does not contain the elements that are part of the configuration itself.

To export a library:

1. Click File → **Export**.
2. Select **Library Configuration** and click **Next**.
3. Select the type of library configuration to export.
   - Export a method configuration and all its method plug-ins
   - Export one or more method configuration specifications

   Click **Next**.
4. Select a method configuration to export. All referenced method plug-ins in the selected configuration are included in the export. Click Next. The application will perform an integrity check and report possible errors.
5. Specify a destination directory for the exported configuration. Use Browse to navigate to the location for the export.
6. Click Finish to begin the export process. The export process will take a few seconds.

Related topics

Export XML[229]
Publish Configurations as Web Sites[230]

---

[228] Method libraries can be comprised of content from many types of methods and whole families of different processes. A method configuration defines a logical subset of a method library. You use method configurations to define the scope of your authoring work and when publishing or exporting content.

[229] Method library content can be exported in XML format. You can choose to export the entire method library or select individual method plug-ins for export.

[230] You can use the Publish Method Configuration wizard to generate a Web site based on a method configuration.

10.4. Import a Library Configuration

To import a library configuration:

1. Click **File** to select **Import**. The Import wizard opens.
2. Select **Library Configuration**. Click **Next** to continue.
3. Specify the directory containing the library configuration to import. Use **Browse** to navigate to the file.
4. Review the changes that will be made to the current method library. Click **Finish**.
5. Specify the **location of your backup file**. The backup is a copy of your library before the merge takes place. You have the option to Skip the backup.
6. A window indicates when the import is complete. You will have the option to inspect the error log. Click **OK**.

Related topics

Export a Library Configuration[231]
Export XML[232]
Export to Microsoft Project[233]

10.5. Export a Method Plug-in

Method plug-ins can be exported. This provides a convenient way to distribute method content to other users.

To export a method plug-in:

1. Click **File** and click **Export**.
2. Select **Method Plug-ins** and click **Next**.
3. Select the method plug-ins to export and click **Next**.
4. Select each method plug-in to review its dependencies and click **Next**.
5. In the next page confirm the export and click **Next**.
6. Enter a location for the exported method plug-in. You can use **Browse** to navigate to the location for the exported folder.
7. Click **Finish** to begin the export process.

Related topics

Export XML
Export to Microsoft Project

---

[231] You can export a method configuration and all its method plug-ins or export one or more method configuration specifications.

[232] Method library content can be exported in XML format. You can choose to export the entire method library or select individual method plug-ins for export.

[233] You can export a capability pattern or delivery process as an XML file that can be opened in Microsoft Project. This XML file contains links to specific pages in the original method library.

10.6. Import a Method Plug-in

The import process requires a library. You must open an existing library or create a new one before importing.

To import a method plug-in:

1. Click **File** → **Import**. The Import wizard opens.
2. Select **Method Plug-ins** and click **Next**.
3. Specify the directory containing the method plug-ins to import. You can use **Browse** to navigate to the location of the file. There are two options for importing:
   - Check base plug-ins
   - Ignore and remove unresolved references
4. Review the changes that will be made to the current method library. Click **Finish** to begin the import process.
5. Specify **the location of your backup file**. The backup is a copy of your library before the merge takes place. You have the option to **Skip** the backup if you want.
6. You will be prompted when the import is complete. You will have the option to inspect the error log. Click **OK**.

Related topics

Export a Method Plug-in[234]
Import a Library Configuration
Import XML[235]

10.7. Export XML

Method library content can be exported in XML format. You can choose to export the entire method library or select individual method plug-ins for export.

It is also possible to import method library content in XML format, if the data follows the correct XML schema. You can download the latest XML schema from eclipse.org.

Due to the complexity of an import operation and its impact on a method library's integrity, it is a good practice for third-party vendors to export their library content in XML format and use the Import wizard to import the XML into a new or existing method library.

To export an XML:

1. Click **File** and select **Export**.
2. Select **XML** and click **Next**.
3. Select the type of method library content to export:
   - Export the entire method library

[234] Method plug-ins can be exported. This provides a convenient way to distribute method content to other users.

[235] Method content previously exported (including by Rational Method Composer) as an XML file can be re-imported. An imported XML file will become a method plug-in with a configuration.

- Export one or more method plug-ins

4. If you chose to export one of the method plug-ins, a window opens so that you can select the method plug-ins to export. If you chose to export the entire method library, skip the next three steps.
5. Select the method plug-ins to export and click **Next**.
6. Review the dependencies for the method plug-in that you are exporting. Click **Next**.
7. Confirm the export settings and click **Next**.
8. Enter a name and location for the XML file and click **Finish**. You can use **Browse** to navigate to the location of your choice.

Related topics

Import XML[236]
Export to Microsoft Project
Publish Configurations as Web Sites[237]

10.8. Import XML

Method content previously exported (including by Rational Method Composer) as an XML file can be re-imported. An imported XML file will become a method plug-in with a configuration.

The import process requires a library. You must open an existing library or create a new one before importing.

To import XML:

1. Click **File** and click **Import**.
2. Select **XML**. Click **Next**.
3. Specify the XML file to be imported. You can use **Browse** to navigate to the file.
4. Select one of these merge options:
   - **Override** existing method library content with the imported content
   - **Merge** imported content into the existing method library
5. Select one of these check base plug-in options:
   - Check base plug-ins
   - Ignore and remove unresolved references
6. Specify the **location of your backup file**. The backup is a copy of your library before the merge takes place. You have the option to Skip the backup if you want.
7. A window opens indicating when the import is complete. You can inspect the error log. Click **OK**.

[236] Method content previously exported (including by Rational Method Composer) as an XML file can be re-imported. An imported XML file will become a method plug-in with a configuration.

[237] You can use the Publish Method Configuration wizard to generate a Web site based on a method configuration.

Related topics

Export XML
Export to Microsoft Project

11. Sharing Content Using Version Control Systems

Contents

11.1. Using CVS[238] to Share Libraries and Elements

Contents

11.1.1. Install and Configure CVS

CVS assumes an optimistic usage model in which multiple users make changes to files simultaneously. However, because compare and merge operations are not supported; the CVS Watch/Edit function should be used to broadcast notifications to other users who may be working on files concurrently.

To install CVS:

1. Select **Window** → **Preferences**.
2. Expand the tree menu for **Team** on the left side of the Preferences window, and then expand the CVS sub-menu.
3. Click **Watch/Edit**. Set the Watch/Edit check boxes as follows:
   - CVS Watch/Edit: Select *Configure projects to use Watch/Edit on checkout*.
   - When read-only files are modified in an editor: Select *Send a CVS edit notification to the server*.
   - Before a CVS edit notification is sent to the server: Select *Always prompt*.
   - Update edited files: Select *Prompt to update if out of date*.
4. Click **OK**.

[238] Concurrent Versions System (CVS), also known as the Concurrent Versioning System,

5. Switch to the **CVS Repository Exploring** perspective.  This perspective shows files on the CVS server.
6. Right-click anywhere in the blank area of the CVS repository view. Select **New → Repository location**.
7. Enter settings supplied by the CVS system administrator. These settings should include the following:
   - Host
   - Repository path
   - User
   - Password
   - Connection type

Now the view can be populated.

Related topics

Create a New View with CVS
Add New Libraries to CVS[239]
Add a Method Plug-In to CVS[240]
Add Elements to CVS*Error! Bookmark not defined.*
Delete Elements under CVS[241]
Edit Elements under CVS[242]
Move Elements under CVS

11.1.2. Create a New View with CVS

To create a new view with CVS:

1. Switch to the **CVS Repository Exploring** perspective. 
2. Go to the **HEAD** or **Branch**.
3. To check out a library, right-click it and select **Check out**.

By default the library is checked out in the application workspace.

Related topics

Install and Configure CVS
Add New Libraries to CVS[243]
Add a Method Plug-In to CVS[244]

---

[239] You can add a method library to CVS so that you can control the version of various files in your library and allow multiple content authors to work in parallel.

[240] You can add method and process elements to CVS so that you can control changes to each element as its content is written and its relationships are defined.

[241] You can use CVS to ensure that when you delete a method or process element, the dependent files are appropriately version-controlled.

[242] You can use CVS to track changes that happen to the content and relationships of method and process elements. You can use CVS to track who changed what, when, and why.

[243] You can add a method library to CVS so that you can control the version of various files in your library and allow multiple content authors to work in parallel.

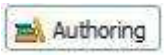
Add Elements to CVS[Error! Bookmark not defined.]
Delete Elements under CVS[245]
Edit Elements under CVS[246]
Move Elements under CVS

11.1.3. Add New Libraries to CVS

You can add a method library to CVS so that you can control the version of various files in your library and allow multiple content authors to work in parallel.

If your CVS server is not pre-populated with a method library, follow these steps to create a new library and add it to version control:

Add a new library to CVS:

1. Make sure you are in the **Authoring** perspective. 

2. Click **File** → **New** → **Method Library**.
3. In the **New Method Library** window, enter a Name, Description, and Location for the new library. Click **Finish**.
4. Switch to the **Resource** perspective.
5. Select the **Project Explorer** view.
6. Right-click the folder having the same name as the library created in Step 3.
7. Click **Team** → **Share Project**. The window opens.
8. To add your library to a new or existing repository location, follow the instructions in the window.

Related topics

Install and Configure CVS
Create a New View with CVS
Add a Method Plug-In to CVS[247]
Add Elements to CVS[Error! Bookmark not defined.]
Delete Elements under CVS[248]
Edit Elements under CVS[249]
Move Elements under CVS

[244] You can add method and process elements to CVS so that you can control changes to each element as its content is written and its relationships are defined.

[245] You can use CVS to ensure that when you delete a method or process element, the dependent files are appropriately version-controlled.

[246] You can use CVS to track changes that happen to the content and relationships of method and process elements. You can use CVS to track who changed what, when, and why.

[247] You can add method and process elements to CVS so that you can control changes to each element as its content is written and its relationships are defined.

[248] You can use CVS to ensure that when you delete a method or process element, the dependent files are appropriately version-controlled.

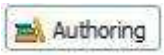
[249] You can use CVS to track changes that happen to the content and relationships of method and process elements. You can use CVS to track who changed what, when, and why.

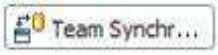
11.1.4. Add a Method Plug-In to CVS

Adding new method plug-ins to CVS version control allows multiple authors to work separately.

You must add your library to version control before performing the following steps:

To add a method plug-in to CVS:

1. Make sure that you are in the **Authoring** perspective 

.
2. Click **File** → **New** → **Method Plug-in**.
3. In the *Send an Edit Notification to CVS* window, click **Yes** to lock *library.xmi* for editing. If the file is currently being edited by another user, a window opens with the option to continue or not. Click **No** so that you do not need to merge the files later. **Exit** the operation.
4. In the New Method Plug-in window, enter the required information then click Finish.
5. Switch to the **Team Synchronising** perspective 

.
6. Make sure that the plug-in that you created in Step 4 is listed with *library.xmi* and any other dependent files.
7. Select the top-level folder for your project and click Commit.

Related topics

Install and Configure CVS
Create a New View with CVS
Add New Libraries to CVS[250]
Add Elements to CVS*Error! Bookmark not defined.*
Delete Elements under CVS[251]
Edit Elements under CVS[252]
Move Elements under CVS

11.1.5. Add Elements to CVS

You can add method and process elements to CVS so that you can control changes to each element as its content is written and its relationships are defined.

These steps apply to elements such as roles, tasks, work products, guidance, categories, configurations, capability patterns, and delivery processes.

To add method and process elements CVS:

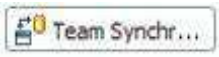
1. Make sure that you are in the **Authoring** perspective 

.
2. Use the tree browser to choose the location for the new element.

[250] You can add a method library to CVS so that you can control the version of various files in your library and allow multiple content authors to work in parallel.

[251] You can use CVS to ensure that when you delete a method or process element, the dependent files are appropriately version-controlled.

[252] You can use CVS to track changes that happen to the content and relationships of method and process elements. You can use CVS to track who changed what, when, and why.

3. Right-click the parent element and select **New**. Choose the appropriate element type.
4. In the *Send an Edit Notification to CVS* window, click **Yes** to lock *plugin.xmi* for further editing. If the file is currently being edited by another user, a window opens with the option to continue or not. Click **No** so that you do not need to merge the files later. **Exit** the operation
5. Enter content for the new element in the editor window. While an element is being edited, an asterisk (*) is next to the element's name at the top of the editor window. This indicates that the version cached in memory is out-of-synch with the local file system.
6. Click **File** → **Save**. The copy in the local file system is updated and the asterisk (*) is cleared.
7. Switch to the **Team Synchronising** perspective 

.
8. Make sure that the element created in Step 5 is listed with *plugin.xmi* and any other dependent files.
9. Select the top-level folder for your project and click Commit.

Related topics

Install and Configure CVS
Create a New View with CVS
Add New Libraries to CVS[253]
Add a Method Plug-In to CVS[254]
Delete Elements under CVS[255]
Edit Elements under CVS[256]
Move Elements under CVS

11.1.6. Delete Elements under CVS

You can use CVS to ensure that when you delete a method or process element, the dependent files are appropriately version-controlled.

These steps apply to elements such as roles, tasks, work products, guidance, categories, configurations, capability patterns, and delivery processes.

To delete method and process elements under CVS:

1. Open the **Authoring** perspective 

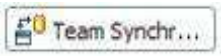
.
2. Use the tree browser to choose the element that you want to delete.
3. Right-click the element and click **Delete**.

[253] You can add a method library to CVS so that you can control the version of various files in your library and allow multiple content authors to work in parallel.

[254] You can add method and process elements to CVS so that you can control changes to each element as its content is written and its relationships are defined.

[255] You can use CVS to ensure that when you delete a method or process element, the dependent files are appropriately version-controlled.

[256] You can use CVS to track changes that happen to the content and relationships of method and process elements. You can use CVS to track who changed what, when, and why.

4. In the *Send an Edit Notification to CVS window*, click **Yes** to lock the needed files. If a file is currently being edited by another user, a window opens with the option to continue or not. To avoid needing to merge the files later, click **No** and stop deleting that element.
5. Click **File** → **Save**.
6. Switch to the **Team Synchronising** perspective 

.
7. Make sure that the element that you deleted in Step 4 is listed with other dependent files.
8. Select the top-level folder for your project and click **Commit**.

Related topics

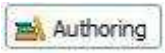
Install and Configure CVS
Create a New View with CVS
Add New Libraries to CVS[257]
Add a Method Plug-In to CVS[258]
Add Elements to CVS*Error! Bookmark not defined.*
Edit Elements under CVS[259]
Move Elements under CVS

11.1.7. Edit Elements under CVS

You can use CVS to track changes that happen to the content and relationships of method and process elements. You can use CVS to track who changed what, when, and why.

These steps apply to elements such as roles, tasks, work products, guidance, categories, configurations, capability patterns, and delivery processes.

To track changes with CVS:

1. Make sure that you are using the **Authoring** perspective 

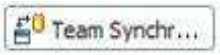
.
2. Use the tree browser to choose the element that you want to edit.
3. Double-click the element to open it in the editor window.
4. In the *Send an Edit Notification to CVS window*, click **Yes** to lock the needed files. If a file is currently being edited by another user, a window opens with the option to continue or not. To avoid needing to merge the files later, click **No** and stop editing that element.
5. In the **Editor** window, edit the element content. An asterisk (*) is next to the element's name at the top of the editor window while it is being edited. This indicates that the version cached in memory is out-of-synch with the local file system.

---

[257] You can add a method library to CVS so that you can control the version of various files in your library and allow multiple content authors to work in parallel.

[258] You can add method and process elements to CVS so that you can control changes to each element as its content is written and its relationships are defined.

[259] You can use CVS to track changes that happen to the content and relationships of method and process elements. You can use CVS to track who changed what, when, and why.

6. Click **File** → **Save**. The copy in the local file system is updated and the asterisk (*) is cleared.
7. Switch to the **Team Synchronising** perspective 

.
8. Ensure that the element that you edited in Step 5 is listed with other dependent files.
9. Select the top-level folder for your project and click **Commit**.

Related topics

Install and Configure CVS
Create a New View with CVS
Add New Libraries to CVS[260]
Add a Method Plug-In to CVS[261]
Add Elements to CVS*Error! Bookmark not defined.*
Delete Elements under CVS[262]
Move Elements under CVS

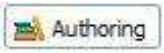
11.1.8. Move elements under CVS

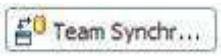
You can use CVS to move a method or process element in the library. The version control system makes changes to the appropriate files.

Before moving a method element, make sure that the element and any files that depend on it are committed.

These steps apply to elements such as roles, tasks, work products, guidance, categories, configurations, capability patterns, and delivery processes.

To move method and process elements under CVS:

1. Open the Authoring perspective. 

2. Use the tree browser to choose the element that you want to move.
3. Right-click the element and select Move.
4. In the Send an Edit Notification to CVS window, click Yes to lock the needed files. If a file is currently being edited by another user, a window opens with the option to continue or not. To avoid needing to merge the files later, click No and stop moving that element.
5. In the Move window, select the destination for the element. Click OK.
6. If you move the element to a different plug-in, the window in Step 4 opens and prompts you to send a notification to the CVS server. Click Yes.
7. Click File > Save.
8. Switch to the Team Synchronizing perspective. 

[260] You can add a method library to CVS so that you can control the version of various files in your library and allow multiple content authors to work in parallel.

[261] You can add method and process elements to CVS so that you can control changes to each element as its content is written and its relationships are defined.

[262] You can use CVS to ensure that when you delete a method or process element, the dependent files are appropriately version-controlled.

9. Make sure that the element that you moved in Step 5 is listed with the other dependent files.
10. Select the top-level folder for your project and click Commit.

Related topics

Install and Configure CVS
Create a New View with CVS
Add New Libraries to CVS[263]
Add a Method Plug-In to CVS[264]
Add Elements to CVS*Error! Bookmark not defined.*
Delete Elements under CVS[265]
Edit Elements under CVS[266]

11.2. Version Control Reference

Contents

11.2.1. Version Control for Specific Files

The following table describes the function of specific files and the implications for making changes to these files in a version-controlled environment.

Table 10 - Function of files impacted by version control systems

File	Usage
library.xmi	▪ The file *library.xmi* contains a reference to every plug-in in your method library. ▪ When you create new packages or processes, EPF Composer does not modify this file. This file is modified only when a plug-in is created. ▪ This file is shared by all plug-ins in your method library, so you need to be especially careful to coordinate changes being made by different individuals.

[263] You can add a method library to CVS so that you can control the version of various files in your library and allow multiple content authors to work in parallel.

[264] You can add method and process elements to CVS so that you can control changes to each element as its content is written and its relationships are defined.

[265] You can use CVS to ensure that when you delete a method or process element, the dependent files are appropriately version-controlled.

[266] You can use CVS to track changes that happen to the content and relationships of method and process elements. You can use CVS to track who changed what, when, and why.

File	Usage
plugin.xmi	▪ There is one *plugin.xmi* file for each plug-in. ▪ This file contains a reference to every content element in the plug-in, in addition to the presentation names, brief descriptions, and relationships for each element, such as guidance, inputs or outputs (tasks), performing role (tasks), work products responsible for (role), and so on. ▪ It also contains the definition of what is included in each standard category in addition to the definition and contents of each custom category. ▪ When more than one person works on a plug-in, you need to be especially careful about coordinating changes to this file and any files that it references.
method content element.xmi	▪ There is one file per method content element. ▪ It contains the descriptive text for the method content element for all fields except the name, presentation name, and brief description. ▪ The **file name** (<*method content element*>) is the same as the '**name**' field for the element. *(One exception to this: if you create a second method content element with the same name and that element is located in the same directory in your library, EPF Composer will append '2' to the end of the actual file name. The appended '2' will not appear in the name field.)*
model.xmi	▪ There is one *model.xmi* file for each **process** (capability pattern or delivery process). ▪ This file contains a reference to the descriptors (role, task, and work product), in addition to the names and brief descriptions for each of them. ▪ However, the brief description and presentation name for the process itself is kept in the *plugin.xmi*.
content.xmi	▪ There is one *content.xmi* file for each process (capability pattern or delivery process). ▪ This file contains the descriptive text for the descriptors of the process elements, with the exception of the names, presentation names, and brief descriptions.
configuration.xmi	▪ There is one *configuration.xmi* file for each configuration. ▪ Files of this type reside in the **\configurations** folder. ▪ They specify what is included in each configuration, including which content packages, which process packages, and which processes (capability patterns and delivery processes) are included, as well as the specification of what is included in each view.
.lock	▪ This is a temporary file located at the same level as *library.xmi*. It should not be placed under version control.
.project	▪ This is a temporary file located at the same level as *library.xmi*. It should not be placed under version control.

File	Usage
models.xmi	▪ There is one *models.xmi* file for each method library. ▪ The file contains references to all estimating models and estimating factors in the method library. ▪ The *models.xmi* file resides under folder called estimation. ▪ This file is modified when a new estimating model is created, an existing estimating model is renamed, a new default estimating factor is created, or one of the default estimating factors is modified. ▪ This file is created in the second step of the estimating model wizard when a user clicks **Next**. ▪ Rational ClearCase prompts to "add to source" two elements, a folder named estimation and the models.xmi file. Select the *Keep checkout option* and click **OK** to add to source control. ▪ This file is shared by all estimating models in the library, so care must be taken to coordinate changes made by anyone who uses it.
diagram.xmi	▪ All diagrams are stored in this file.

Related topics

Integration with Rational ClearCase[267]

11.2.2. Common Actions Impact on Specific Files

Changes made to method elements often affect more than one file. In multiuser environments it is important to understand which actions will affect multiple files and act accordingly.

For more information about actions by multiple users, see Multiuser scenarios with IBM Rational ClearCase[268].

It is a good practice to have all file updates to a library under version control, be performed with the integrated version control support provided by EPF Composer. This integration protects users from unexpected file conflicts when performing actions that involve multiple files.

For example, if a local file is out-of-date relative to the server, EPF Composer will detect this conflict and generate a warning. In this situation users should heed the warnings and update their library appropriately.

The safest way to ensure that files with cross-dependencies are kept in sync with one another is to update the entire library. Do not simply refresh the individual element that is out of sync.

The following table describes the impact of common actions on specific files managed by a version control system.

[267] IBM Rational ClearCase can be integrated with EPF Composer, if a properly configured Rational ClearCase server is available.

[268] Precautions are needed when multiple users attempt to modify the same files at the same time.

Table 11 - Impact of common actions on files managed by Rational ClearCase

Action	Impact	Comments
Create a new plug-in.	▪ A directory is created and added to version control. ▪ A plugin.xmi is created in the new plug-in directory and added to version control. ▪ The library.xmi file is changed.	▪ Need to place new directory and new plugin.xmi file under version control. ▪ Be sure to check in the new plugin.xmi before checking in library.xmi.
Delete a plug-in.	▪ The entire directory for the plug-in is deleted. ▪ The library.xmi file is changed.	▪
Rename a plug-in.	▪ The name of the plug-in directory is changed. ▪ The library.xmi file is changed.	▪ Check-in these changes as close together as possible, checking in the directory name change first, followed by library.xmi.
Move a content element to another plug-in.	▪ File moves from one directory to another. ▪ The plugin.xmi file for both plug-ins changes.	▪ Check-in all changes as close together as possible, checking-in the plugin.xmi file last.
Create a new method content element (*task, role, work product, guidance, custom categories, or standard category*).	▪ A new method content.xmi file is created and added to version control if one or more rich text fields of the element are populated. ▪ The library.xmi file is changed.	▪ Need to place the new file under version control before checking in plugin.xmi. ▪ Note: Each method content.xmi is placed in a specific folder for each content type. For example, method content.xmi for role is placed in **\roles** folder.
Change the name of a method content element (*task, role, work product, guidance, custom category, or standard category*).	▪ The <method content>.xmi file name is changed. ▪ The library.xmi file is changed.	▪ "Name" is the Name: field for the element. Changing this field changes the file name of the element. ▪ Check in both changes as closely together as possible.
Delete a method content element *(task, role, work product, guidance, custom categories, or standard category*).	▪ The method content.xmi file is deleted. ▪ The library.xmi file is changed.	▪

Action	Impact	Comments
Create a new process (capability pattern or delivery process).	▪ The plugin.xmi file is modified. ▪ New directory and new model.xmi and content.xmi files are created for the new CP/DP and added to version control.	▪ The plugin.xmi file is updated to include a reference to the new capability pattern or delivery process. Be sure to place the new files under version control before checking in plugin.xmi. ▪ Note: It is advisable to double-check that your new process is selected in your configuration. If it is not, you will not see it in the Configuration View.
Delete a process (*Capability pattern or delivery process*).	▪ The plugin.xmi file is modified. ▪ The directory for the process and its contents are deleted.	▪
Rename a process (*Capability pattern or delivery process*).	▪ The plugin.xmi is modified. ▪ The directory for the process is renamed.	▪ Check in these changes as close together as possible.
Create, rename, or delete content package or process package.	▪ The plugin.xmi file is modified. ▪ Note: Content and process packages do not result in new directories being created in the file structure.	▪ When you create a new package, you may want to select it in your configuration. This will cause \configurations\configuration_name.xmi to be modified.
Change the name of a process element (*Capability pattern or delivery process)*	▪ The directory name for your CP/DP is changed to match the new name. ▪ The plugin.xmi file is changed to reference the new directory name.	▪
Change the Presentation Name of a method element or process.	▪ The plugin.xmi file is modified	▪
Refine the descriptive text for a method element (all fields except the name, presentation name, or brief description).	▪ The method content.xmi file is updated	▪

Action	Impact	Comments
Create a new custom category.	▪ The plugin.xmi file is modified.	▪
Update the brief description for a method content element (*task, role, work product, guidance, custom categories, standard category, or process (Capability pattern or delivery process))*	▪ The plugin.xmi file is modified.	▪
Assign a relationship to an element, *such as adding guidance, tool mentors, input or output work product, responsible role, performing role, and so on.*	▪ The plugin.xmi file is modified.	▪
Categorise an element by assigning it to a domain, work product kind, role set, or discipline.	▪ The plugin.xmi file is modified.	▪
Assigning variability to an element by making it a contributor, extension, or replacement for another element.	▪ The plugin.xmi file is modified.	▪
Assigning a shape or node icon.	▪ The plugin.xmi file is modified.	▪
Add or change descriptive text for a process (not the brief description).	▪ The content.xmi file is modified.	▪
Update the brief description for an activity in a process (*Capability pattern or delivery process*).	▪ The model.xmi file is modified.	▪
Modify or create a diagram for a process.	▪ The model.xmi file is modified.	▪
Update the name or presentation name of an activity.	▪ The model.xmi file is modified.	▪
Set or update the entry or exit state of a work product.	▪ The model.xmi file is modified.	▪

Action	Impact	Comments
Set or update activity relationships (predecessors, extend, copy).	▪ The model.xmi file is modified.	▪
Create a configuration.	▪ The configuration name.xmi file is created under \configurations folder.	▪ Be sure to add \configurations \configuration name.xmi to source control.
Delete a configuration.	▪ The configuration name.xmi file is deleted.	▪ Be sure to delete \configurations \configuration name.xmi in your version control system.
Rename a configuration.	▪ The configuration name.xmi file is renamed to the new configuration name.xmi.	▪
Modify a configuration.	▪ The configuration name.xmi file is modified	▪

Related topics

Version Control for Specific Files
Installing and Configuring CVS[269]

11.3. Using Rational ClearCase

IBM® Rational® ClearCase® can be integrated with EPF Composer, if a properly configured Rational ClearCase server is available.

The Rational ClearCase integration simplifies the task of managing a method library under version control. After installing the appropriate version control adapters, the authoring perspective is supplemented with the following features:

- Automatic prompts to place new method library files under source control.
- Ability to identify and check out all files needed to complete a particular operation. This is significant because the set of files impacted by an operation are not obvious. Sometimes more than one file is involved. For more information about version control, see Version Control for Specific Files.
- Simplified procedures for renaming and deleting files. For more information about working with files, see Deleting Files with ClearCase.

Consult the EPF Composer Help files for more information.

[269] CVS assumes an optimistic usage model in which multiple users make changes to files simultaneously. However, because compare and merge operations are not supported, the CVS Watch/Edit function should be used to broadcast notifications to other users who may be working on files concurrently.

12. Appendix

Contents

12.1. Keyboard Shortcuts

For a complete list of standard keyboard shortcuts in Windows, see the Keyboard Assistance information from Microsoft:

http://www.microsoft.com/enable/products/keyboard.aspx

All of main menu commands in EPF Composer have keyboard shortcuts. A list of these shortcuts can be displayed by using the Help menu and selecting Key Assist, or by using the shortcut **Ctrl+Shift+L**.

To invoke the context menu while using the process editor, press **Shift+F10**. This menu allows you to open or create diagrams.

To invoke the context menu while using the diagram editor, press **Shift+F10**. This menu allows you to create diagram objects such as activities, task descriptors, and so on.

Keyboard Navigation

EPF Composer uses standard Microsoft Windows navigation keys. A list of common keyboard shortcuts is shown below:

- To traverse to the next link, button, or topic, press **Tab** inside a frame page.
- To go to the next link, button or topic node from inside a frame or page, press **Tab**.
- To expand and collapse a tree node, press the **Right** and **Left** arrows.
- To move to the next topic node, press the **Down** arrow or **Tab**.
- To move to the previous topic node, press the **Up** arrow or Shift+Tab.
- To scroll all the way up or down, press **Home** or **End**.
- To go back, press **Alt+Left** arrow; to go forward press **Alt+Right** arrow.
- To go to the next frame, press **Ctrl+Tab**.
- To move to previous frame, press **Shift+Ctrl+Tab**.
- To print the current page or active frame, press **Ctrl+P**.
- To move objects around, press the **CTRL+Right** or the **CTRL+Left** arrow keys.

Authoring Contents

You can use the keyboard to navigate the Authoring perspective by using the following key combinations:

- To expand or collapse a Library tree node, press the **Right** or **Left Arrow**.
- To move to the next topic node, press the **Up Arrow** or **Down Arrow**.
- To create a method element, move to the element node and press the **Application key**. If your keyboard does not have an Application key, refer to information from your operating system vendor on adapting keyboards.
- Press the **Up Arrow** or **Down Arrow** to navigate between menus; press **Enter** to make the choice take effect.
- Press **Ctrl+Tab** to navigate between different sections of the element.
- **Ctrl+PageUP/PageDown** to navigate between different tabs.

Element Editor

You can use the keyboard to navigate in the rich text editing view for an element by using the following key combinations:

- Press **Enter** to open the editing view.
- Press **Tab** to navigate between sections of the editing view.
- Press **Shift+Left Arrow** or **Shift+Right Arrow** to choose text.
- Press **Ctrl+PageUp** or **PageDown** to switch between the Rich Text and HTML tabs.

Process Editor

You can use the keyboard to navigate in the process editor by using the following key combinations:

- Press **Ctrl+Tab** to navigate between different sections of a process element.
- Pres **Ctrl+Page Up** or **Ctrl+Page Down** to switch between different tabs of a process element.
- Press the **Application key** to open a menu. If your keyboard does not have an Application key, refer to information from your operating system vendor on adapting keyboards.
- Press **Escape** to navigate between different level menus.
- Press **Up Arrow** or **Down Arrow** to navigate between lines.
- To expand or collapse a tree node, press **Right** or **Left Arrows**.
- To navigate to the property tab for a process, use the **Application key** and choose the **Show Properties View** menu.
- In the Properties tab for a process, press **Tab + Up Arrow** or **Down Arrow** to navigate between different sections like **General** and **Documentation**.
- Press the space key to check or clear a checkbox in **Properties** tab.

Diagram Editor

- To open a diagram editor of a process, press the **Application key**. If your keyboard does not have an Application key, refer to information from your operating system vendor on adapting keyboards.
- In the opened diagram, press the **Application key**, from which you can add any elements and nodes.
- Press **Escape** to navigate between different level menus.
- Press **Tab** to navigate to the **Palette** tab and use **Up Arrow** or **Down Arrow** to navigate.
- To move around an element in a diagram, press **Ctrl+Up/Down/Left/Right**.
- Press **Up/Down/Left/Right key** to navigate between different elements.
- To navigate to **Properties** tab for an element, use the **Application key** and choose the **Show Properties View** menu.
- In the **Properties** tab for a diagram, use **Tab and Up Arrow** or **Down Arrow** to navigate between different sections.

Publishing

- In the Publish Wizard, press **Up Arrow** or **Down Arrow** to switch between publish format. Press **Tab** to navigate between buttons.
- In the third window of the Publish Wizard, press **Up Arrow or Down Arrow** and the **Space key** to check or clear tree items.
- In the fourth window of the Publish Wizard, press the **Space key** to check or clear a checkbox. Use **Up Arrow** or **Down Arrow** to navigate checklists.

12.2. Preferences

Set preferences to define default application behaviours. To access the Preferences window, click **Window** → **Preferences**. Change a setting in the Preferences and click Apply. Click OK to set the preference and close the window. To restore the preferences to the default settings, click Restore Defaults.

12.2.1. Method Parameters

The following table describes each preference in the Method section that can impact EPF Composer.

Table 12 - EPF Composer preferences

Setting	Description
Method > Authoring	Use this page to set the default path for the method library. You can also set the following preferences: ▪ Discard unresolved references (off by default): The option can help you clean up your library and remove errors that are reported in the Problems view. However, you should analyse why these references were broken and be clear about your intention to remove them before you select this

Setting	Description
	option. ■ Use new Extends semantics (off by default): If the option is not selected, Extends variability works as described in Method Content Variability[270] . If the option is selected, semantic rules for Extends are available as specified in the OMG finalised SPEM 2.0 specification (Software and Systems Process Engineering Metamodel). See http://www.omg.org/issues/spem2-ftf.open.html#Issue11284 and http://www.omg.org/spec/SPEM/2.0/ for details. ■ History size for previous path choices (default is 10) ■ Enable method library validation (off by default): This option enables some additional commands in the EPF Composer user interface (for example, the Validate button in the Library view's toolbar as well as the context menu) that allow you to run validation operations on the method library for finding inconsistencies. ■ Show External ID fields in editors (off by default): When selected, one extra name field will be displayed in element's form editor called External ID. The content of this field will be published as a second name with every element where it is applied to.
Method > Authoring > Library View Options	Use this page to choose the default element link type when dragging an element from the Library or Configuration view into a rich text editor field. The choices are: ■ Method element (default) ■ Method element with type prefix ■ Method element with custom text
Method > Authoring > Process Editor	Use this page to show or hide columns in the Work Breakdown Structure, Team Allocation, and Work Product Usage tabs of the Process editor. ■ Use the menu, and the Add, Remove, Up and Down buttons to change the layout of all process editor tabs. These changes also impact how the tree structure is published to HTML. Only the columns that you select will be published in the order that you specify here. ■ You can also set the preference for switching to the process's default configuration when activating a Process editor. The default is to prompt the user. ■ Inherit suppression states (on by default): When applying a capability pattern to a process, this option decides if the

[270] Method content variability allows elements in one content package to modify or reuse elements in other content packages without directly modifying the original content. Variability provides a mechanism for making changes to the published Web site while keeping the components separate and optional.

Setting	Description
	applied pattern should have the same elements suppressed as the original pattern or if all element suppressions should be removed. Elements can be suppressed via the Properties view and through the context menu operations.
Method > Authoring > Process Editor > Breakdown Element Attributes	Use this page to specify the default values for some of the attributes of process elements created in the Process editor. The following attributes are available for selection for Phase, Iteration, Activity, and Milestone: ▪ Optional (off be default) ▪ Multiple Occurrences (off by default) ▪ Planned (on by default) ▪ Event-Driven (off by default) ▪ Ongoing (off by default) ▪ Repeatable (off by default) The following attributes are available for selection for Role, Task, and Work Product Descriptors: ▪ Optional (off be default) ▪ Multiple Occurrences (off by default) ▪ Planned (off by default) ▪ Event-Driven (off by default) ▪ Ongoing (off by default) ▪ Repeatable (off by default)
Method > Publishing/Browsing	Use this page the set the following preferences: ▪ Default path: Set the default path for publishing Web sites. ▪ Feedback URL: Set the default URL for Feedback that should be used for a new method configuration. ▪ Include method content in descriptor pages: This option defines how the browsing perspective presents descriptor pages (default is false). See Publishing Configurations as Web Sites *Error! Bookmark not defined.* for details.
Method > Publishing/Browsing > Activity Diagrams	Use this page to set the preferences for the following activity diagrams and activity detail diagrams: ▪ Publish activity diagrams for unmodified activity extensions (off by default) ▪ Publish activity detail diagrams that have not been created in process editor (off by default) ▪ Number of tasks per row (default is 10) See Publishing Configurations as Web Sites[271] for details.

[271] You can use the Publish Method Configuration wizard to generate a Web site based on a method configuration.

Setting	Description
Method > Publishing/Browsing > Role Diagram	The parameters of this page define the layout of the Role Diagram that is published with a role page.
Method > Publishing/Browsing > Supporting Plug-in	Supporting method plug-ins are method plug-ins which content will only be published if it is referenced by another method plug-in. ▪ Include descriptor relationships to linked elements in supporting plug-ins (off by default): This option defines an exception of this rule for relationship from a descriptor to a method content element. The default is to not publish such elements.

12.2.2. Alternate Help Browser

This procedure describes how to use a speech-enabled browser to read the online help.

Pages in the Help system are normally displayed through an internal browser included with the application. This internal browser can be deselected, which causes the default system browser to be used to display Help pages.

To set alternate help browser:

1. Select **Windows** → **Preferences**.
2. Click **Help**.
3. Select **Use External Browser**.

12.2.3. Fonts

The following table lists the four main fonts that the Eclipse workbench uses:

Table 13 - Fonts used

Font	Description
Banner	▪ Used in Plug-in Development Environment (PDE) editors, on welcome pages, and in the title area of many wizards. ▪ For example, the New Project wizard uses this font in the top title.
Header	▪ Used as a section heading. ▪ For example, the welcome page for the Eclipse platform uses this font for the top title.
Text	▪ Used in text editors.
Dialog	▪ Used in dialog boxes.

To set these fonts, click **Window** → **Preferences** → **General** → **Appearance** → **Colours** and **Fonts**. Several other secondary font settings are also available on the Colours and Fonts preference page.

12.2.4. Accessibility

The appearance of the **caret** used in the text editor can be changed by setting preferences on the **Accessibility** preferences panel. To set the accessibility preferences:

1. Select **Windows** → **Preferences**.
2. Expand the tree node for **General** → **Editors** → **Text Editors** and select **Accessibility**.
3. Set the following preferences:

Option	Description
Use custom caret	▪ This option replaces the original caret with a custom caret and shows a different caret for Overwrite and Insert mode. ▪ Default is on.
Enable thick caret	▪ This option replaces the original caret with a more visible, thicker caret. ▪ Default is on.
Use characters to show changes in vertical ruler	▪ Quick Diff shows the changes in a vertical ruler using colours. ▪ Colour-blind persons can enable this option to show the differences with different characters in the line number ruler. ▪ Default is off.

12.2.5. Accessibility Features

EPF Composer is based on Eclipse and offers several accessibility features, which are part of the Eclipse development platform. Those features are summarised as below:

- User interface elements compatible with assistive technology are rendered using Microsoft® Active Accessibility (MSAA) APIs.
- All features can be operated using the keyboard instead of the mouse.
- Screen-reader software such as Freedom Scientific's JAWS can be used to hear what is displayed on the screen.
- Voice recognition software, such as IBM® ViaVoice® can be used to enter data and to navigate the user interface.
- Screen can be magnified in the graphical views.
- Fonts and colours defined by Eclipse can be set using the Window menu and selecting Preferences.

Note: Accessibility features mentioned in this document apply to the Windows® operating system.

Accessibility of Published Configurations

All content in the method library is accessible. Published configurations based on this content will also be accessible with some exceptions noted below. All diagrams and icons have HTML "ALT" attributes, which means they have text-based descriptions that can be recognised and spoken by text-to-speech converters, including speech-enabled web browsers. Obviously there is less information contained in the textual descriptions

than in the diagrams; however a reasonable effort has been made to accommodate visually impaired users.

All data tables have row and column titles compatible with speech-enabled web browsers. Tables that are merely used to align page elements (layout tables) do not use row or column headers.

New method content created or imported into EPF Composer may not have the required HTML tags to be compatible with speech-enabled browsers. Authors of new content should use the HTML editing mode in the Rich Text Editor to manually insert appropriate ALT attributes next to "IMG SRC" tags.

12.3. EPF Composer User Roles and Tasks

There are four primary user roles of the EPF Composer:

- Method Author
- Process Author
- Process Configurator
- Practitioner

Method Author

The Method Author uses the tool on a regular basis to provide standard processes for use in an organisation. The Method Author uses the full functionality of the tool to:

- Create plug-ins.
- Create new method elements.
- Extend existing method elements.
- Create reusable capability patterns by reusing method elements.
- Create delivery processes by reusing capability patterns and method elements.
- Create custom categories for use as views in a configuration.
- Create and modify configurations.
- Publish configurations or processes.

Process Author

The Process Author's goal is to produce a delivery process for their projects by reusing method elements. The Process Author uses the tool occasionally, as project needs dictate, typically supporting one or, more likely, several projects by specifying the processes to be followed. The Process Author uses the process authoring and configuration publishing functionality of this tool to:

- Create plug-ins.
- Create reusable capability patterns by reusing method elements.
- Create delivery processes by reusing capability patterns and method elements.
- Create custom categories for use as views in a configuration.
- Create and modify configurations.
- Publish configurations or processes.

Process Configurator

The Process Configurator's goal is to produce a delivery process for their projects by rapidly leveraging ready-made plug-ins. The Process Configurator uses this tool occasionally, as project needs dictate, typically supporting one or several projects by specifying the process for the projects. The Process Configurator uses the configuration publishing functionality in this tool to:

- Create and modify configurations.
- Publish configurations or processes.

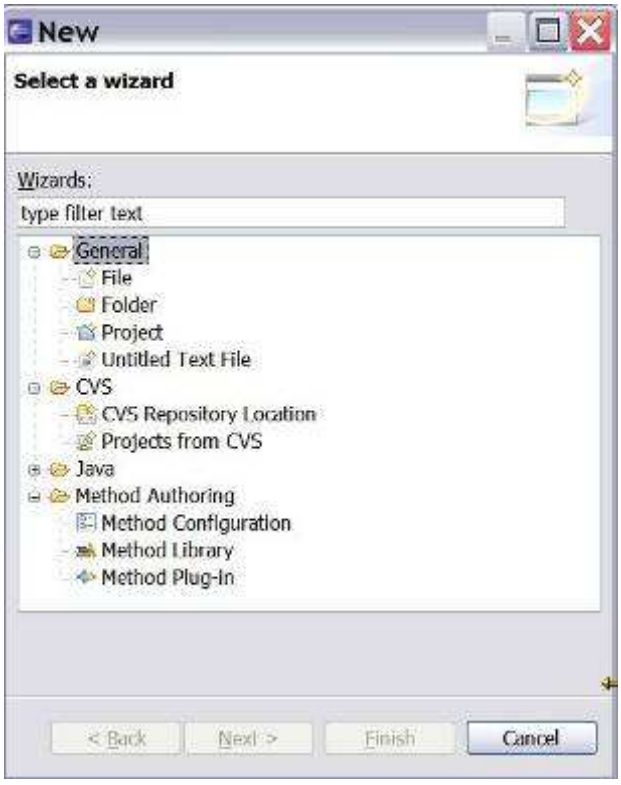
Practitioner

A Practitioner's goal is to use correctly the organisation's processes and best practices effectively. A Practitioner uses a published configuration on a regular basis driven by the work being performed to view processes and methods.

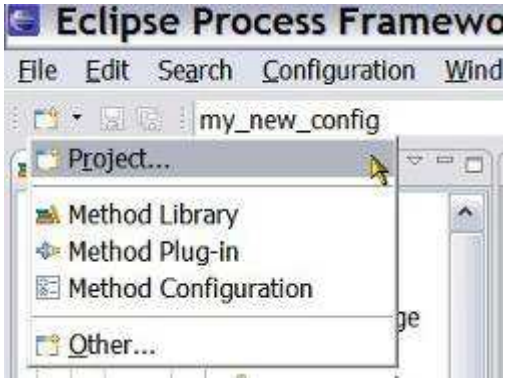
12.4. Open Questions

12.4.1. New Project - Etc.

What effects have the Wizards in General: New File, Folder, Project or Untitled Text File?



In particular, New Project:

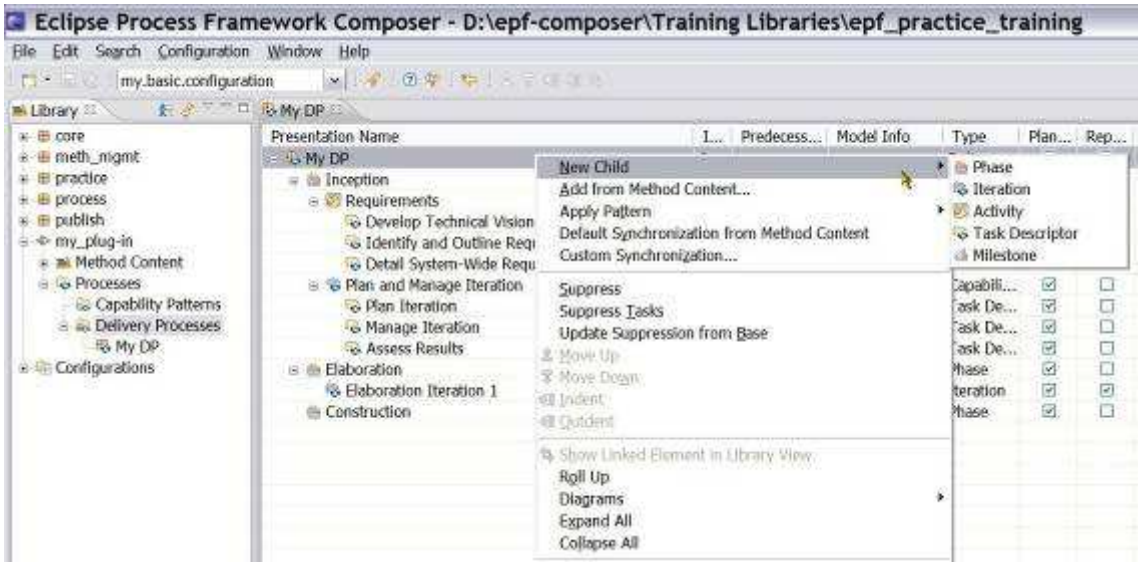


12.4.2. Phase, Iteration, Activity

Some explanation of how the terms: "Phase, Iteration and Activity" relate to each other would be appropriate.

Typically, a process is created by defining its phases and iterations within or across these phases. Phases and iterations are then further broken down into levels of activities. Finally, you can populate a work breakdown structure's activity with task descriptors.

Based upon the tutorials:



It appears that anything can nest under anything and it is not clear why there are three different terms, which all seem to indicate a period, a duration but with different connotations.

In would seem, a delivery process may have N Phases, which may go through Y iterations: Each iteration goes through all N Phases. In that case, one iteration includes all the phases and the Delivery Process with the N Phases are like the type and an iteration is the instantiation of a type. Activity is a subset of a phase. The sentence that the delivery processes is reiterated would then be clearly defined.

TOGAF seems to support this: "These ADM iteration cycles are intended to span multiple phases of activity and allow formal review upon completion of each multi-phase iteration cycle."

*Iterations, Phases and Activities are **Activity Types** and each type can be transformed into another type using the Properties View.*

13. Glossary

Activity

Activities are the main building blocks for processes. An activity is a collection of work breakdown elements such as task descriptors, role descriptors, work product descriptors, and milestone. Activities can include other activities.

Activities can be presented in work breakdown structures and activity diagrams that graphically describe the flow of work by showing which activities precede other activities. Phase and iteration are special types of activities that define specific properties.

Artefact

An artefact is a tangible work product that is consumed, produced, or modified by one or more tasks. Artefacts may be composed of other artefacts. For example, a model artefact can be composed of model elements, which are also artefacts.

Roles use artefacts to perform tasks and to produce other artefacts. Each artefact is the responsibility of a single role, making responsibility easy to identify and understand, and promoting the idea that every piece of information produced in a method requires the appropriate set of skills. Even though only one role is responsible for an artefact, other roles may use the artefacts.

Checklist

A checklist is a specific type of guidance that identifies a series of items that need to be completed or verified. Checklists are often used in reviews such as a walkthroughs or inspections.

Concept

A concept is a specific type of guidance that outlines key ideas associated with basic principles underlying the referenced item. Concepts normally address more general topics than guidelines and may be applicable to several work products, tasks, and activities.

Deep copy

Deep copy is a mechanism for copying all inherited references from one activity to another. Deep copy will resolve all references at all levels of extension. For example, if an activity contains an applied extended capability pattern which itself contains another capability pattern, a deep copy would create local copies for all levels of nesting. A normal copy would only copy the first level of nesting.

Deliverable

A deliverable is a collection of work products, usually artefacts. Deliverables are used to define typical or recommended content in the form of work products packaged for delivery. Deliverables are also used to represent an output from a process that has value, material or otherwise, to a client, customer, or other stakeholder.

Delivery process

A delivery process is the process that covers a whole development lifecycle from beginning to end. A delivery process can be used as a template for planning and running a project. It provides a complete lifecycle model with predefined phases, iterations, and activities.

Discipline

A discipline is a categorisation of tasks that are related to a major area of concern and cooperation of work effort. For example, on a software development project, it is common to perform certain requirements tasks in close coordination with analysis and design tasks. Separating these tasks into separate disciplines makes the tasks easier to comprehend. Disciplines can be organised using discipline groupings.

Domain

A domain is a hierarchy of related work products grouped together based on timing, resources, or relationship. While a domain categorises many work products, a work product belongs to only one domain. Domains can be further divided into sub-domains.

Estimating considerations

Estimating considerations are a specific type of guidance which provide sizing measures or standards for sizing the work effort associated with performing a particular piece of work and instructions for their successful use.

Estimating guideline

An estimating guideline is a specific type of guidance that provides sizing measures, or standards for sizing the work effort associated with performing a particular piece of work and instructions for their successful use. It may be comprised of estimation considerations and estimation metrics.

Example

An example is a specific type of guidance that describes a representative instance of a completed work product.

Guideline

A guideline is a specific type of guidance that provides additional information on how to perform a particular task or set of related tasks. Guidelines may provide additional details, rules, and recommendations on work products and their properties. They can describe best practices and different approaches for doing work.

Iteration

An iteration is a group of activities that are repeated more than once. Iterations are used to organise work into repetitive cycles.

Library view

The library view shows all method plug-ins and configurations in a method library. The library view is available only in the authoring perspective.

Milestone

A Milestone describes a significant event in a project, such as a major decision, completion of a deliverable, or meeting of a major dependency such as the completion of a project phase.

Outcome

An outcome is an intangible work product that may be a result or state. Outcomes may also be used to describe work products that are not formally defined.

Phase

A phase is a type of activity that represents a significant period in a project. Phases typically conclude with a management checkpoint, milestone or set of deliverable artefacts.

Practice

A practice represents a proven way or strategy of doing work to achieve a goal that has a positive impact on work product or process quality. Practices are defined orthogonal to methods and processes. They could summarise aspects that impact many different parts of a method or specific processes. Examples for practices would be manage risks, continuously verify quality, architecture centric, and component based development, to name a few.

Preview

Preview displays method content in browser format similar to how it will appear to an end user browsing a published web site.

Report

A report is a predefined template of a result that is generated on the basis of other work products as an output from some form of tool automation. For example, a report may combine a graphical model from a design tool with textual information documents.

Reusable asset

A reusable asset provides a solution to a problem for a given context. The asset has rules for usage which are the instructions describing how the asset should be used.

Roadmap

A roadmap is a specific type of guidance that describes how a process is typically performed. Often processes can be much easier understood by providing a walkthrough of a typical instance of the process. In addition to making the process practitioner understand how work in the process is being performed, a roadmap provides additional information about how activities and tasks relate to each other over time.

Role

A role is a well-defined set of related skills, competencies, and responsibilities. Roles can be filled by one person or multiple people. One person may fill several roles. Roles perform tasks.

Role set

A role set is used to group roles with certain commonalities together. For example, in a software development environment, an Analyst role set could be used to group together roles such as Business Process Analyst, System Analyst and Requirements Specifier. Each of these roles work with similar techniques and have overlapping skills, but may be responsible for performing certain tasks and creating certain work products. Role sets can be organised using role set groupings.

Role set grouping

Role sets can be categorised into role set groupings. For example, different methods might define similar role sets which need to be distinguished from each other on a global scale.

Step

A step is a part of the overall work described for a task. The collection of steps defined for a task represents all the work that should be considered to achieve the overall goal of the task. Not all steps are necessarily performed each time a task is invoked in a process. Steps are generally unordered and can be performed in any order.

Supporting material

Supporting material is a generic type of guidance containing information not specifically covered by the other guidance types.

Synchronisation

Synchronisation is a mechanism whereby changes to information in method elements can be automatically updated in related process elements. For example, if a name of a method content element is changed, the new name will be displayed in all processes that use that method element.

There are two types of synchronisation:

- Custom synchronisation at the activity level will update descriptors in activities by bringing in task descriptor's associations.
- Default synchronisation at the activity level will update activities by bringing in task descriptor's associations.

Task

A task is an assignable unit of work. Every task is assigned to a specific role. The duration of a task is generally a few hours to a few days. Tasks usually generate one or more work products.

Template

A template is a specific type of guidance that provides a work product with a predefined table of contents, sections, packages, and headings. Templates provide a standardised format, as well as descriptions of how the sections and packages are supposed to be used and completed. Templates can be provided for documents as well as conceptual models or physical data stores.

Term definition

Term definitions define specific terms, concepts, or other ideas relevant to method and process content. A term definition is not directly related to any content elements, but relationships are derived when the term is used in the description text in a content element.

Tool mentor

A tool mentor is a type of guidance that shows how to use a specific software application to accomplish a piece of work.

White paper

A white paper is a special type of guidance, which includes reports and recommendations. White papers can be read and understood as independent documents, in isolation of other method elements and guidance.

Work product

Work product is a general term for task inputs and outputs, descriptions of content elements that are used to define anything used, produced, or modified by a task. The three types of work product are:

- Artefact
- Outcome
- Deliverable

Work product descriptor

A work product descriptor is a work product in the context of one specific activity. Every breakdown structure can define different relationships of work product descriptors to task descriptors and role descriptors. One work product can be represented by many work product descriptors, each within the context of an activity with its own set of relationships.

Work product kind

Work product kind is another category for grouping work products. A work product can belong to multiple work product kinds.

14. Document Management

14.1. Document Details

Fields	Content
Document #	
Document name	EPF Installation, Tutorial and Manual
Description	Document with installations instructions, a number of tutorials and a manual

14.2. Document History

Date	Version	Status	Author(s)	Comments
4 – Feb - 2010	0.14	Draft	BT	Section reviewed by GS

14.3. Author and Reviewers

Role	Name	Initials	Contact Info
Author	Bjorn Tuft	BT	bjorn.tuft@gmail.com
Reviewer	Gerhard Schneider	GS	gerhard.schneider@innoreg.com