OpenETCS Toolchain

- Documentation -

Overview

How to Contribute

Writing documentation is everybody's task. Don't ask for permission to contribute, just do it.

As of this writing (4-Mar-2014), the documentation is stored in a single Wiki page. To prevent editing conflicts, please follow these rules:

- Note that this page uses MediaWiki Markup (not gitHub wiki Markup)
- If you are missing documentation, please create a stub for it: Find the place where you think it would be appropriate, and insert a section name, preceded by TODO. That way we know that something is missing.
- If you want to contribute content to a section, then:
  - First edit the section name, by removing the TODO and replace it with your name, e.g. "Michael: How to Contribute". Then safe the page. That way, someone looking at the page (hopefully) knows not to edit in the same section.
  - Work on the section as you see fit.
  - When you're done, remove your name. However, if the content is not complete, put the word TODO back into the section name.
  - If you get a conflict upon safe, hopefully the conflict is in a different section. In that case, copy your contribution (which is hopefully limited to one section), reload the page, and insert your content.

Artwork

- Please make sure that artwork is correspondingly licensed. Here is a CC-licensed icon set that is used for tips, hints, warnings, etc.: http://omercetin.deviantart.com/art/PixeloPhilia-32PX-Icon-Set-157612627
- You need to upload artwork via git to /tool/documentation-artwork. Please use folders to organize the pictures.
- To use an image, find the raw URL and put it in double square brackets.

You can use the following icons:

Pointing out potential problems, please use sparingly.

Tips and Tricks

Further resources, especially for developers (link to code, link to specs, etc.

🏳️

Things that are flagged for later, e.g. marking things that will be implemented shortly

TODO License

Installation

System Requirements

The minimal requirement for java is Java 6 (aka Java 1.6).

Some functionalities (e.g. ProR "Rich Text" extension) won't work with Java 6 and require Java 7. However, core tools of openETCS tool chain (e.g. Papyrus SysML editor) have not been qualified for Java 7.

Installation procedure

- The openETCS Toolchain is available in the following URL for downloading: https://openetcs.ci.cloudbees.com/job/openETCS-tycho/28/artifact/tool/bundles/org.openetcs.releng.products/target/
- Unzip main folder to extract the content in the desired location.
- Launch openETCS executable file to run Eclipse.
- Select your workspace.
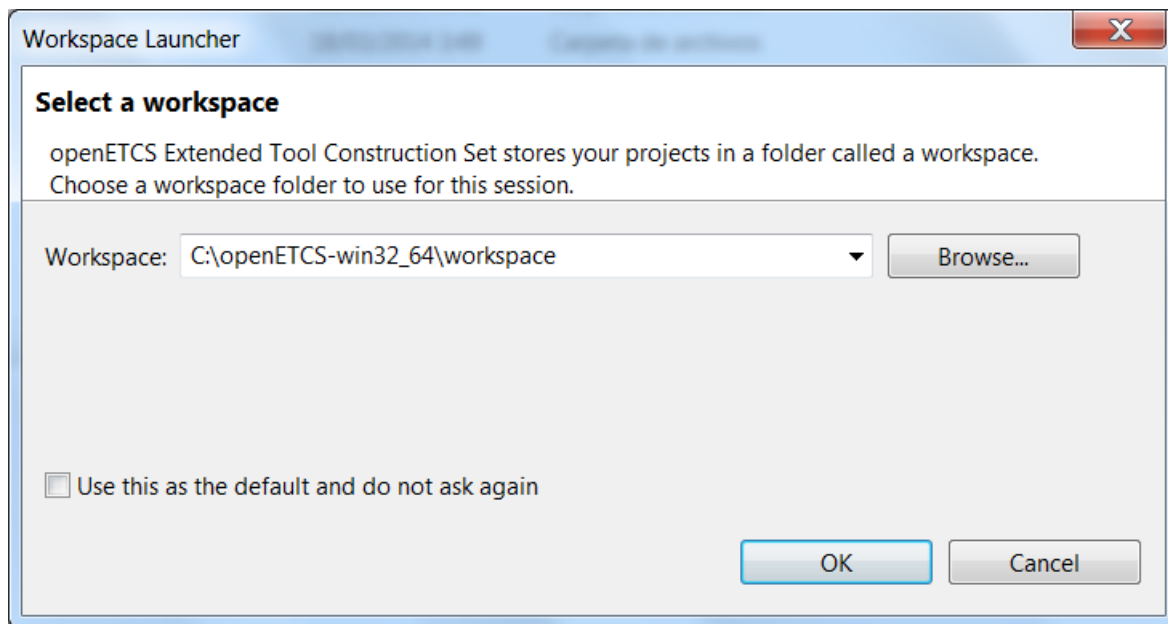
Installing ProR Essentials

There are a number of free (but currently not open) extensions that are useful for working with openETCS. As they are not open, they have to be installed manually. Currently this includes:

- **Suspect Links** - indicates if source or target of a link have changed.
- **Rich Text** - shows formatted text properly formatted, and allows editing with a rich text editor.

The **Rich Text** extension requires Java 7 from Oracle, as it relies on the JavaFX component.

Here are the steps:

- Select <tt>Help | Install new Software...</tt>
- You need to enter a new update site in the <tt>Works with</tt> box (if you entered it before, you can select it from the drop down): <tt>http://update.formalmind.com/essentials</tt>
- From the Category <tt>ProR Essentials for Eclipse RMF</tt> select:
  - <tt>ProR Linkmanagement Presentation Feature</tt>
  - <tt>ProR Xhtml Rtf Presentation Feature</tt>

Make sure you select version 0.10 or better. The update site structure will probably change in the future.

- Follow the installation wizard.
- The use of the feature is explained further down.

TODO Upgrade procedure

Getting-Started

TODO Toolchain basics

TODO Elements

the idea behind this requirements is to find a first reference to enter the tool chain. Like in short reference sheets or FAQs within this reference the elements which are of current interest (meaning which are to be touched in future use/ development) could have a short description to enable the partners to collaborate effectively within the tool chain.

One example could be listing for a plugin:

- Path it resides in
- Short description of usage/ functionality
- Definition of data for input
- Definition of data for output
- Elements of current interest:
  - here a list of parameters could reside
  - definition of API for other modules
  - naming other elements/ functionality which are expected to be touched

Including their internal data structure in case they are foreseen for use by the following tool chain

Bug report

Bug should be reported at this address: https://github.com/openETCS/toolchain/issues.

- Create a new issue: Just click on the new issue button

- Select the Bug report label

- Fill the title with a precise naming
- Fill the comment that describes more precisely your bug

If the bug is related to a tool that is developed outside openETCS project:

- A toolchain contributor/openETCs tool chain manager will assign this issue
- The assignee should forward the issue to the corresponding project
- The assignee should put a link in the bug report issue.

TODO Eclipse Basics

TODO Projects in Eclipse

-> TODO Address #12 from modeling repository here

Workspaces in Eclipse

Tutorial

User-Documentation

TODO Overview

Basic Tasks

Git Version Control

https://github.com/openETCS/toolchain/wiki/Introduction-to-Git

TODO Modeling Blocks (SysML)

Data Dictionary

This plugin contains the data dictionary plugin which contains data structures, variables, messages, etc. from the ETCS System Requirements Specification (SRS). The plugin registers a UML model, which contains the information from the SRS. After registration, the UML model is available as a UML library.

Plugins

The plugin consists of the following parts:

```
org.openetcs.datadictionary
org.openetcs.datadictionary.feature
```

Usage

To use the plugin, right click on the SysML model from the Papyrus model explorer and select <tt>Import | Import Registered Package</tt>. The select <tt>openETCSDataDictionary</tt>.
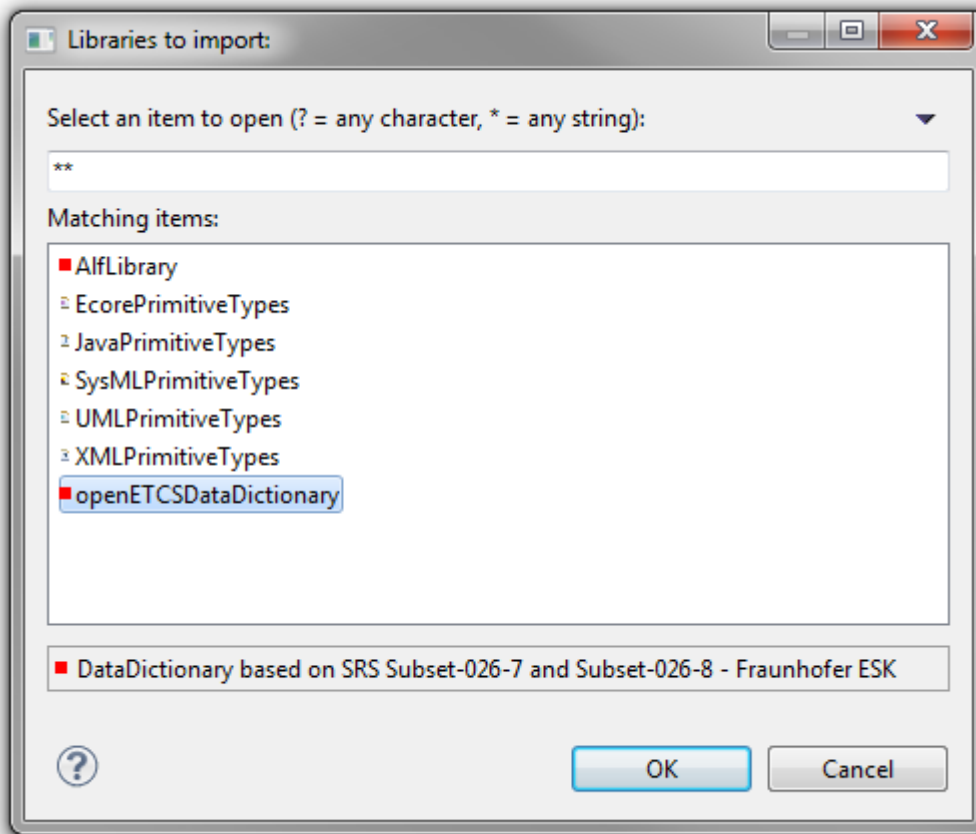
Transformation

There is a standalone application that generates the <tt>.uml</tt> file from the Bitwalker XML files. The tool <tt>org.openetcs.datadictionary.transform</tt> can be found in the standalone folder of this repository. To generate the <tt>.uml</tt> file just execute it as a standalone Java application. The generated <tt>.uml</tt> file is saved into the <tt>/generated</tt> folder.

Further Links

- [Data Dictionary Plugin Development](#)
- [Data Dictionary Design Specification](#)
- [Integration-of-Data-Dictionary-into-Papyrus](#) (Original Proposal)
- [D2.4 OpenETCS Methods: Definition of the methods used to perform the formal description](#)
- https://github.com/openETCS/toolchain/tree/master/tool/bundles/org.openetcs.datadictionary
- https://github.com/openETCS/toolchain/tree/master/tool/bundles/org.openetcs.datadictionary.feature
- https://github.com/openETCS/toolchain/tree/master/tool/standalone/org.openetcs.datadictionary.transform

Authoring Requirements (ProR)

Requirements are managed with [Eclipse RMF](#). Please check out the RMF documentation for now:

- [Tutorial](#)
- Note that the RMF documentation is included in the built-in Help of openETCS.

Tracing Requirements and SysML Models

The Tracing Features allow the linking of *ProR Requirements* and *SysML Model Elements*. This is realized within the requirements model by using the internal links (SpecRelations) and requirements that act as proxies to the SysML Model element. Note that both, links and proxies, can be extended with additional attributes.

Change Tracking is done by using the Change Management Plug-in described below.

This section describes how the tracing feature is configured and used.

Resources:

- Code available at https://github.com/openETCS/toolchain/tree/master/tool/bundles/org.openetcs.pror.tracing
- Specification available at https://github.com/openETCS/toolchain/tree/master/PositionPapers/FormalMind

Before Getting Started

We assume that you have openETCS installed, and open with a ProR requirements model and a SysML model.

We suggest switching to the Papyrus Perspective, to have all relevant Views on the screen. (<tt>Window | Show View | Other...</tt>)

Creating the required data types

A number of datatypes must be created that are being used in the tracing configuration. Please create the following elements, or make sure that they exist. The names are just recommendations that are used throughout this chapter:

- You need the requirements model open and focused.
- The datatype configuration dialog is opened via <tt>ProR | Datatype Configuration...</tt>.
- Datatypes for the Proxy Element:
  - **Proxy Type:** This is a <tt>Spec Object Type</tt> for the proxy elements.
  - **Description:** This is an <tt>Attribute Definition String</tt> that is a child of **Proxy Type:**. This is where the URL to the linked element is stored.
  - **T_Proxy:** This is the <tt>Datatype Definition String</tt> that is associated with **Proxy Attribute:**.

- Datatypes for the Link Element:
  - **Link Type:** This is a <tt>Spec Relation Type</tt> that connects the **Proxy Type:** with the requirement.

You are free to add additional attributes to **Proxy Type:** and **Link Type:**, which will simply be ignored by the Plugin.

Configuring Tracing Plug-In

- You need the requirements model open and focused.
- Select <tt>ProR | Presentation Configuration</tt>
- From the dropdown <tt>Select Action...</tt> pick <tt>Tracing</tt>
- Upon selecting the newly created configuration element, the property view shows 7 entries:
  - **Attribute Names:** Enter the attribute name you picked earlier (**Description**). Note: That you need to type the name exactly as you created it earlier.
  - **Datatype:** The datatype you created earlier (**T_Proxy**).
  - '*Link From Target: '* Whether the source of the link is the SysML Proxy, and the target the requirement, or the other way around.
  - **Link Type: ''' The SpecRelationType you created earlier (**Link Type''').
  - '*Package Prefix: '* (Advanced users) This is the package namespace of the Java object that is being dragged on the requirement. If the namespace fits the beginning of the dragged object, then the drop operation is enabled. The default (org.eclipse.uml) fits all Papyrus model elements. Changing this allows to configure multiple tracing integrations without interfering with each other.
  - **Proxy Attribute:** The attribute from the Proxy attribute, from the drop down (**Description (String) Proxy Type**)
  - **Proxy Type: ''' The SpecObjectType you created earlier (**Proxy Type:''')

Using the Tracing Plug-In

Traces are established with Drag&Drop from the Papyrus **Model Explorer** to the ProR **Specification Editor**. Thus, both must be visible.

The model explorer will be empty if the Papyrus editor is not visible. Therefore, you must rearrange the editors so that both the ProR Requirements Editor and Papyurus Model Editor are visible at the same time. Here is an example on how this can look like:
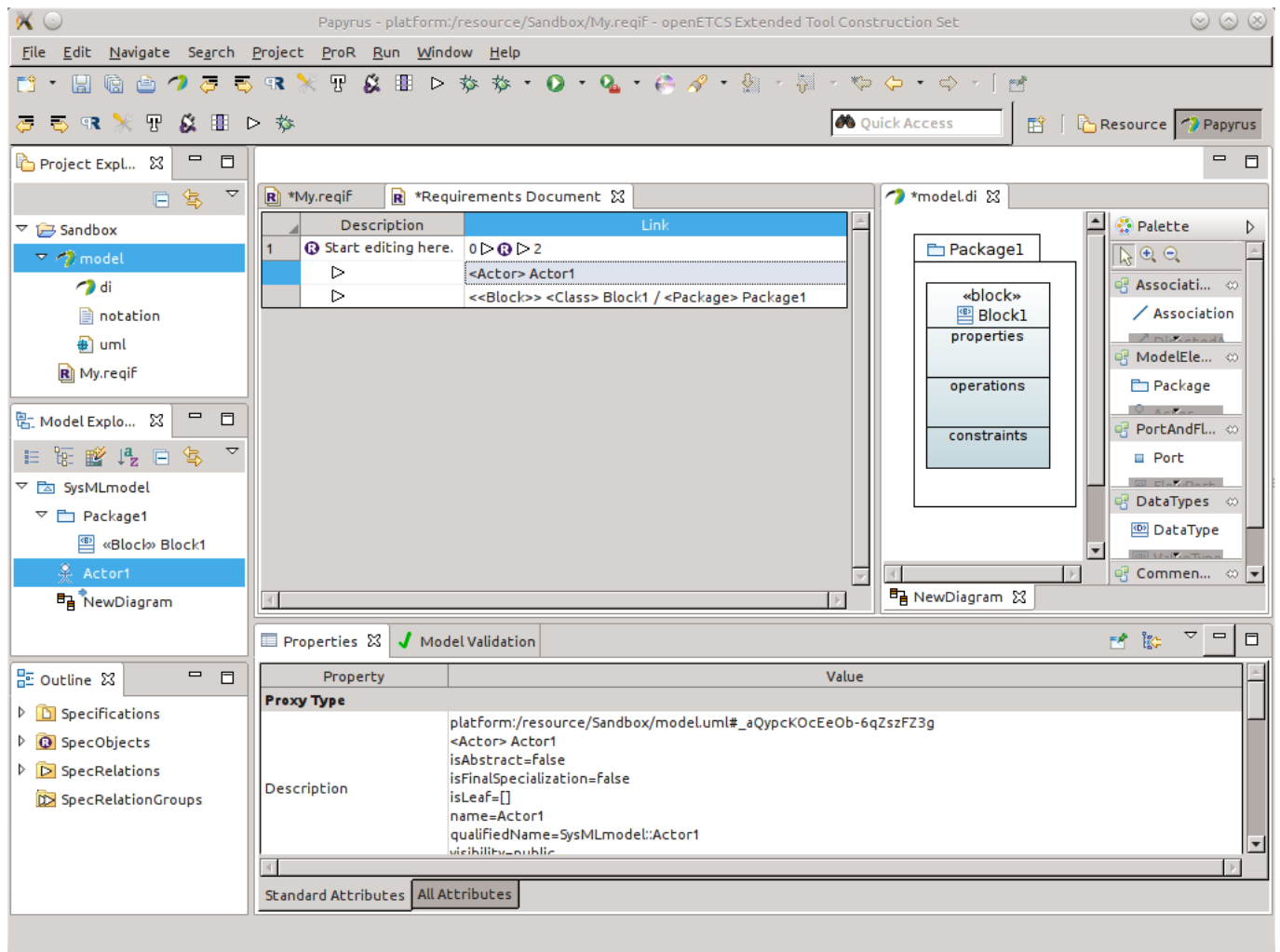
Tipps, Tricks, Notes, etc.

- Model Changes from the Papyrus Model will propagate upon saving.
- The current implementation is not very efficient. Should you encounter performance problems, things can be improved. Note, however, that many performance issues are caused by Papyrus (e.g. opening a model can take several seconds). This has nothing to do with this plugin.
- The original spec was asking for a change tracking mechanism. This, too, will be available as part of Toolsprint 7. In fact, this will be realized with the existing suspect link plugin. Due to a pending ProR release, this is not available yet.
- Help improving the documentation by adding more information here.

Suspect Links

This features allows traces to be flagged, if source or target of that link has changed. Currently, this works for all links within requirements, as well as for links between requirements and SysML model elements (created with the tracing feature.

For now, you need to explicitly install this feature, as described [above](#).
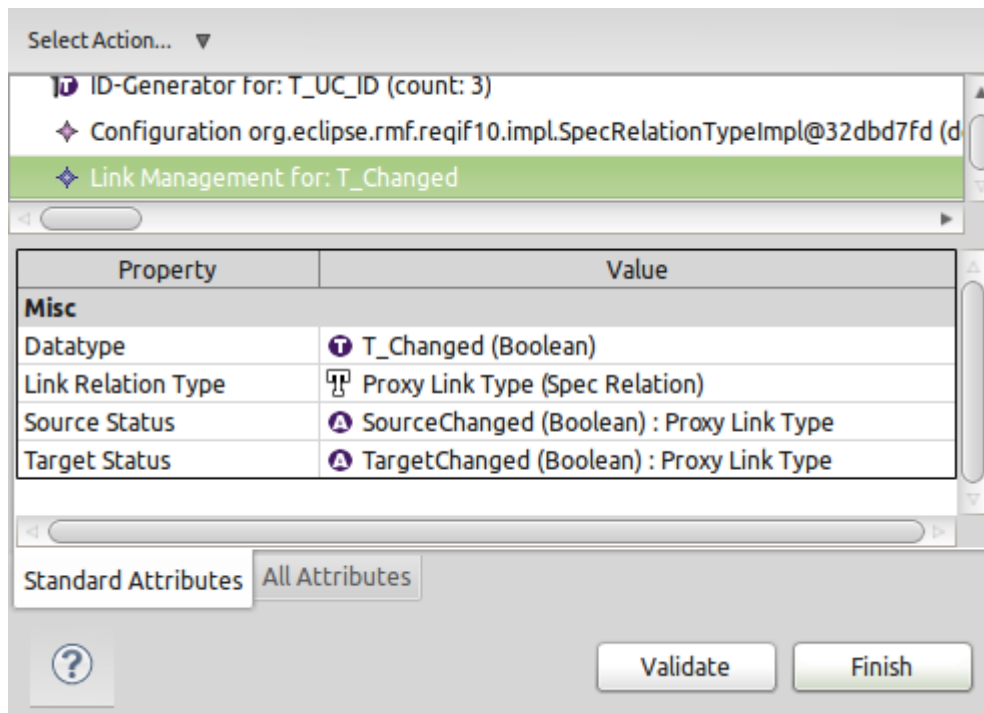
Configuration

The feature needs to be configured only once as follows:

- You need a **SpecRelationType** for this feature.
- The **SpecRelationType** must have two boolean attributes. We recommend calling them **SourceChanged** and **TargetChanged**.
- Don't forget to create a Boolean Datatype for the boolean flags (e.g. '*T_Changed*').
- The **SpecRelationType** should now look like this (here it's called **Proxy Link Type**):

- You also need to configure the **Link Management Pesentation**.
- Go to <tt>ProR | Presentation Configuration...</tt>
- From the <tt>Select Action...</tt> drop-down, select <tt>Linkmanagement</tt>. This should be configured as shown below:

- The fields have the following values:
  - **Datatype** - the datatype of the boolean flags (earlier called **T_Changed**).
  - **Link Relation Type** - the type we created earlier.
  - **Source Status** - the attribute of the flag that indicates that the source changed.
  - **Target Status** - the attribute of the flag that indicates that the target changed.
- This completes the configuration.

Use

The use is simple:

- If the source or the target of a link changed, the corresponding flag will be set, indicated by a yellow triangle with an exclamation mark.
- The flag can also be set manually be double-clicking on the cell.
- The flag can be reset by double-clicking as well (this should happen after verifying that the trace in its current form is still valid.
- If you link to a an element that is not a requirement (e.g. a SysML model element), you need to save the model for the flag to be set.

Tip: Typically, you see the flag only in the properties view. But of course you can add the corresponding flags (**SourceChanged** and **TargetChanged**) to your Specification View as well, giving you a quick overview of changes, if you expand the links.
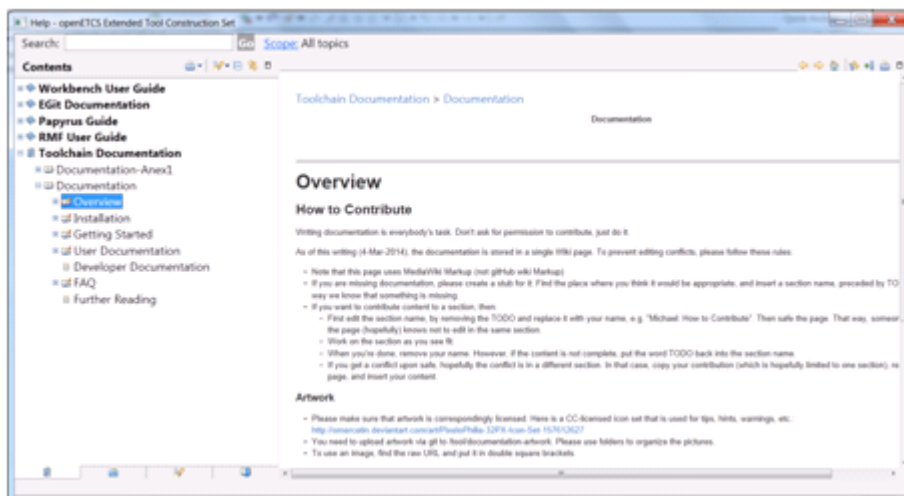
Generating Documentation

Eclipse toolchain plug-in generates Eclipse Help documentation (a hierarchy of HTML files) and PDF documentation from toolchain repository.
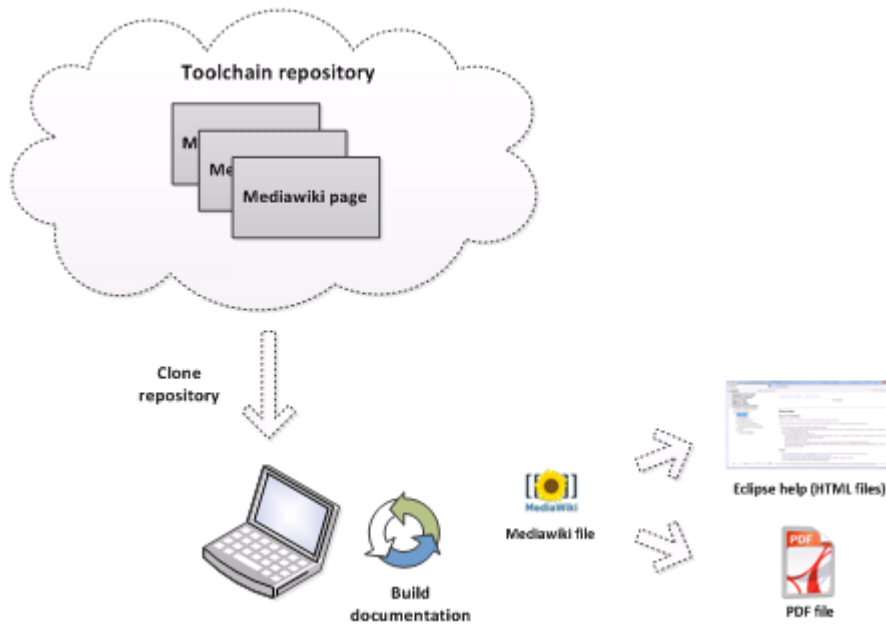
It is necessary to create a Jenkins job to generate automatically toolchain documentation. This job launches mediawiki file cloning process from Github repository to the userâ€™s local machine. In this way, toolchain documentation is cloned. Also, Jenkins job activates documentation generation running build.xml file using Ant to generate Eclipse help, PDF documentation and a table of contents dynamically.

Eclipse help HTML files and a PDF file are generated in "org.openetcs.toolchain \OpenETCS_toolchain" folder.

The following image shows generated Toolchain Documentation openETCS Guide available selecting <tt>Help | Help Contents</tt>in the tool.

Tip: Subpage links should include real section name. Moreover, the spaces between words should be replaced with underscore. E.g.
[[Documentation#Tracing_Requirements_and_SysML_Models|tracing feature]]

Before Getting Started

We assume that you have openETCS Toolchain and Jenkins installed.

Jenkins job configuration

- Jenkins job creation

The first step is to create a new free-style Jenkins job named "openETCS_toolchain_job". This job will launch cloning process and documentation generation daily.
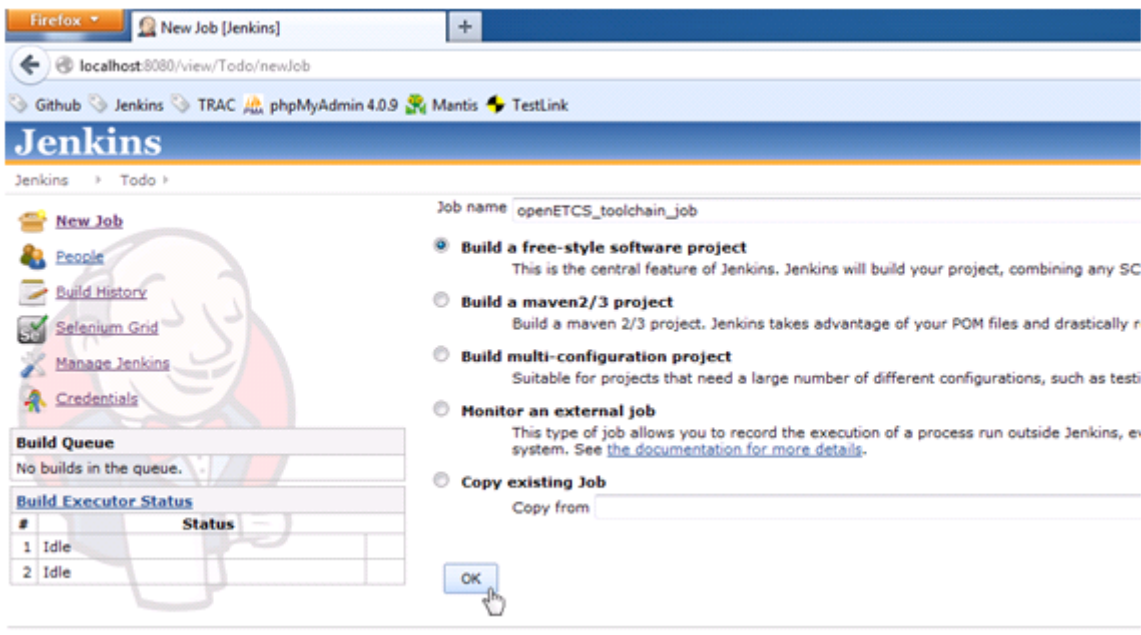
- Jenkins job configuration

In Source Code Management section, select Git and introduce de OpenETCS toolchain wiki URL as is shown in the following image. Also, it is necessary to select Check out to a subdirectory as Additional Behaviours option to specify the local destination folder within the plug-in folder (org.openetcs.toolchain\cloned_repository). Finally, select Clean after checkout as Additional Behaviours option.

The following image shows Source Code Management section configuration. In this case, it has to be taken into account that openETCS Eclipse tool is placed in C:\ and the toolchain plug-in is located in â€œpluginsâ€? folder.

In order to automate the job execution, introduce @daily in the Build Triggers section to specify that the job will be executed daily. Anyway, the user could launch the job execution whenever he/she wants.

In the Build section, select Invoke Ant as Add build step option to allow Jenkins to locate and execute build.xml file using Ant. The path to the file should be specified using Build File textbox. From the previous example, the following path is needed:

C:\openETCS-win32_32\plugins\org.openetcs.toolchain

Finally, push Save button to save the job and its settings.

Developer-Documentation

Setting up the development environment

To set up a development environment for the openETCS tool, perform the following steps:

1. Download the Eclipse Modeling Tools (Kepler SR2)
2. Import the Eclipse projects from this git [repository](#).
 A tutorial on how to configure Egit may be found [here](#).
   1. From Menu File>Import... select Git>Project from Git
   2. Choose "Clone URI"
   3. Get the URL of your repository
   4. Choose a protocol (e.g. https or ssh)
   5. Choose the branch you want to clone and where the local copy will be set on your computer.
   6. Click Import existing project and select a project.
3. Set the target platform. It's located in org.openetcs.releng.target/openETCS-Kepler.target

The target platform defines what need to be build to create the openETCS product.

1. To run openETCS as configured, use org.openetcs.releng.products/openETCS.product

To Build the OpenETCS product

1. Install latest Maven
2. Run Maven in the org.openetcs.releng.parent

       mvn clean verify

OpenETCS Project description

The openETCS core product definition is made of the following project

- org.openetcs.releng.parent: use for the maven build
- org.openetcs.releng.products: List of included features, branding, licencing ...
- org.openetcs.releng.target: Definition of the target platform
- org.openetcs.target.feature: Collection of features that should be included in the target platform
- org.openetcs.core.feature : ??
- org.openetcs.product: OpentETCS start up definition, splash screen ..
- org.openetcs.intro: Welcome page definition

TODO Guidelines for plugin integration

- Feature and Use cases description
- documentation
- basic tests description

Guidelines for licence and credit information

Licence Template

The licence should be stored as an html or a text file in the root folder of each project, in a file named LICENSE.txt or .html.

```
/*
 * Copyright (c) 2014 [NAME OF THE OTIGINAL AUTHOR OR AUTHORS]
 *
 * The contents of this [TYPE OF ARTEFACT, e.g. file] are under the
European
 * Union Public Licence (EUPL), Version 1.1.
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at:
 *       https://joinup.ec.europa.eu/software/page/eupl/licence-eupl
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions
and
 * limitations under the License.
 *
 * Contributor(s):
 *    [Name of Contributor(s)]
 */
```

Licence template for Eclipse new feature

For a new feature or plug-in in eclipse platform the following information should be provided in the feature.xml file:

- Feature Description (optional url and text)
- Copyright Notice (optional url and text)
- License Agreement (shared license, local license, optional url and text)
- Sites to visit

```
<?xml version="1.0" encoding="UTF-8"?>
<feature
      id="org.openetcs.[featureName].feature"
      label="[Short feature title]"
      version="0.1.0.qualifier">

   <description>
      [
```

```
        Feature description
    ]
</description>

<copyright>
    Copyright (c) [NAME OF THE OTIGINAL AUTHOR OR AUTHORS]
</copyright>

<license url="https://joinup.ec.europa.eu/system/files/EN/EUPL
%20v.1.1%20-%20Licence.pdf">
    European Union Public Licence
                            V.1.1
              EUPL Â© the European Community 2007

This European Union Public Licence (the â€œEUPLâ€?) applies to the
 Work or Software
(as defined below) which is provided under the terms of this Licence.
 Any use of the
Work, other than as authorised under this Licence is prohibited (to
 the extent such use
is covered by a right of the copyright holder of the Work).

The Original Work is provided under the terms of this Licence when the
 Licensor (as
defined below) has placed the following notice immediately following
 the copyright
notice for the Original Work:

                    Licensed under the EUPL V.1.1

or has expressed by any other mean his willingness to license under
 the EUPL.
</license>

<url>
    <discovery label="OpenETCS Project Site" url="http://
www.openetcs.org"/>
    <discovery label="[Feature Name] Project Site" url="[Site to
 visit]
</url>

 ...
```

FAQ

Troubleshooting

ETCS

openETCS

Further-Reading