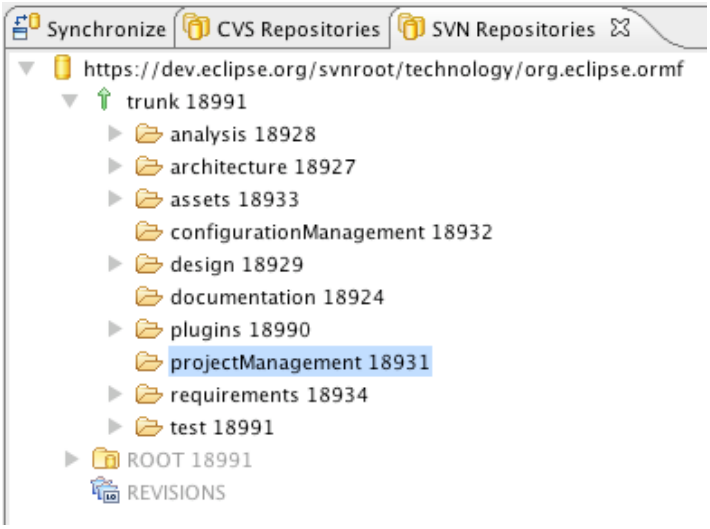
Guide to ORMF's repository structure

Introduction

This document describes the way in which the ORMF repository has been structured. It provides a basis for developer on the ORMF team to find required information and to appropriately use the repository to store their own contributions.

The high level structure

At the highest level, the ORMF repository is organised in a set of ten folders, as indicated in the figure below. Of these, seven represent specific workflows in the development process and, as such, they are



meant to contain all documents, diagrams and other relevant artefacts that pertain to the specific workflow For instance, the project's Test Plan is stored in the test folder, whilst various design UML diagrams will be stored in the design folder. The seven folders in question are the following ones: **`analysis`**, **`architecture`**, **`configuration-Management`**, **`design`**, **`projectManagement`**, **`requirements`** and **`test`**.

The remaining three folders are as follows:

assets

This folder contains all general assets that belong to the project overall and that do not have any other place to live in cleanly in the structure. A very good example of these is the ORMF logo.

documentation

This folder holds all documentation that gets published by various means, including but not limited to the ORMF website or its Wiki. The idea is to hold all source material, that does not typically get published directly, in this folder.

plugins

The figure below show the current contents of the **`plugins`** folder.

This contains the implemented code proper, with all the relevant Eclipse plug-ins for both client and server components. The plug-ins are split between an ORMF directory, containing the generic framework plug-

ins, and a Useme directory, containing the specific plug-ins that make up the Useme exemplary tool. Feature projects and on-line help plug-ins are contained in this folder as well.

N.B.: It is very important to note that the high level folders described above do not contain their artefacts directly. The artefacts are instead placed in one or more Eclipse projects that are only meant as containers for these documents, digrams etc.

We prefer this organisation over a monolythic project that contains all possible artefacts because it is clearer, more intuitive and more modular. It also appears to be a more "standard" way to categorise things in the Eclipse ecosystem.

All the team members have to remember to do is to check out the project to which they want to add artefacts, NOT the high level folders. Please see section How to use the repository below for further details.

How to use the repository

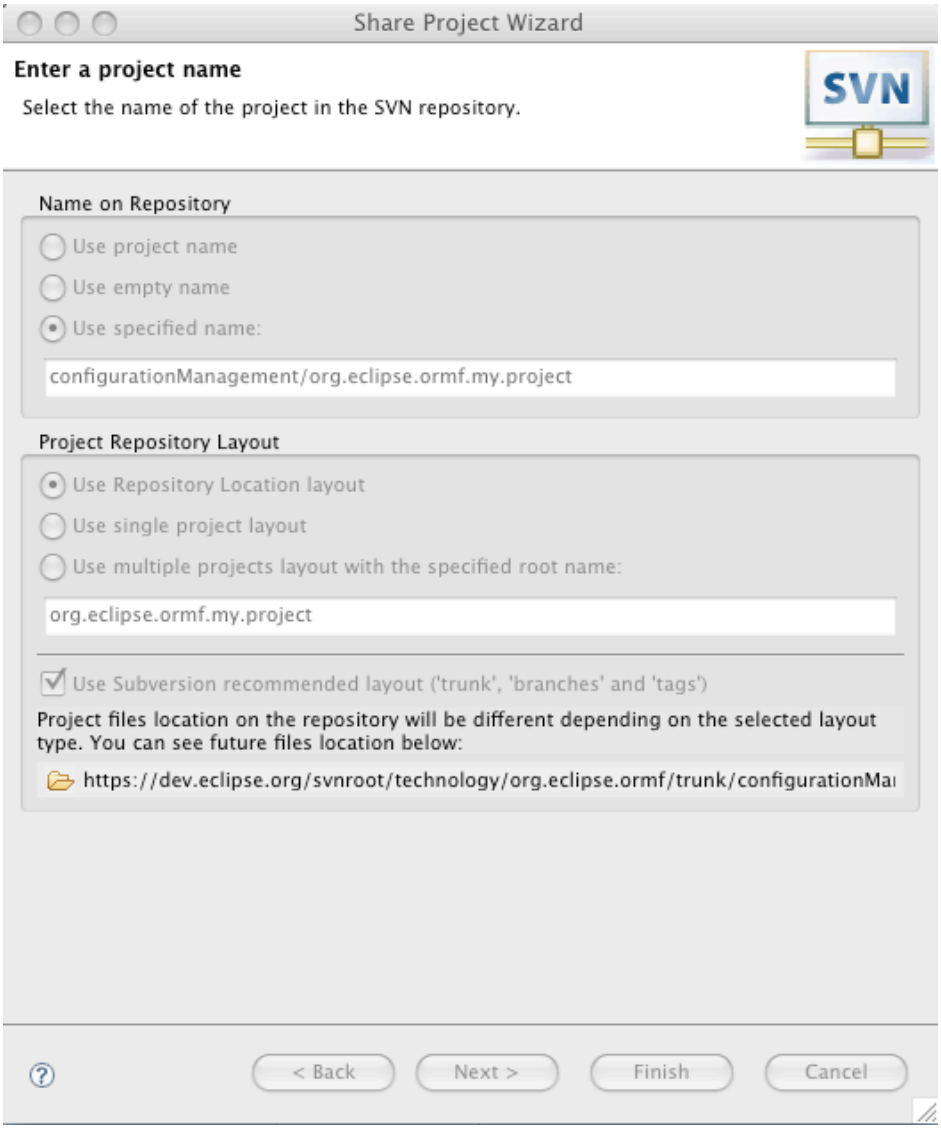
As mentioned above, each high level folder contains zero or more Eclipse projects. Any document, diagram or other type of content is added to the project, not directly to the high level folder.

When needing to work with an artefact, you must check out the relevant project, not the containing folder.

If the artefact does not yet exist, you need to check out the relevant project in your workspace, add the new artefact and then check the project back in.

If the high level folder is either empty or contains projects which cannot naturally contain your contribution, you need to create a new Eclipse project in your workspace, add the desired content to it and then share that project in the appropriate high level folder. The name of the new project should be indicative of its content. It should always be prefixed with "**`org.eclipse.ormf`**" and, if the intuitive name is a multi word one, the words should be separated by a dot. For example, artefacts that describe the ORMF testing strategy have been placed in the **`org.eclipse.ormf.test.strategy`** project, residing within the high level **`test`** folder.

A little care is needed in order to add the newly created project to the right folder in the repository. If, for example, I have created the **`org.eclipse.ormf.my.project`** project and I want to add it to the **`configurationManagement`** folder, I need to invoke the **`Team --> Share`** project action in the usual way. The wizard is instantiated. The default parameters in the first page of the wizard are acceptable. On the second page, I need to define the parameters as indicated in the figure below.



The **Name on Repository** radio button needs to be set at "**Use specified name**" and the corresponding text field needs to be filled in with the folder name followed by a slash followed by the project name.

In this specific case, the text in the text field needs to read as **configurationManagement/org.eclipse.ormf.my.project**, as indicated in the figure.

The Project Repository Layout element can be left at its default values.

The read only field at the bottom of the wizard page provides a useful confirmation of the path of the project being shared in the repository.

From this point on, all ensuing check ins and check outs are completely transparent and hassle free.

Conclusion

The structure of the ORMF repository has been designed to provide a complete container for all artefacts produced during the development process. Its intent is to be exhaustive, clear and intuitive to use, as well as extensible to future needs.

This document's aim is to describe that structure and how it is meant to be utilised.

Any suggestions or comments regarding either the structure itself or this document are welcome.