Leverage Eclipse project capabilities to simplify VOB content handling for IBM Rational ClearCase Remote Client

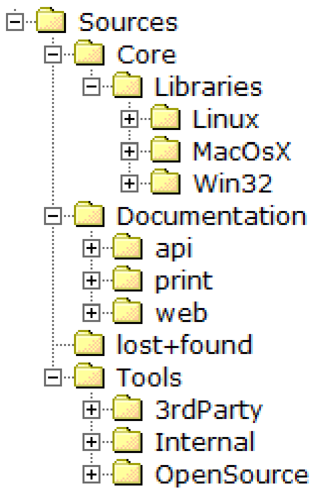
A project or role based approach to managing VOB elements

Fred Bickford IV
Senior Advisory Software Engineer
August 29, 2011

1 Introduction

The purpose of this paper is to demonstrate an approach using Eclipse functionality that can allow IBM Rational ClearCase Remote Client (CCRC) users to load and unload sources from repositories in a logical manner.

This example will use a simple directory structure that is in a ClearCase VOB (Versioned Object Base). The directory structure in a real_world situation could contain many more directories and be much more complicated, yet for the purpose of this paper the example will be kept simple.



What this paper will cover (using the above as an example) is a mock company that has employees who need to use CCRC to load and work with objects in various locations in the VOB. Traditionally a CCRC user would browse and add load rules to their view and CCRC would copy those directories and their contents down to the client. In a simple VOB structure this is fairly easy, however, imagine a case where you have directories nested 10 or 100 levels from a VOB root.

Some users who are confused about what to load into their View will choose the whole repository, leading to longer load times and excess disk space being consumed. Some users may require a whole repository to be loaded entirely; this approach is meant to be a just in time loading model.

In this approach you will see how you can use functionality from a way Eclipse handles projects to treat this tedious task into a simpler "project" based approach. The word project is used in a very liberal sense and is not aligned to a Rational ClearCase Unified Change Management (UCM) project, for example, however, it could be. When we refer to "project" this could be simply a group of users who share a simple role.

From the above example there are two scenarios that will be used:

1. Core Linux development team who are working on an Open Source project.
2. Adding additional teammates who are coming on the team.

As mentioned previously, the example will use a fairly simple directory hierarchy for the purpose of explanation.

2 Requirements

There are some requirements that must be met (or at least understood) to use the approach outlined in this paper.

1. The Eclipse SDK or similar Eclipse IDE must be used with IBM Rational ClearCase Remote Client for Eclipse [plug-in] **.

2. The directory structure would need to be hierarchical, meaning that if a directory or area of the VOB is to be logically considered a project, all the source under that directory is considered part of that "project".

3. It is allowed that a small XML control file (.project hidden file) can be stored inside the directory that is targeted to be treated as a "project", and that this file is accessible by end-users.

3 Eclipse Projects

Eclipse handles projects inside the IDE based on the nature or type of the project and presents to the user capabilities based on that project. For example, Eclipse would present technology capabilities differently between a CDT based project (C Development Toolkit) and a Java project. An example of this would be how Eclipse would build each of these projects, what compilers are used, and other considerations.

The usage of projects as covered in this paper will not be technology based (although your use of Eclipse could be). We will use what Eclipse calls a simple project. This project is not based on any specific technology or tools..

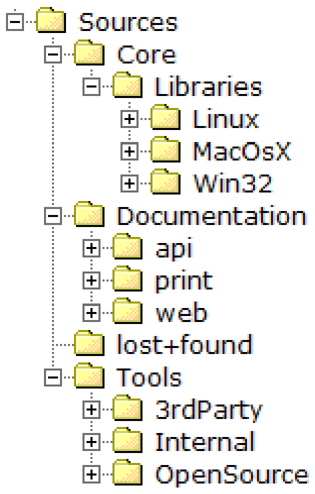
As mentioned in the Requirements section, we do have to consider the “projects” we will use in this paper, as any set of directory structures underneath a specific directory in the VOB will serve as the project.

This will become clearer as you walk through the steps for setting up that project and also serving as a consumer of that project.

4 Project and Team Organization

Scenario 1

The first scenario will be a Linux core developer, based on the existing VOB directory structure (seen below)



```
Sources
    Core
        Libraries
            Linux
            MacOsX
            Win32
    Documentation
        api
        print
        web
    lost+found
    Tools
        3rdParty
        Internal
        OpenSource
```

That developer normally would be required to have the following load rules:

```
load /Sources/Core/Libraries/Linux
load /Sources/Tools/
```

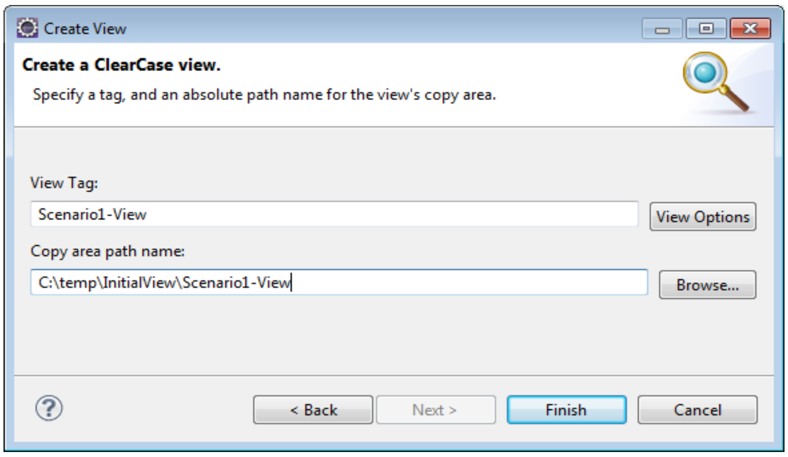
This is fairly easy, yet if this was say a dozen or more different directories, the developer:

- may make mistakes loading.
- may just decide to load everything.
- could load sources that they do not have permission to work on and more.

Eclipse can treat any .project file as something that should be considered the root of a "project". In this first scenario the developer will load two directories. In each of these directories there is already a .project file with the contents below (replace **DirectoryName** with the parent folder name).

```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
       <name>DirectoryName</name>
       <comment></comment>
       <projects>
       </projects>
       <buildSpec>
       </buildSpec>
       <natures>
       </natures>
</projectDescription>
```

The next step will be to now load those projects and subsequent directory substructures into a view.



Create View

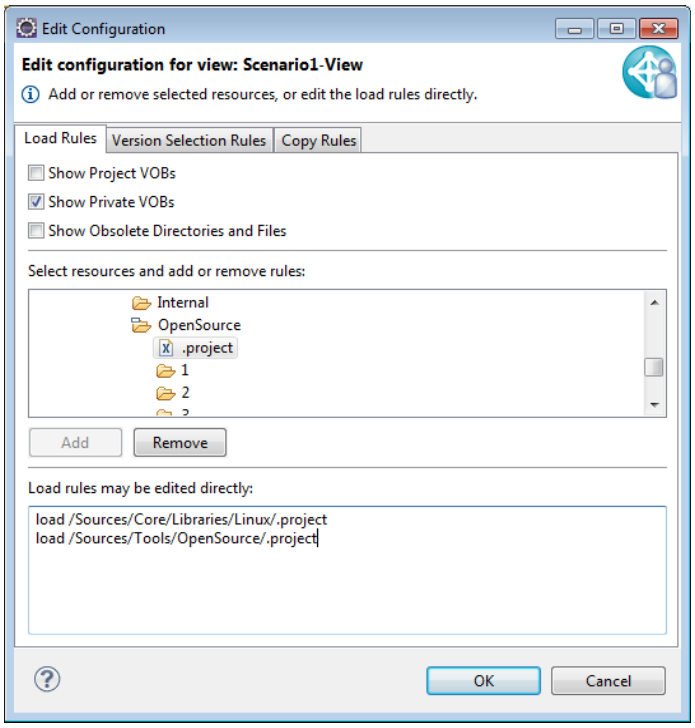
Create a ClearCase view.

Specify a tag, and an absolute path name for the view's copy area.

View Tag: Scenario1-View

Copy area path name: C:\temp\InitialView\Scenario1-View

Just project files will be loaded into the view.



Edit Configuration

Edit configuration for view: Scenario1-View

Add or remove selected resources, or edit the load rules directly.

- [ ] Show Project VOBs
- [x] Show Private VOBs
- [ ] Show Obsolete Directories and Files

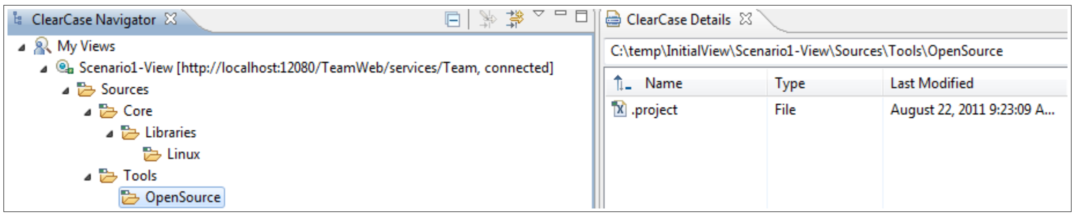
Select resources and add or remove rules:

```
Internal
OpenSource
  .project
  1
  2
```

Load rules may be edited directly:

```
load /Sources/Core/Libraries/Linux/.project
load /Sources/Tools/OpenSource/.project
```

If you look at what has been loaded from a ClearCase perspective in Eclipse, you only see the two .project files:

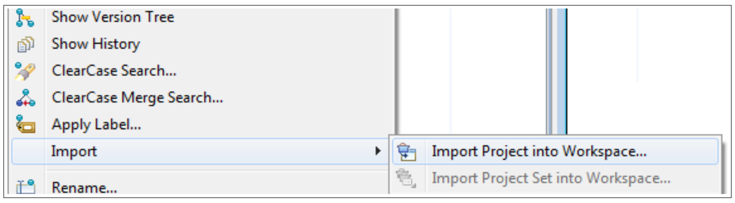


C:\temp\InitialView\Scenario1-View\Sources\Tools\OpenSource

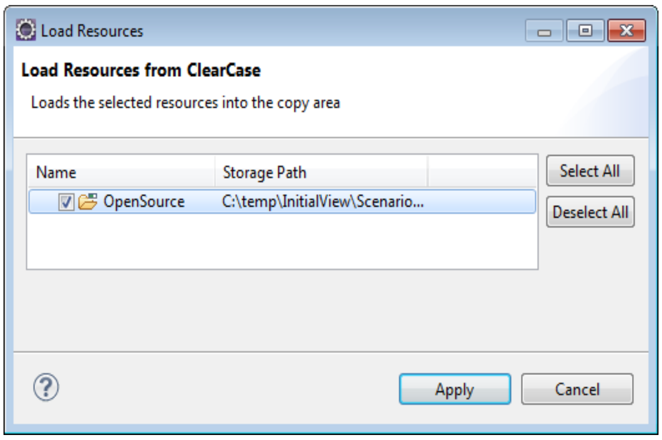
Name	Type	Last Modified
.project	File	August 22, 2011 9:23:09 A...

However, you will bring these into your eclipse workspace, and load the contents that are represented by these “projects”.

Right click on the .project file and import this project into the workspace:


Show Version Tree
Show History
ClearCase Search...
ClearCase Merge Search...
Apply Label...
Import
Rename...
Import Project into Workspace...
Import Project Set into Workspace...

You will be prompted that the folder will need to be loaded

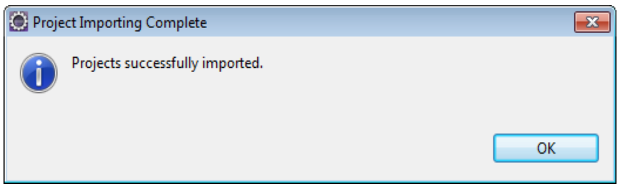


Load Resources from ClearCase

Loads the selected resources into the copy area

Name	Storage Path
✓ OpenSource	C:\temp\InitialView\Scenario...

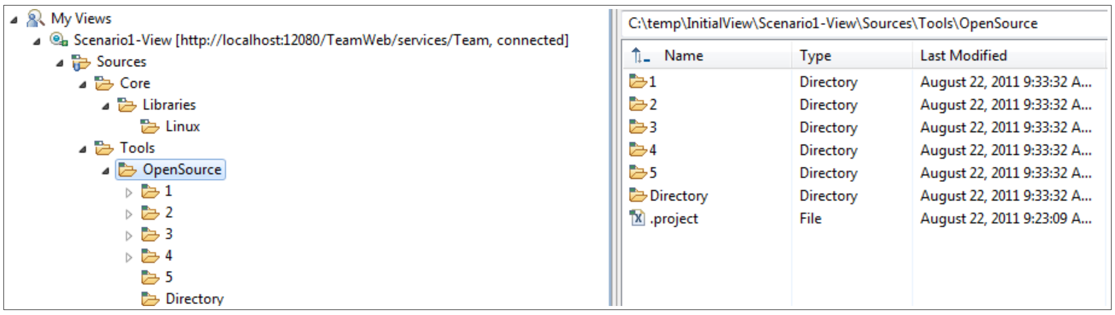
And you will get a notification of the project being loaded:



Project Importing Complete

Projects successfully imported.

And in a ClearCase Perspective you will see the additional loaded objects

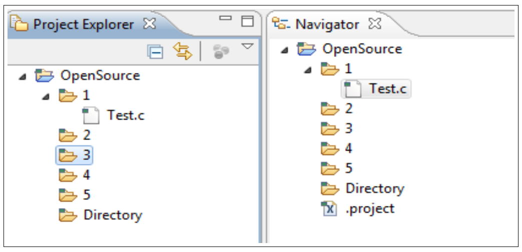


- My Views
  - Scenario1-View [http://localhost:12080/TeamWeb/services/Team, connected]
    - Sources
      - Core
        - Libraries
          - Linux
      - Tools
        - OpenSource
          - 1
          - 2
          - 3
          - 4
          - 5
          - Directory

C:\temp\InitialView\Scenario1-View\Sources\Tools\OpenSource

Name	Type	Last Modified
1	Directory	August 22, 2011 9:33:32 A...
2	Directory	August 22, 2011 9:33:32 A...
3	Directory	August 22, 2011 9:33:32 A...
4	Directory	August 22, 2011 9:33:32 A...
5	Directory	August 22, 2011 9:33:32 A...
Directory	Directory	August 22, 2011 9:33:32 A...
.project	File	August 22, 2011 9:23:09 A...

In Comparison when looking at a Eclipse perspective (using the resource perspective as it is not specific to a certain programming language) you will see things based on the project:



Project Explorer

- OpenSource
  - 1
    - Test.c
  - 2
  - 3
  - 4
  - 5
  - Directory

Navigator

- OpenSource
  - 1
    - Test.c
  - 2
  - 3
  - 4
  - 5
  - Directory
  - .project

Other than the Eclipse perspective now allowing you to work in more of a logical project approach (treating these source folders as "projects", this really does not help specifically with loading objects. The load rules when doing the import of the project file, did change from:

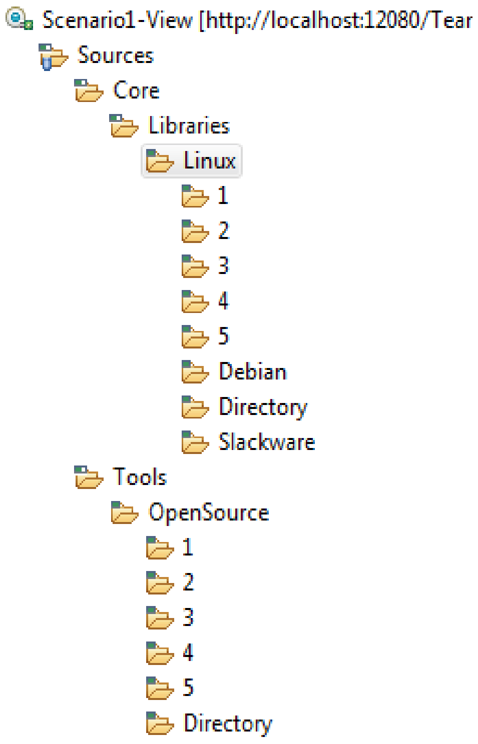
```
load \Sources\Core\Libraries\Linux\.project
load \Sources\Tools\OpenSource\.project
```

to

```
load \Sources\Core\Libraries\Linux\.project
load \Sources\Tools\OpenSource
```

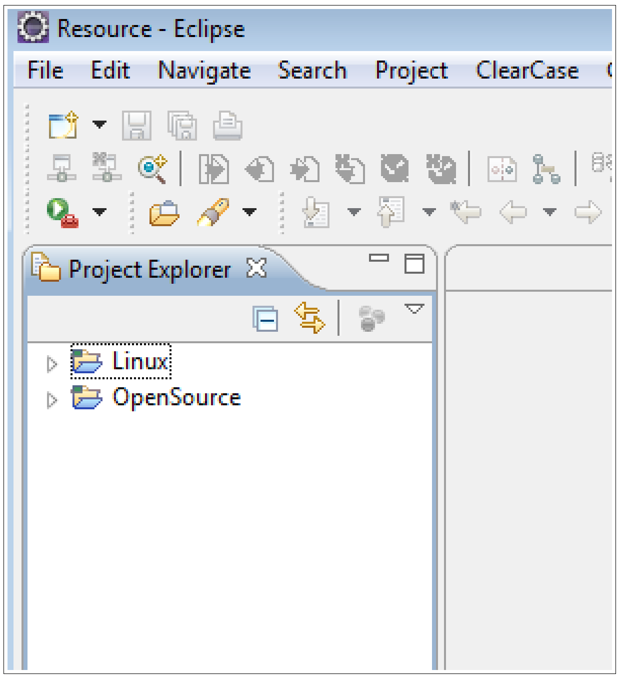
Users could do that themselves, but what if the specific "projects" or directories were all over the VOB? Having the user locate all the .project files could be just as problematic as supplying correct load rules without eclipse.

Lets go back into our Scenario1-View, and import the other .project file. (`\Sources\Core\Libraries\Linux\.project`). If you look at what you have loaded from a physical perspective (like the ClearCase Perspective), it looks like:



```
Scenario1-View [http://localhost:12080/Tear
  Sources
    Core
      Libraries
        Linux
          1
          2
          3
          4
          5
          Debian
          Directory
          Slackware
    Tools
      OpenSource
        1
        2
        3
        4
        5
        Directory
```

and in the logical view in Eclipse it looks like:



```
▷ Linux
▷ OpenSource
```

Scenario 2

However, imagine that you may have dozens of projects, how do you make loading of objects easier on co-workers and teammates?

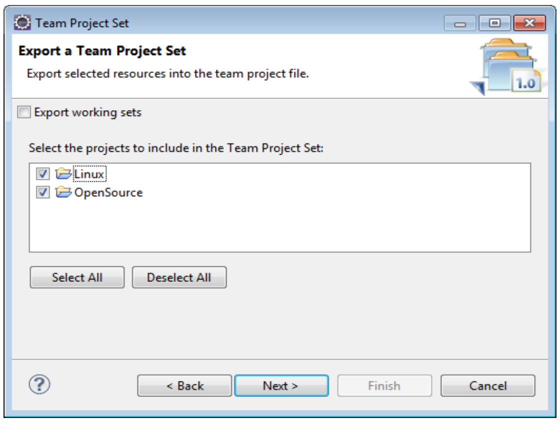
Eclipse provides an ability called Team Project Sets, which can help users bundle projects together to make it easier to have related projects combined in a workspace. From a ClearCase point of view, this allows users to create combinations of projects (folders) for various teams or just based on certain roles or requirements.

In this example you will create a Team Project Set to make it easier for a co-worker who is functionally working on the same sources, and needs to have access to the same load rules.

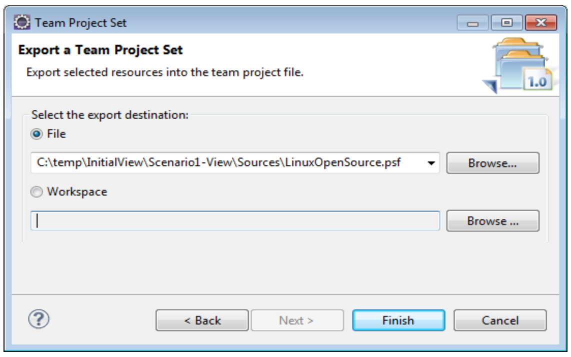
You could copy the same load rules from one view to another, however, by using Team Project Sets you have greater flexibility.

The project set needs to be created by one user who loads the appropriate projects into their workspace. Since in Scenario1-View you have all your projects present, you will use that view.

From Eclipse (logical perspective) go to `File>Export>Team>Team Project Set` and click `Next`.


Team Project Set
Export a Team Project Set
Export selected resources into the team project file.
☐ Export working sets
Select the projects to include in the Team Project Set:
☑ Linux
☑ OpenSource

Then select both projects. Remember this could be any combination and any number of projects in the VOB. Click `Next`.

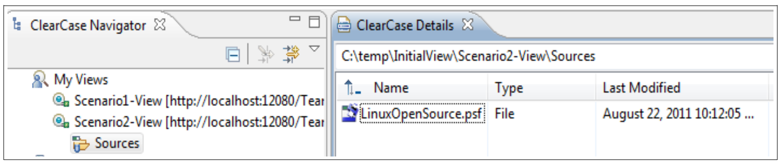

Team Project Set
Export a Team Project Set
Export selected resources into the team project file.
Select the export destination:
● File
C:\temp\InitialView\Scenario1-View\Sources\LinuxOpenSource.psf
○ Workspace

The only requirement is that all of the projects are able to be located in the VOB and the project set file (.psf) ** is at some easily accessible location. For this example you will store the file in the root of the VOB. If Unified Change Management is in use, a common location may be on the integration stream under a baseline that is accessible for everyone.

The file extension needs to be .psf , however, the file name can be changed to reflect the projects. In this case we renamed it to reflect the team role.

After saving, add the file to source control (to ensure other users can leverage it).

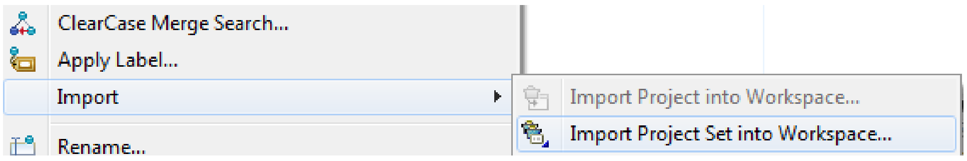
Now as a new user you will create a view and ONLY load the `LinuxOpenSource.psf` file in the view.


ClearCase Navigator
My Views
Scenario1-View [http://localhost:12080/Tea
Scenario2-View [http://localhost:12080/Tea
Sources
ClearCase Details

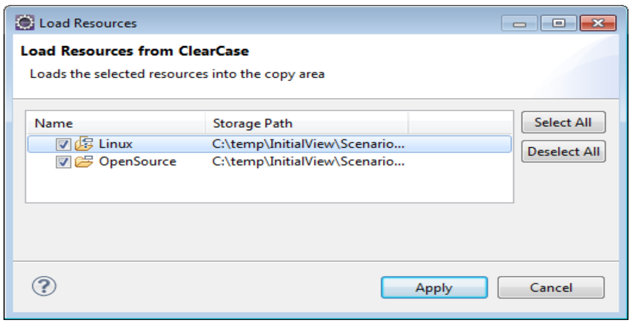
C:\temp\InitialView\Scenario2-View\Sources

Name	Type	Last Modified
LinuxOpenSource.psf	File	August 22, 2011 10:12:05 ...

If you right-click on the Team Project Set file, you can import this project set into our workspace.


ClearCase Merge Search...
Apply Label...
Import
Rename...
Import Project into Workspace...
Import Project Set into Workspace...

You will be prompted to load the resources that comprise this collection of “projects”


Load Resources
Load Resources from ClearCase
Loads the selected resources into the copy area
Select All
Deselect All
Apply
Cancel

Name	Storage Path
☑ Linux	C:\temp\InitialView\Scenario...
☑ OpenSource	C:\temp\InitialView\Scenario...

And in one action you have gone from loading

```
        load \Sources\LinuxOpenSource.psf
```

to

```
        load \Sources\LinuxOpenSource.psf
        load \Sources\Core\Libraries\Linux
        load \Sources\Tools\OpenSource
```

5 Conclusion

This paper demonstrates that with a small amount of administrative configuration in Eclipse and ClearCase, ClearCase Remote Client for Eclipse can leverage Eclipse projects (.project) to allow directory structures to be treated as “projects” for the benefit of logical organization.

Using Project Set Files (.psf) can also allow for the organization of various “project” collections based on employees roles, specific work teams or just team collaboration.

The use of projects and team Project Set Files is not specific to ClearCase as a Change Management Tool and is applicable to all tools that support the Eclipse “Team” model, thus, making this approach more extensible and something that should be comfortable to users of other CM tools.

6 Additional Resources

Team Project Set File:
http://wiki.eclipse.org/PSF

Importing Team Project Sets in CCRC:
https://publib.boulder.ibm.com/infocenter/cchelp/v7r1m0/topic/com.ibm.rational.clearcase.ccrc.help.doc/topics/c_ccrc_eclipse_psf.htm

Eclipse Team:
http://www.eclipse.org/eclipse/platform-team/