

Using EA

Setting up Version Control

by Dermot O'Bryan

All material (c) Sparx Systems 2005

<http://www.sparxsystems.com>

Table of Contents

INTRODUCTION	3
THE MECHANISM	3
DEPLOYMENT MODEL	3
BASIC CONCEPTS OF VERSION CONTROL	5
INSTALLING VERSION CONTROL.....	6
PREPARATION BY VERSION CONTROL TYPE	8
SCC COMPLIANT	8
<i>Server</i>	8
<i>Client</i>	8
<i>Test Client – Server</i>	8
<i>Connecting EA to the SCC</i>	8
CVS.....	12
<i>Server</i>	12
<i>Client</i>	12
<i>Test the Client – Server</i>	13
<i>Configuring EA</i>	14
<i>Some CVS technical Points:</i>	16
SUBVERSION.....	17
<i>Server Setup</i>	17
<i>Client</i>	19
<i>Test the Client – Server</i>	19
<i>Configuring Version Control with Subversion</i>	20
TESTING THE INSTALLATION.....	22
APPENDIX:.....	25
1. VERSION CONTROL DEPLOYMENT SCHEMAS	25
<i>Private Model</i>	25
<i>Shared Model</i>	26
<i>Work Group Model</i>	27
3. SUBVERSION URLS	28
4. VERSION CONTROL PRODUCTS.....	28
3. LINKS	29
<i>CVS links</i>	29
<i>CVS Client Installation on Windows</i>	30
<i>Subversion links</i>	30

Introduction

Enterprise Architect supports version control of packages through an interface to third party version control repositories. The three main types of repositories supported are the Microsoft SCC compliant version control applications, the open source CVS version control and Subversion version control.

This paper presents the key processes for setting up EA to work with the main version control systems, as well as giving a general insight into working with third party products that claim to support the SCC interface.

Note: This document is available prior to the release of EA version 6.0. Support for Subversion will not be available until EA version 6.0 is released.

The Mechanism

In general, version control systems consist of a repository that holds a "master" copy of each revision, and a local working copy of these files on each user's PC. It is the local working copy of the files that users change, before the file is "committed" or "Checked-In" to the repository, creating the next revision.

When setting up a version control system, a directory must be created on each user's PC that will contain the local working copies. The method for doing this varies from product to product and you should consult the relevant documentation for your particular product. CVS and Subversion refer to this directory as the "Working Copy" directory, whereas SCC products generally refer to this directory as the "Local Project Path".

When EA adds packages to version control, it does so by first exporting the package to an XMI file, then this XMI file is added to version control. EA must be instructed to create these intermediate files in the users' "Working Copy" directory that has previously been set up for use with their Version Control product. The path to this directory is specified as part of EA's Version Control configuration procedure.

When a user "checks-out" a package, EA sends a command to the Version Control system, to "check-out" the file. The Version Control system then puts the latest version of the file into that user's working directory. Any previous version of the file in that directory is overwritten with the one coming out of the Version Control system. EA then imports the package file into the model, replacing the existing version of the package.

With the Check-in, the package is exported as an XMI file, overwriting the existing local working copy of the file, the new file is then checked-in to the version control system.

For each of the processes involving version control, Enterprise Architect keeps track of the version control status of all packages.

Deployment Model

The deployment model is the relationship between your EA application, the EA repository and the version control product. When setting up a development environment to support team development, you will need to decide on what kind of deployment model to adopt. This would include decisions about the connectivity between the user's workstations, the EA repository(s) and the version control system.

There is not really any "correct" way to set up your development environment, although certain deployment models will support particular ways of working, some may be more suited to your needs than others.

If your team needs to see the latest changes to the model almost instantly, then a shared model stored in a DBMS repository (or an EAP file on a network drive) is a good way to achieve that. If there are users that will be "experimenting" with different ways of modelling things and want the rest of the team to see only the end product, then private models stored locally is the better option. It really depends on how you want the team to work.

Either way, a private or a public model will interface with the version control system.

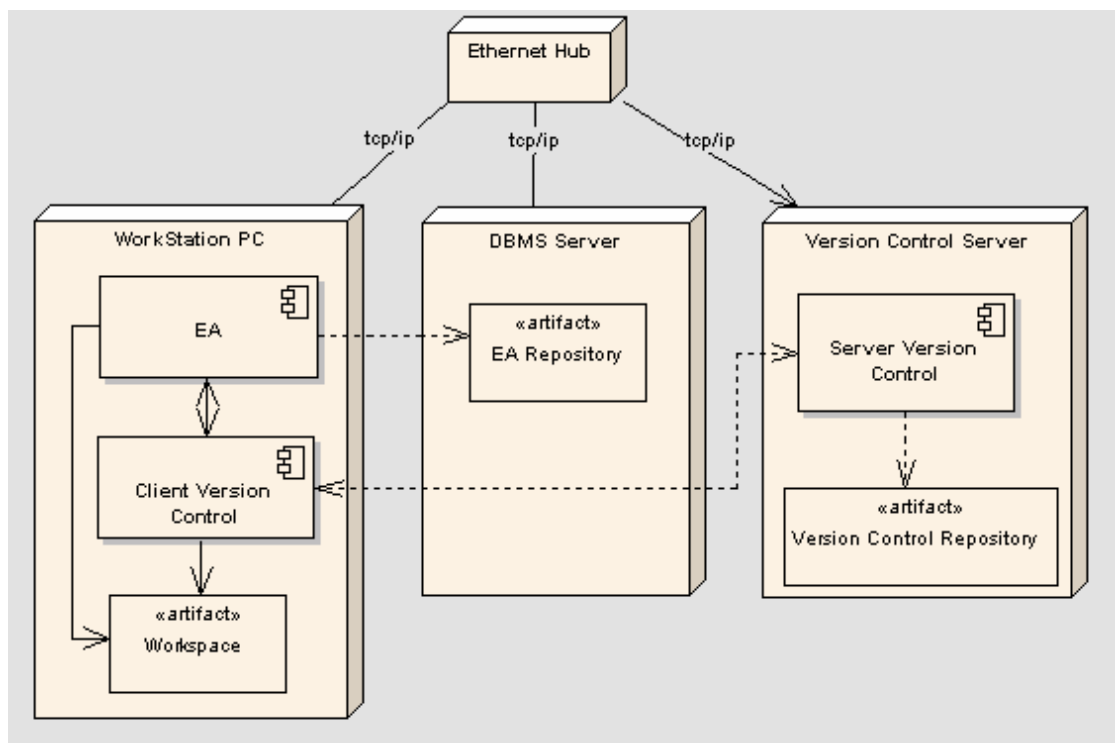
For more information on the deployment of Enterprise Architect using different repository types see the Deployment white-paper available from:

http://sparxsystems.com.au/downloads/whitepapers/EA_Deployment.pdf

In order to make the version control functionality available the system needs to be set up to interface with an external version control application. There are several ways in which EA can be interfaced with the Version Control facility.

The simple and common scenario is where users are setup to share a central EAP file or DBMS database. This configuration allows users to see other users' packages without explicitly having to retrieve them. Version control regulates access to packages, and maintains package revision history.

Below is a model of a simple setup showing a workstation connected to the Server running the version control as well as a data server for EA's repository:



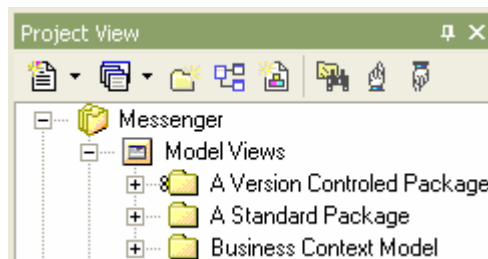
As stated above there can be a number of variations on this basic schema. Some standard variations of this are set out in the appendix: [Version Control Deployment schemas](#).

Basic Concepts of Version control

The Version Control features of Enterprise Architect allow the user to version control any package or tree of packages. The version control provides the following functionality:

- Coordinating the sharing of packages between users.
- Saving a history of changes to EA.
- The ability to retrieve previous versions.

Once version control is set up for a repository, the main screen section that a packages version control state is viewable is in the **Project View**.

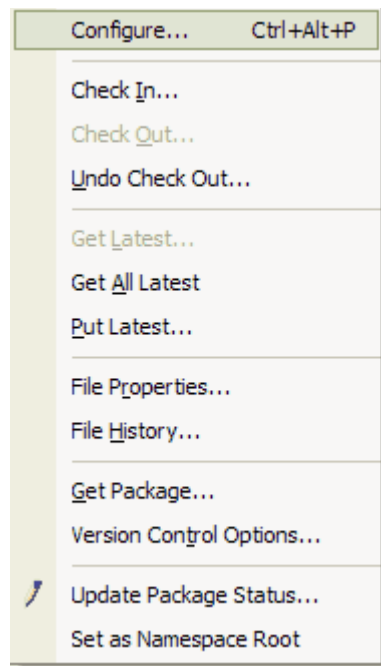


This is also where the key functions of *Checking In* and *Checking Out* packages are performed.

You can see the version control state for a package (or package tree) in EA's **Project View**. The following table shows the different icons representing the version control status of a package:

Icon	Definition with Respect to Version Control
	Package is not version controlled
	Package is version controlled and checked In (or checked out to another).
	Package is version controlled and Checked Out
	This icon is also used when a package is controlled but not <i>version</i> controlled (i.e. exported).

The basic functions for *Checking In* and *Checking Out* a package are accessible by selecting a package in the **Project View**, then by right-clicking, from the context menu select: *Package Control*. This brings up the set of version control functions available for use on the current package:



For more information on the functions available above see EA's help file under [Model Management | Version Control](#).

Installing Version control

In order to use version control in EA it is necessary to firstly install the external version control system that you are wanting EA to communicate with. This setup process is dependant on each particular version control product. Given the number of different products that are supposedly compatible with SCC, it is not appropriate to give the full details for setting up each of these in this document. What we can provide, is a brief outline of the set up process for the key systems and refer you to appendices that have reference links for common version control products.

The following is a summary of the key steps that are required for setting up the external version control system: (For information more specific to each class of product, see [Preparation by Version Control Type](#) below.)

1. **Install your Version Control product.**

Typically a server process will run on a remote server machine and the client process runs on the local machine where you will be running EA. So in most cases this requires configuring at least two machines – the Server and a client workstation. Installing each of these is as follows:

- a. Install the server side Version Control software on the server.
 - i. Check that the installation sets up a version control repository. If not, create a repository on the server. (CVS and Subversion require this).
- b. Install the workstation Version Control client software on your workstation.
 - i. Ensure that this is configured to address the server (by server name, TCP/IP, port etc).

2. Create a Local Work Space

For SCC products, this step varies greatly. Some will run a wizard; some will require an existing folder to be “nominated” as the Local Project path.

For CVS and Subversion, this step means creating a local working folder and “checking-out” a module/subtree from the repository.

3. Verify Externally that the Newly Installed System Works.

Run the version control client from outside of EA. This is a simple step but it helps circumvent any later confusion!

A common process for this is to create a text file and have it entered into the version control system. Check that it is marked as being in the version control system.

4. Setup up EA to interface with the Version Control.

Once the version control system has been externally verified as operating, you can then start the process of setting up EA to interface with it. This is explained further in the following section: [Preparation by Version Control Type](#). This contains sub-section deals specifically with SCC, CVS and Subversion Version control systems.

5. Test the installation

Once EA has been configured you can then place an EA package under Version Control and test that it has been accepted by the Version Control.

Preparation by Version Control Type

The following gives some more detail on the general set up required for the key types of version control systems supported by EA.

SCC Compliant

SCC compliant version control products typically have three components;

1. The server, which creates and manages the repository.
2. The GUI client, that provides an interface through which users may add, check in and check out files to/from the repository.
3. A SCC client. This is a programmatic interface that allows IDE's such as EA, to programmatically interact with the server.

To set up version control within EA, first you need to install your (SCC compliant) version control product, on both the server as well as the client machine.

Server

To setup on the server, refer to the documentation provided with the SCC application. A version control repository must be set up using the SCC server and access to that repository must be available to all intended users.

Client

When setting up the client machine, make sure that you install the SCC client module (it may be optional) and then set up your project space within that product. Typically you need to associate a *local working folder* with your source control project. Take note of the location in which this is set up, as this will be used for configuring EA's interface to Version Control.

Test Client – Server

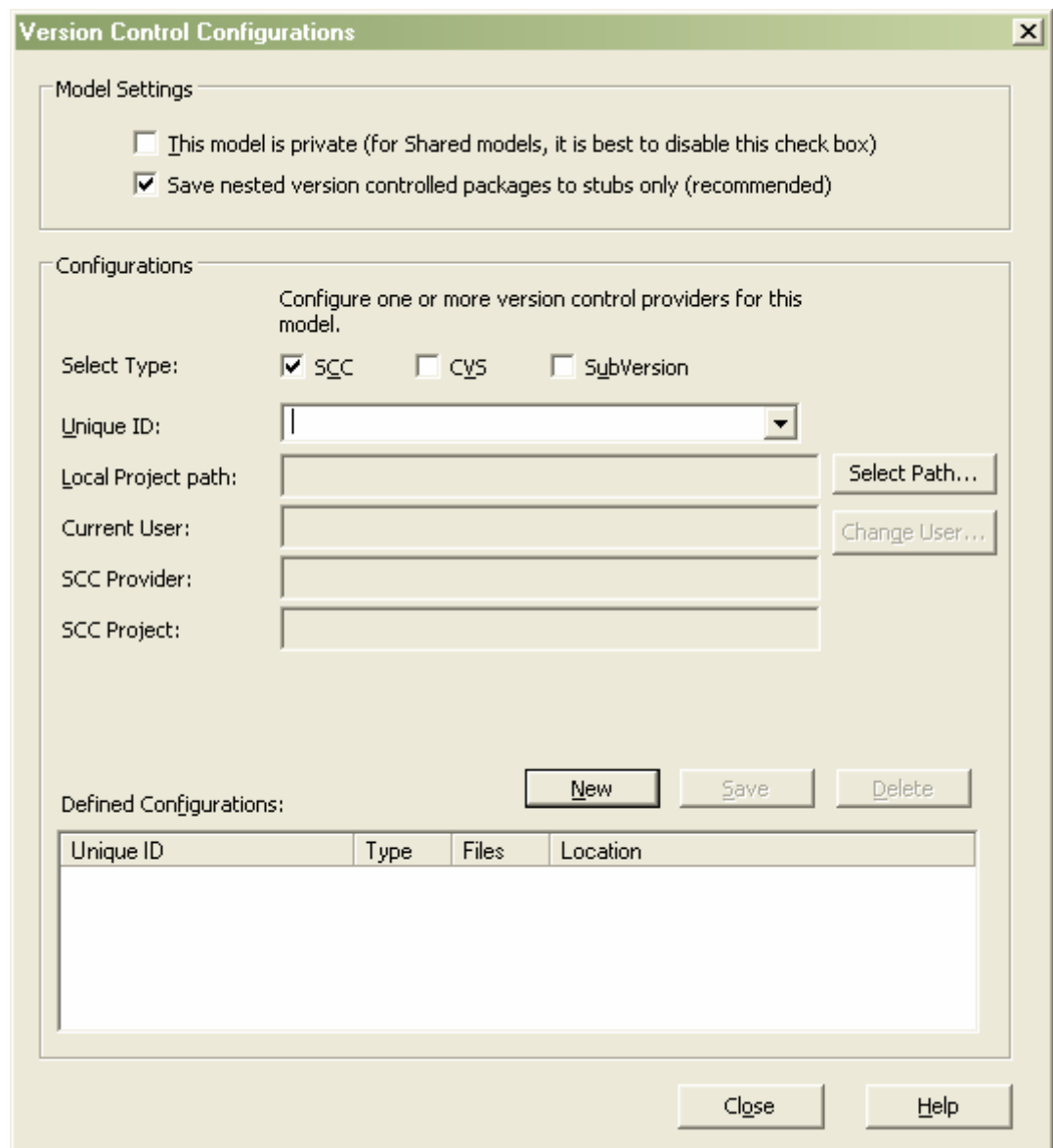
Always test the connectivity of the client and server by manually placing a file on the version control system and check its status using the GUI client module.

Connecting EA to the SCC

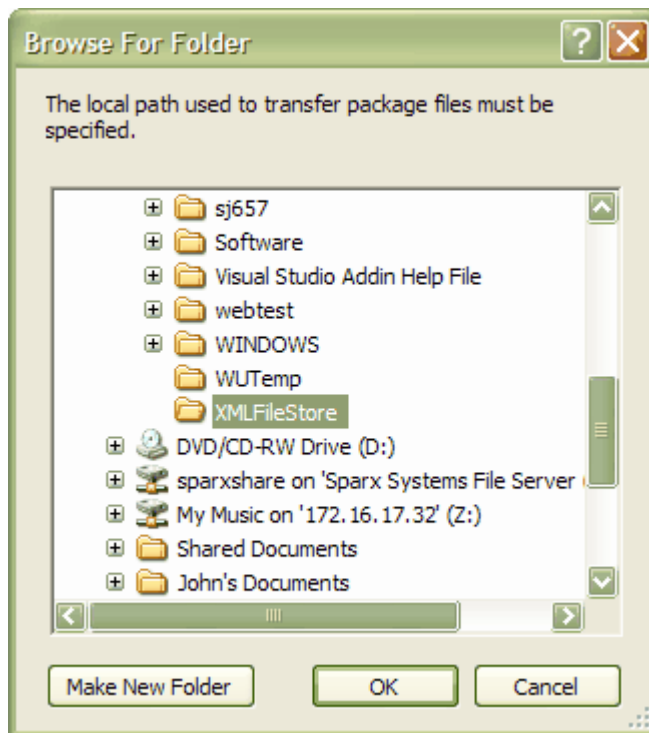
To set up a connection in EA with the SCC client/server installed above – open EA and perform the following:

1. Open/Create the EA model you wish to place under version control.
2. From the *Project | Version Control* submenu, select *Set Version Control Options*.

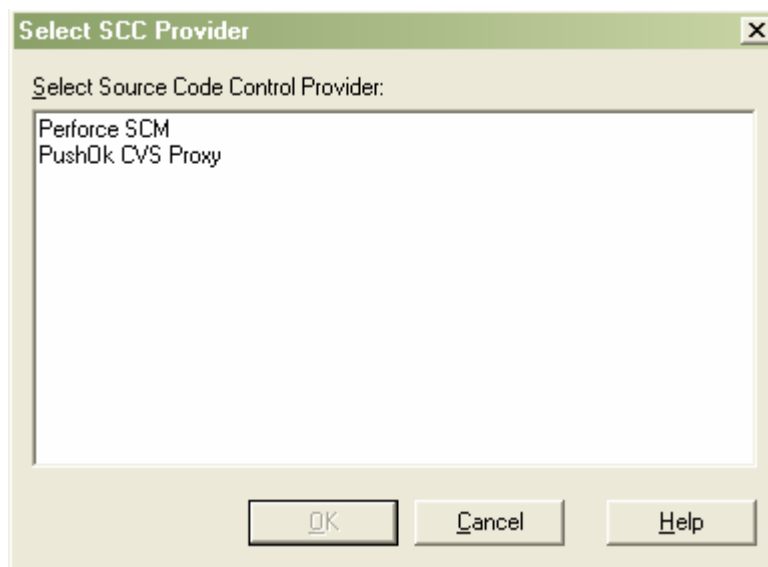
This will invoke the following dialog:



3. Press the *New* button and ensure that the *Select Type* check box is set to *SCC*.
4. Type in a name for the configuration you are about to create.
5. Click on the Local project path [\[Select Path\]](#) button.
6. This will bring up the following window prompting for the location of the *local working folder* used by the Version Control Client. Select the directory where the local workspace is stored. This should have been defined when setting up the workstation side of your version control product.



On confirming this with **OK**, you will be prompted with the dialog shown below. In this window EA lists all of the SCC interface clients that it has found.



Not all products install the SCC interface client by default. If after installing your product, it does not show in the above list, it would indicate that you do not have a client module for that product installed on that machine or the windows registry was not updated correctly.

Assuming that your version control product does provide an SCC interface client, but it does not appear in the list presented by EA, you may need to re-install your version control client and select an option specifying that you want to install the SCC interface client. Your Version Control vendor should be able to provide information and instructions in this regard.

Next, launch EA and carry out the process listed in: [Testing the installation](#).

CVS

CVS version control consists of two key components;

1. The server, which creates and manages the repository.
2. The client, that provides the interface through which users add, check in and check out files to/from the repository,

A CVS server *Repository* will contain *Modules*. You will need to know the name of the Module that you will be accessing.

To set up version control within EA, first you need to install your CVS server on the server machine and then install the CVS client on each user's workstation machine.

The following are some steps that must be completed before you can use CVS with EA. Refer to your CVS documentation for details.

Server

You will need to create a *repository* containing a *module* that you can use to control your EA package files. CVS does not create a *repository* by default when installing the server, so specific CVS commands must be used to set up a *repository*.

You will need to know the name of the *module* you will be accessing. This is used later, when setting up the client.

You must also set up *UserID's* for all users that will be accessing the repository.

Client

On each client workstation, you need to check-out the *module* you will be using, from the *repository*, thereby creating a local working copy of the module. Choose a suitable directory in which to create this working copy, EA will be exporting and importing packages files using this directory.

The local working directory (*Working Copy Path*) that you set up here needs to be specified in EA when defining the Version Control Configuration.

Once installed, you can then run the following commands in the DOS prompt to set up the client workspace and connect into the server.

Note: These commands are case sensitive.

The steps to carry out this are as follows:

1. Open a DOS prompt.
2. Navigate to the directory where you will be storing the client local workspace. The directory that will be defined within this directory will need to be defined in the EA Version Control configuration as the *Working Copy Path*. An example would be:

```
cd \myCVSWorkspace\MyLocalProjectDirectory
```

3. In the DOS prompt login into our CVS repository on the server using the following command:

```
cvs -d :pserver:<User>@<ServerName>:<PathToRepository> login
```

Substitute the fields:

- <User> with the User-name defined on the server.
- <ServerName> with the Name of the server.
- <PathToRepository> with the URL of the repository.

Example: `cvs -d :pserver: joan@CVSServer:/cvs login`

Note: The <ServerName> may require an IP address rather than the name if the connection is across a WAN.

4. Checkout the CVS module on the server using the following DOS command:

```
cvs -d :pserver:<user>@<serverName>:<PathToRepository> co  
<ServerModule>
```

Example: `cvs -d :pserver: joan@CVSServer:/cvs co TeamModule`

The <user> and <ServerName> are as above and the <ServerModule> is the name of the **Module** you will use in the repository defined on the server.

The above command will create a subdirectory in your current working directory, called <ServerModule>.

It creates local copies of all files contained in the CVS module found at <ServerName>:<PathToRepository>.

Test the Client – Server

Once the CVS set up has been completed, verify that CVS operates correctly before you attempt to use it from within EA

Assuming that you set up your local workspace by checking out module <ServerModule> into the directory “c:\myCVSWorkspace”,

To perform a manual test using CVS:

- At the DOS prompt , execute the following CVS commands;

```
Cd \myCVSWorkspace\<ServerModule>
```

- In this directory create a test file, eg. **Test.txt** - this can be done with the command “echo ‘any old text’ >Test.txt”.

Execute the following CVS commands;

```
cvs add Test.txt  
cvs commit -m"Commit comment" Test.txt
```

```
cv$ update Test.txt
cv$ edit Test.txt
cv$ editors Test.txt
```

The "editors" command should produce output like the following:

```
Test.txt myUserID Tue Aug 9 10:08:43 2009 GMT myComputer
C:\myCVSWorkspace\TeamModule
```

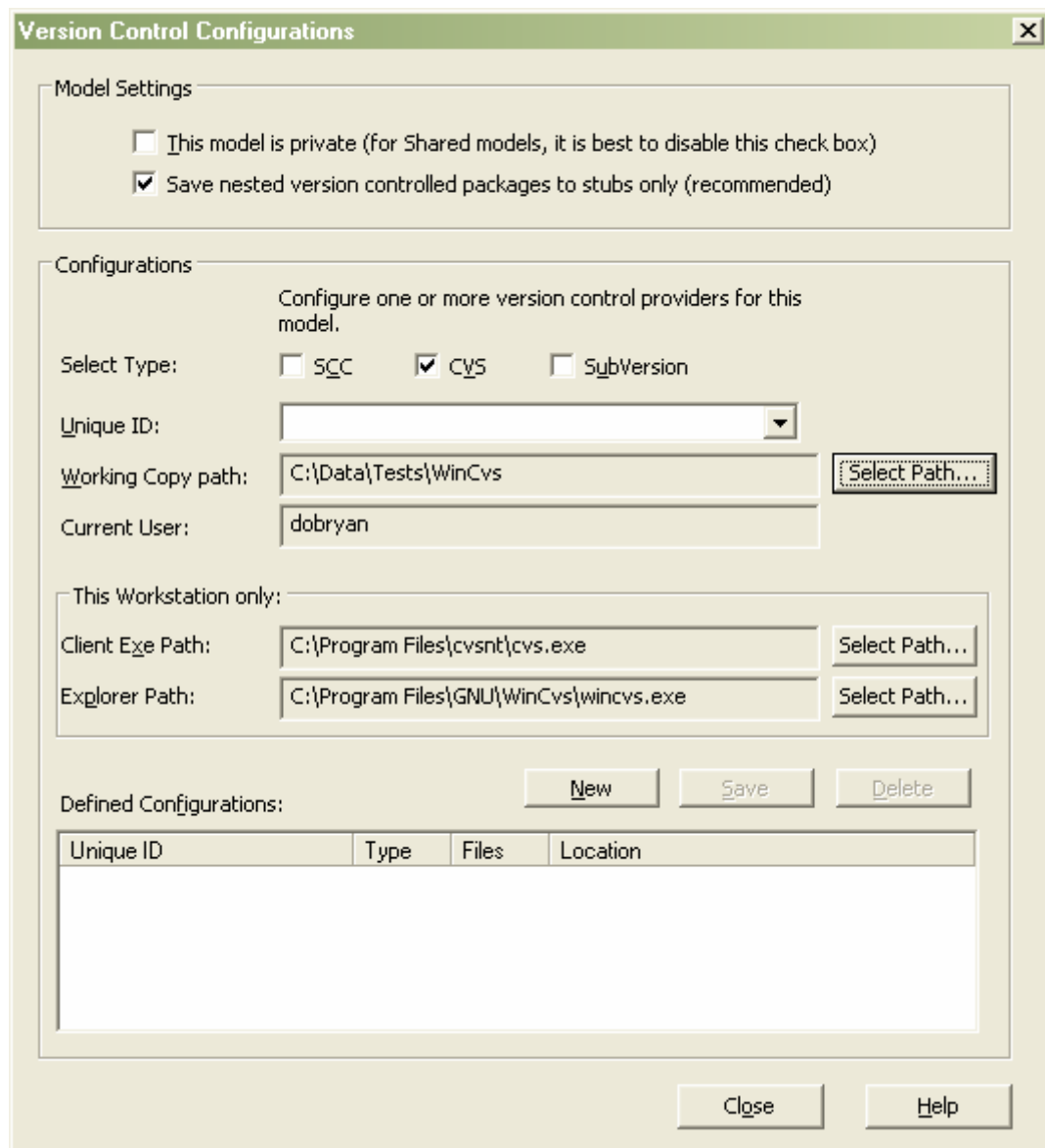
Take note of the **myUserID** that follows the filename. EA must find and use this ID when you create your Version Control Configuration (see example dialog below).

Configuring EA

Once you have confirmed that the CVS client is set up and operating correctly, you are ready to define a Version Control Configuration within EA that will use CVS.

To do this:

1. Open or create the EA package model that you wish to place under version control.
2. From the *Project | Version Control* submenu, select *Set Version Control Options*.
3. Press the *New* button and then ensure that the *Select Type* check box is set to *CVS*.



4. In the *Working Copy path* field browse using [*Select Path...*] to locate the local CVS working folder (i.e. as defined above as `c:\myWorkspace\ModuleName`).
- Example:** From screen shot above: `C:\Data\Tests\WinCvs\`
5. The name in the *Current User* field should reflect the user name used to log in to the remote CVS repository, if this does not happen it indicates that there is a problem with the configuration.
 6. In the *Client Exe Path*, browse using [*Select Path...*] to locate the executable.
 7. In *Explorer Path*, use [*Select Path...*] to locate the executable (i.e. WinCVS.exe).
 8. Press the *Save* button to confirm the configuration.

Next, launch EA and carry out the process listed in: [Testing the installation](#).

Some CVS technical Points:

Connection Methods:

The connection method that you use for CVS is not really relevant to EA. So long as you can successfully issue CVS commands from your CVS working copy directory, without having to supply your CVS userID and password for each command, then EA doesn't care what connection method you use, or where your server and repository resides.

The connection method, whether it is :pserver:, :sspi: or :ssh:, is more of a networking issue, and affects how the CVS server process is run, the userID it assumes and its access rights to various files. There is plenty of information available on the web about how to set up CVS. This is a subject that is quite independent of EA.

The CVSNT page has a link to this page:

http://www.devguy.com/fp/cfgmgmt/cvs/cvs_admin_nt.htm, providing information on installing CVSNT. Look for "Which protocol should I use?".

UserName & Password

When EA issues commands to CVS, it does not supply any username. When you set up CVS, the step where you checkout a CVS module and create your local working copy, should create a subdirectory in your working copy, named "CVS". This subdirectory will contain a file named "Root" which contains your CVS userID and connection method. EA reads this file "Root" and extracts from it, your CVS userID. (If EA does not find the file "Root", it retrieves the Windows userID, but this is NOT the correct ID unless you are using a local repository.) EA uses the userID it finds, to determine whether or not it is you that has an EA package file checked out. You will not be able to modify packages in EA, unless EA has retrieved your correct CVS userID.

EA issues commands to CVS as though it is a user, issuing commands at the command line, from the working directory. If you can use CVS from the command line, then EA will also be able to use it. The only consideration here is that the process it spawns to do this is NOT visible to the user. Therefore, you must have a CVS configuration that does not request passwords from the user, or EA will appear to freeze while the CVS process waits for the user's input.

Subversion

Note: This document is available prior to the release of EA version 6.0. Subversion support will be available in EA version 6.0.

Subversion version control consists of two key components;

1. The server, that creates and manages the repository.
2. The client, that provides the interface through which users add, check in and check out files to/from the repository,

To set up version control within EA, first you need to install your Subversion server on a server machine and then install the Subversion client on each user's workstation machine.

Official Subversion documentation can be found here:

<http://svnbook.red-bean.com/en/1.1/index.html>

Executable files for Subversion can be obtained from here:

http://subversion.tigris.org/project_packages.html#binary-packages.

Note: Enterprise Architect requires Subversion 1.2 or later.

The following are some steps that must be completed before you can use Subversion with EA. Refer to your Subversion documentation for details.

Server Setup

Have your system administrator obtain and install the Subversion server. They will need to create a Subversion repository to use with EA. For more information on repository configuration see:

<http://svnbook.red-bean.com/en/1.1/svn-book.html#svn-ch-5-sect-6>

Chapter 6 in the official Subversion documentation (see above) provides guidance on how to configure the server for different methods of access by the client. Secure connection methods are also covered in this section.

Your administrator should setup **Userids** and passwords for all the users that will access the repository. Your admin should then provide all users with the “**Path to the Repository**”, and ensure that they can all connect.

Note: Subversion repositories can be accessed through many different methods—on local disk, or through various network protocols. A repository location, however, is always defined using a URL. See the Appendix: [Subversion URLs](#)

A Subversion server *Repository* once setup, will contain *Subtrees*. A key step when installing the server side is to create a subtree in the repository for the EA model that you are wanting to version control. Although it actions a process on the server, it is initialized on a client machine.

With Subversion we recommend that each new EA model being added to version control should have a separate repository sub-tree created on the server.

Creating a New Repository Sub-tree on the Server

Although this sets up the sever repository, it is done using a client, so make sure you install a client to perform these tasks.

To create a repository sub-tree perform the following:

1. On a client machine create a temporary directory structure to import into the Subversion repository on the server. This will initialize the repository sub-tree for this EA model.

The directory structure should look like this;

```
tempDir
└─<EA_Model_Name>      `Example C:\tempdir\myEAModel
```

2. Open a command prompt, navigate to directory tempDir
3. Issue the command:

```
svn import <ParentDirectoryName> <RepositoryURL> -m "A meaningful comment"
```

Example:

```
svn import c:\tempDir svn://SVNServer:3690/ -m "New import"
```

As per step 1 above, if c:\tempDir contains the model sub-directory *EA_Model_Name*, then that sub-directory is created in the repository.

Note: that after the import is finished, the original tree is *not* converted into a working copy. To start working, you still need to **svn checkout** a fresh working copy of the tree.

4. You can now delete the directory tempDir and all its contents.
5. For further information see:
<http://svnbook.red-bean.com/en/1.1/svn-book.html#svn-ch-5-sect-6>

Client

You will need to install the Windows Subversion Client executables on the client machines running Enterprise Architect. See the reference to Subversion [executable files](#) above.

Create a Local Working Copy

Once you have created on the server, a sub-tree in the repository for this model, you are ready to create the local Working Copy.

The Working Copy is the directory where EA will create XMI files when it exports the version controlled packages. It is these exported package files that are placed under Subversion control.

To create the local working copy:

1. Choose a suitable directory on your system to create your Subversion Working Copy.
2. Open a DOS command line window.
3. Navigate to the directory that will hold your Working Copy directory.
4. Check-out the model's sub-tree from the repository, with the following command:

```
svn checkout <repositoryURL>/<EA_Model_Name>
```

Where:

- <repositoryURL> is the URL for the repository on the server.
- <EA_Model_Name> is the directory name that you used in setting up the repository sub-tree above.

Example:

```
svn checkout svn://SVNServer:3690/myEAModel
```

Test the Client – Server

Once the Subversion set up has been completed, you should verify that it operates correctly before you attempt to use it from within EA. When carrying out this test you must be able to commit files to the repository, without being prompted for ID or passwords.

To perform a manual test using Subversion, carry out the following:

1. Create a temporary text file to be version controlled in your working copy folder.

2. Open a command prompt in DOS.
3. Navigate to the directory that will hold your Working Copy directory.
4. Add and commit the temporary file to the repository using the following commands:

```
svn add <fileName>
svn lock <fileName>
```
5. Now, update the file from the repository, lock the file. To do this use the following commands:

```
svn update <fileName>
svn lock <fileName>
```
6. Edit and save the file using an editor.
7. Now commit the file once more using the following command:

```
svn commit <fileName> -m"A meaningful comment."
```
8. To review the Subversion status on this file use:

```
svn <fileName> -ur
```

This should show two revisions of the file.

Configuring Version Control with Subversion

Having installed Subversion, both server and client, and created a local working copy from a repository sub-tree, you can now set up your EA model.

To apply version control to your EA model, perform the following steps:

1. Launch EA and open the model for which this Working Copy was created.

- From the main menu select: **Project | Version Control | Set Version Control Options**. The following dialog will open:

Version Control Configurations

Model Settings

This model is private (for Shared models, it is best to disable this check box)

Save nested version controlled packages to stubs only (recommended)

Configurations

Configure one or more version control providers for this model.

Select Type: SCC CVS SubVersion

Unique ID: Svn_demo

Working Copy path: C:\Data\VersionControl\WinCvs

This Workstation only:

Client Exe Path: C:\Program Files\Subversion\bin\svn.exe

Defined Configurations:

Unique ID	Type	Files	Location
-----------	------	-------	----------

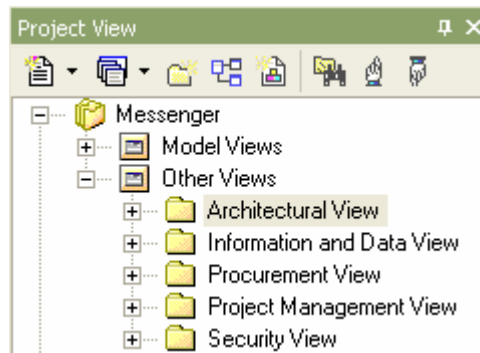
- Press the **New** button
- Set the **Select Type** check box to “*Subversion*”.
- In the **Unique ID** field give the configuration an appropriate name.
- Now click the **Select Path** button to specify the path to your Working Copy directory.
Note: The Working Copy directory was created in step 4 of the section “Creating a Local Working Copy” above.

7. Click on the **Select Path** button to specify the path for your Subversion client executable.
8. Click on the **Save** button to save this configuration.

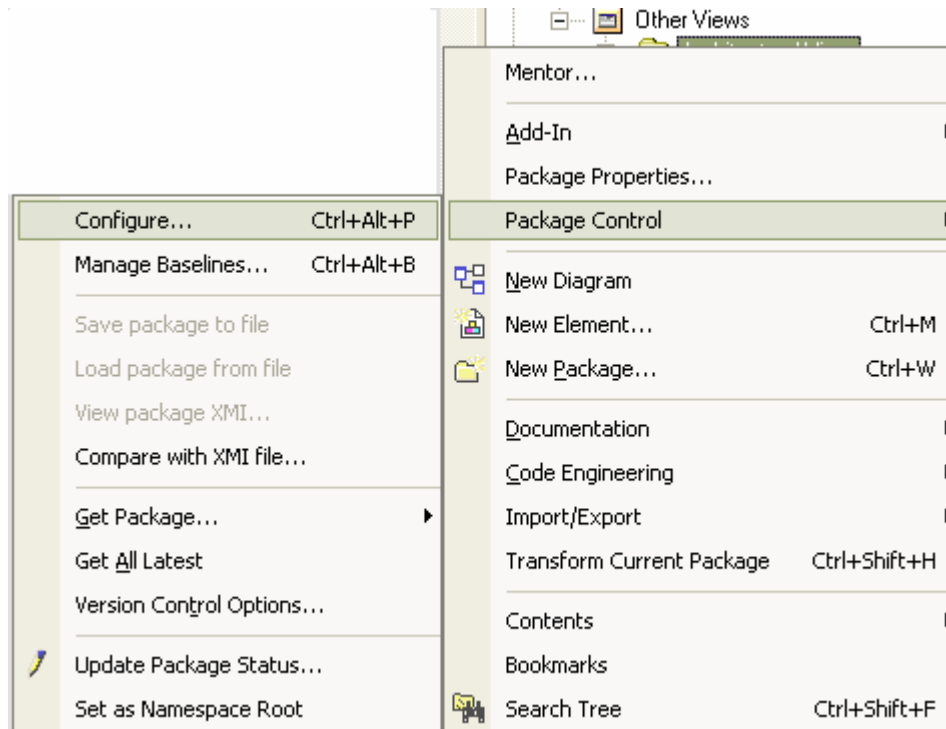
Testing the installation

Once EA has been configured you can then place an EA packages under Version Control. To do this:

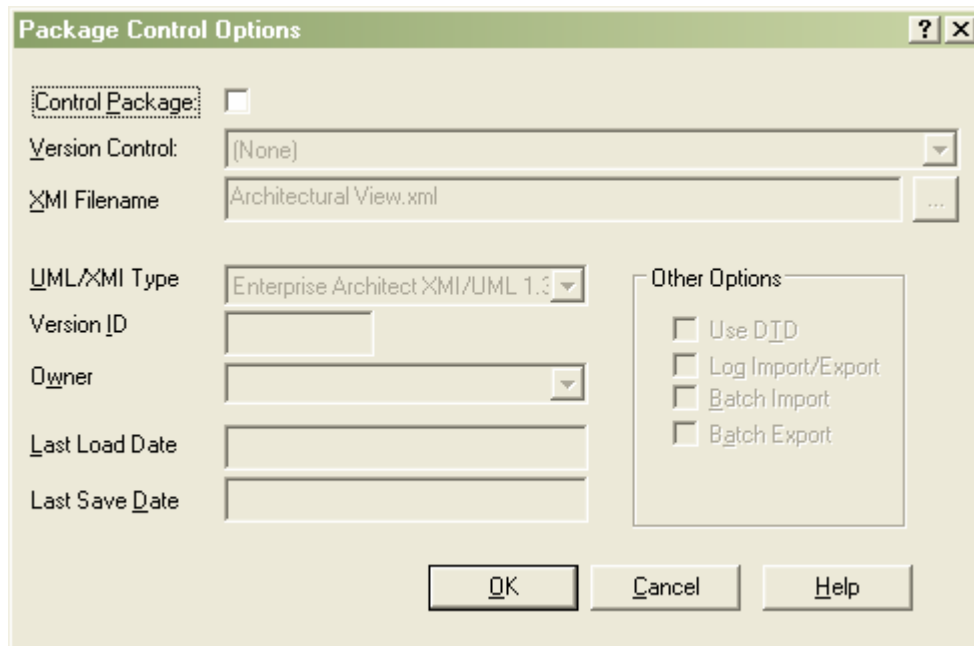
- Select a package from the Project view:



- Right-click on this package and from the context menu select **Package Control | Configure**

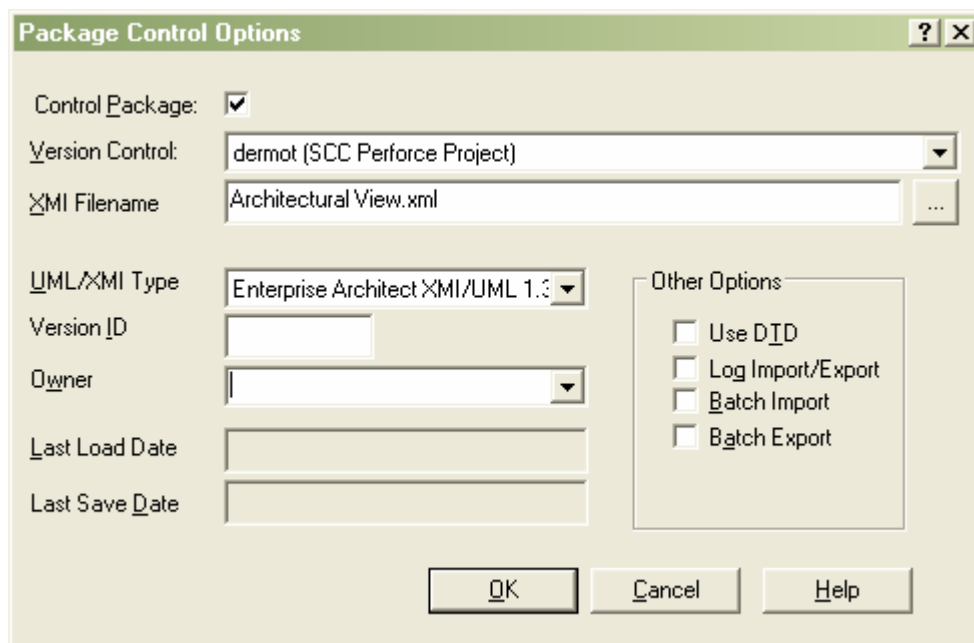


You will then be prompted with the following window that allows you to configure the options for version control on the package selected.




To set the required options for a simple export you can simply configure the following:

- Tick the **Control Package** checkbox to indicate this is a controlled package.
- Select a **Version Control** configuration from the **Version Control** dropdown list; this connects this package to a specific version control configuration.



Unless any changes are required, the remaining entries can be left as default entries.

This has set up the package as a version controlled package. You can then check that package indicates that it has been set to version control with the icon: .

Once this is complete, it can be verified using your local version control software's GUI client to see that the package has been placed under version control.

Note: Users should not modify the version control state of the XML files by directly using the version control product outside of EA.

For more information on using particular functions associated with the version control see the guide to this in EA's Help file.

Appendix:

1. Version Control Deployment schemas

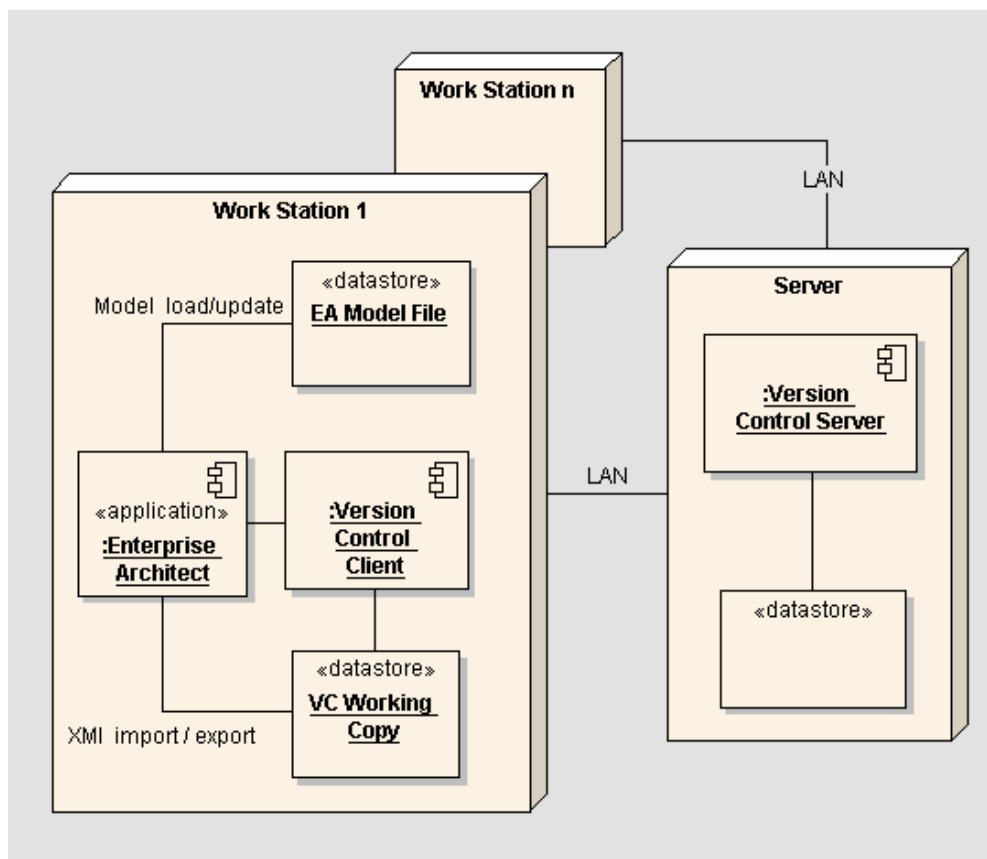
As outlined above there are a number of different schemas for implementing version control with Enterprise Architect. This Appendix covers three broad schemas that may be used. There can be many variations on this type of schema.

These three schemas are as follows:

1. Private Model
2. Shared Model
3. Work Group Model

Private Model

The private model is a simple structure each user hosts their EA model on their own machine and accessing a common version control server.



The Version Control Server (and repository) set up on a server machine, which accessible by all workstations.

There can be any number of workstations. Each workstation has

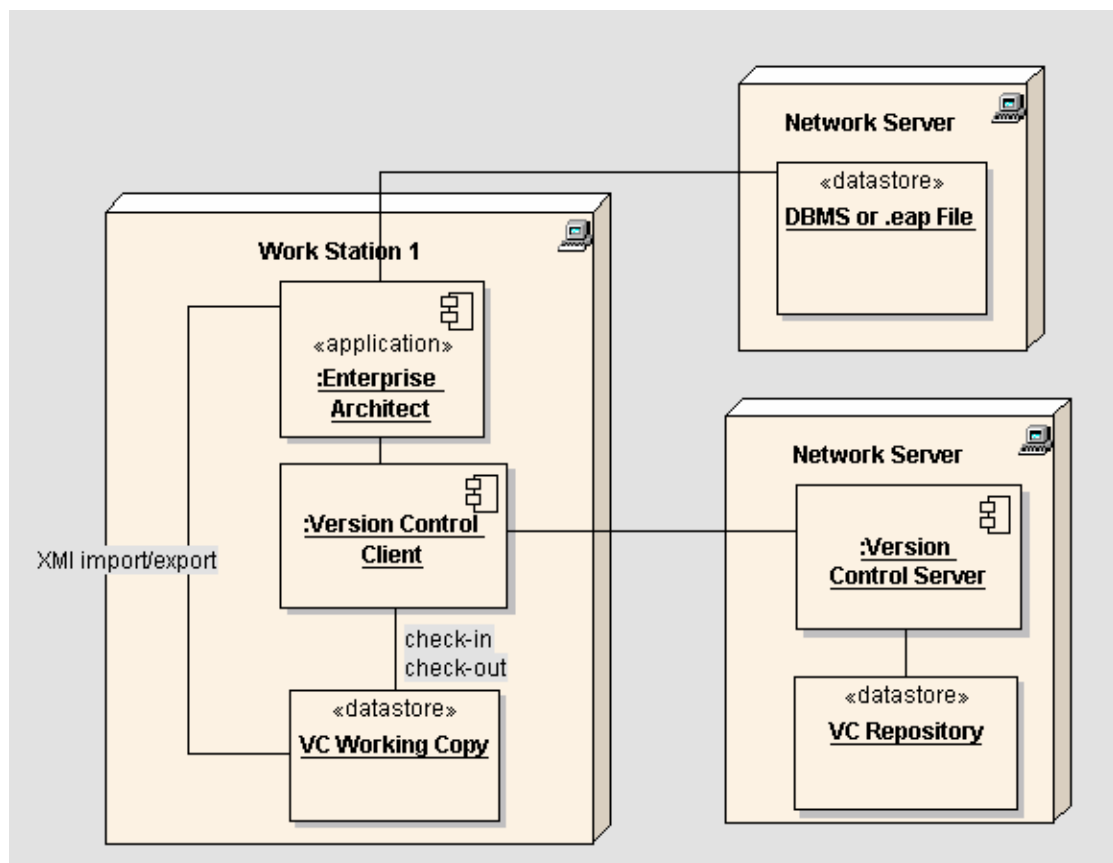
- 1) A private copy of the (same) EA model.
- 2) A private local workspace that contains module of interest in the Version Control repository.

The model is kept up to date by retrieving the latest versions of the package files from Version Control.

Each user's access to edit the model's packages is managed by EA, through the Version Control system.

Shared Model

The simple and common scenario is where users are setup to share a central .eap file or DBMS database. This configuration allows users to see other users' packages without explicitly having to retrieve them. Version control regulates access to packages, and maintains package revision history.



The Version Control Server (and repository) is set up on a server machine, accessible by all workstations.

There is a single copy of the EA model, accessible by all users. This is held either in a DBMS repository or as an .EAP file stored on a shared network drive.

There can be any number of workstations. Each workstation has:

- 1) Accesses to the shared EA model.

- 2) A private local workspace that contains the module of interest from the Version Control repository.

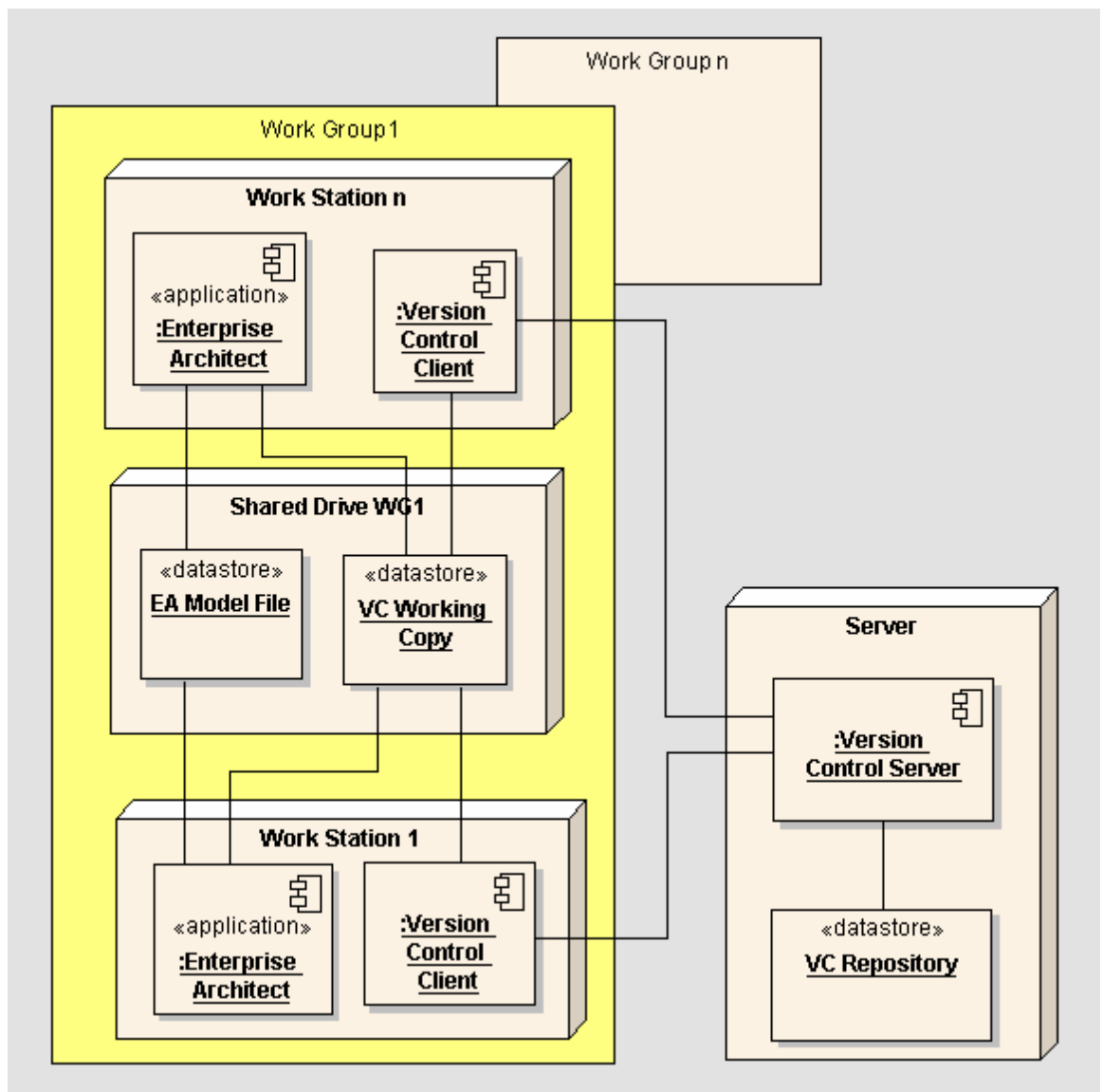
The model is immediately updated as users make changes.

Each user's access to edit the model's packages is managed by EA, through the Version Control system.

Work Group Model

This schema uses a Version Control Server (and repository) set up on a server machine, accessible by all workstations.

A number of work groups are set up where all users within a given work group appear to the Version Control system, to be the same user.



Each work group:

- Behaves like a single user from the Private Models deployment above.

- Has its own private copy of the EA model that is shared within the work group.
- Has just one local Version Control work space.

Within the work group the copy of the model and the local Version Control work space are accessed by all work group members. All work group members use the same Version Control UserID.

Note: This schema is dependant on the Version Control keeping a UserID independent of the PC. CVS does support this but it may not work with SCC products, as the PC's machine name often forms part of the Version Control UserID.

It is envisaged that the size of the work group is small, with high levels of communication between its members, for example, all working within the same room.

EA manages edit access to the packages at a work group level, treating each workstation within a work group as the same user.

3. Subversion URLs

Subversion repositories can be accessed through many different methods—on local disk, or through various network protocols. A repository location, however, is always defined using a URL.

Table 1 describes how different URL schemas map to the available access methods.

Table 1. Repository Access URLs

Schema	Access Method
file:///	direct repository access (on local disk)
http://	access via WebDAV protocol to Subversion-aware Apache server
https://	same as http://, but with SSL encryption.
svn://	access via custom protocol to an svnserve server
svn+ssh:// /	same as svn://, but through an SSH tunnel.

For more information on how Subversion parses URLs, see [the section called “Subversion Repository URLs”](#)

4. Version Control Products

The following is a list of known SCC compliant version control products. Ones that have been tested for use with Enterprise Architect are marked:

Software	Tested	Software	Tested
AccuRev	Yes	Serena PVCS	
CA Harvest		SourceGear Vault	Yes
ClearCase	Yes	Source OffSite	Yes
IBM TeamConnection		Starbase Versions	
Intersolv PVCS		StarTeam	
MKS Source Integrity	Yes	Surround SCM	
Perforce P4	Yes	TeamCoherence	
PVCS		VisualAge TeamConnection	
QVCS		Visual Source Safe	Yes

There are also numerous SCC Wrappers for CVS clients that are available. Given the direct CVS link, these are not needed but can be used. The packages are:

PushOK for CVS
Zues

There is an interface known as Igloo – we have not been successful in running EA with this.

3. Links

CVS links

Overview documentation:

<http://cvsbook.red-bean.com/cvsbook.html#Starting%20A%20Repository>

<http://cvsbook.red-bean.com/cvsbook.html#Accessing%20A%20Repository>

<http://cvsbook.red-bean.com/cvsbook.html#Starting%20A%20New%20Project>

<http://cvsbook.red-bean.com/cvsbook.html#Checking%20Out%20A%20Working%20Copy>

Product downloads:

<https://ccvs.cvshome.org/servlets/ProjectDocumentList?folderID=80&expandFolder=80&folderID=0>

Windows CVS interface: <http://www.wincvs.org/> .

Windows GUI client <http://www.tortoise cvs.org/>

Documentation: <https://www.cvshome.org/docs/manual/cvs-1.11.20/cvs.html>
<http://cvsbook.red-bean.com/cvsbook.html>

Other CVS links:

http://www.devguy.com/fp/cfgmgmt/cvs/cvs_admin_nt.htm

These instructions describe how to install a CVS server on Windows. Client setup is also discussed.

Typical users of CVS do not need to read this page -- instead, the [CVS overview](http://www.devguy.com/fp/cfgmgmt/cvs/) (<http://www.devguy.com/fp/cfgmgmt/cvs/>) is a more suitable page for CVS users. System administrators who install CVS are also encouraged to read the CVS overview page.

<http://www.devguy.com/fp/cfgmgmt/tools/install.htm#CVSCLIENTINSTALL>

CVS Client Installation on Windows

See also the [WinCVS QuickStart Guide](http://www.devguy.com/fp/cfgmgmt/cvs/startup) :
<http://www.devguy.com/fp/cfgmgmt/cvs/startup>

If you want to use the command line on Windows:

<http://www.devguy.com/fp/cfgmgmt/cvs/>

Look for the section **CVS Server Installation**. It has a number of links for various Operating Systems.

http://devguy.com/fp/cfgmgmt/cvs/cvs_ssh.htm

CVS servers can transfer data via the secure shell ([ssh](#)) protocol. SSH is preferable to pserver due to its superior encryption of passwords.

Subversion links

Product downloads: http://subversion.tigris.org/project_packages.html#binary-packages (Must use at least version 1.2)

Install instructions: <http://svnbook.red-bean.com/en/1.1/ch01s07.html>

Documentation: <http://svnbook.red-bean.com/> (latest available here is version 1.14).

The Subversion FAQ list: <http://subversion.tigris.org/faq.html>

<http://tortoisesvn.tigris.org/> TortoiseSVN is a [Subversion](#) client, implemented as a windows shell extension.