Deployment of Enterprise Architect

By Dermot O'Bryan

http://www.sparxsystems.com

Table of Contents

Introduction

Sparx Systems Enterprise Architect is designed for use within large corporate environments. As a scalable modeling platform, Enterprise Architect provides a range of deployment options to accommodate the variety of modern enterprises.

This whitepaper discusses the options available for enterprise-wide sharing of model information.

Typical Enterprise Structures

The following are typical organizational structures encountered when deploying modeling software for a large corporation. Each has its own needs, and software configurations.

- Single Site
  - One building with a large user base on a LAN
  - Multiple buildings with a high speed WAN
- Single Site – Multiple Projects
  - Multiple repositories
  - One repository – many projects
- Multiple Sites
  - Multiple sites with a low speed WAN
  - Multiple sites no specific WAN
  - One large site, i.e. Head Office with multiple external contractors working on customer sites.

For a simple table covering the options for optimizing performance for these different structures; see the appendix: Optimizing Performance

Deployment Features

Enterprise Architect has a number of features designed to simplify deployment across large organisational structures. In combination, these features give users the flexibility to create their own specific layout while allowing for expansion.

These features are dependant on the edition of Enterprise Architect installed, as well as the data storage used. The model's data store is referred to as a "Repository". For more information on the features and repository types available in the different Enterprise Architect editions - see: http://www.sparxsystems.com.au/products/ea_editions.html

Following are features that assist enterprise-wide deployment of Enterprise Architect:

- Choice of repository:
  - EAP (file-based) repository
    - Enables rapid model setup, offline (local) work and replication
  - DBMS (server-based) repository
    - Robust
    - Secure
    - High volume usage
- Data Transfer
  - Facilitates model transfers between EAP files ⇔ DBMS repositories
- Package Control
  - Supports sharing of model branches (packages)
- Version Control
  - Management of controlled packages in team-based development
- Security Permissions for Model Authors
  - Allows granular locking of packages and elements
- Remote Installation

- Automated installation of Enterprise Architect across networks

Choice of Repository

The Enterprise Architect Corporate edition can store models in a file based repository, or a SQL-based DBMS repository. There are pro's and con's for both solutions.

A simple comparison of file-based (EAP) and DBMS repository types is listed below:

Function	EAP	DBMS
Replication	Yes	NO
Number users	1..~10	Unlimited
Non-Corruptible	No	Yes

EAP file Repository

EAP files (Enterprise Architect Project) are based on the Microsoft Jet 3.5 database engine (MS Access '97 format .mdb[1]). Enterprise Architect also supports the Jet 4.0 database engine. For information on using Jet 4.0 (e.g. for Unicode Support), see the appendix: Using Jet 4.0

The benefits of this repository type are:

a) Replication of the repository.

b) Simple file access across a shared network drive.

The limitations of this repository type to consider are:

a) Concurrent access is limited to small groups of users.

b) Under rare circumstances, data corruption may occur if a network/power failure occurs while editing.

DBMS Repository

Using a DBMS model repository overcomes the limitations of file-based repositories. Typically, dedicated DBMS servers provide faster response times for a larger user base than the Jet-based EAP files. Further, any network errors are handled by the ability of the DBMS server to roll back transaction failure caused by external conditions.

Enterprise Architect supports the following types of DBMS server repositories:

- MS SQL Server
- MySQL
- Oracle9i and 10g
- PostgreSQL
- MSDE
- Adaptive Server Anywhere
- Progress OpenEdge

For details on configuring DBMS servers for Enterprise Architect model repositories, see http://sparxsystems.com/resources/corporate/

[1] Note: MS Jet 3.5 is not supported in Windows-64 bit operating systems.

Deployment Schemas

The following section discusses deployment schemas based on the typical enterprise structures outlined in the introduction.

> **Note:** For information on remote (unattended) installation across a corporate network see the appendix: 7: Remote Installation of EA

Single Site

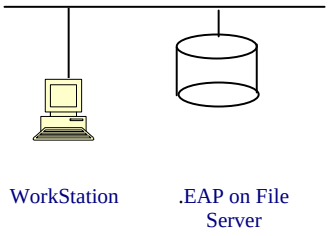
The most common scenario is a single site with a single, shared repository catering for any one specific project. In larger organizations however, projects will typically share a set of common aspects. These can range from common procedural code, through to common report templates that meet corporate standards. The following covers these scenarios.

Single Projects

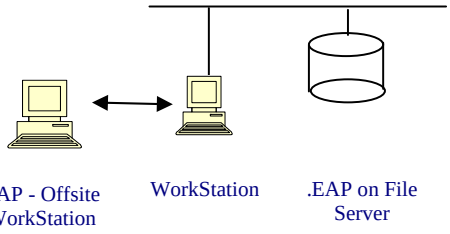
Using EAP files on a shared drive

This is a simple setup which can be used for groups of up to 10 users. A repository file is stored on a file server, which is accessible to users across a local area network (LAN). There is further scope to allow for replication to take place for any repositories that are taken offsite and returned. As noted earlier, limitations of a file-based model repository apply to this approach.

The diagrams below show typical configurations.



Common Drive .EAP File Repository



Offsite Replication Using .Eap

Using a DBMS repository

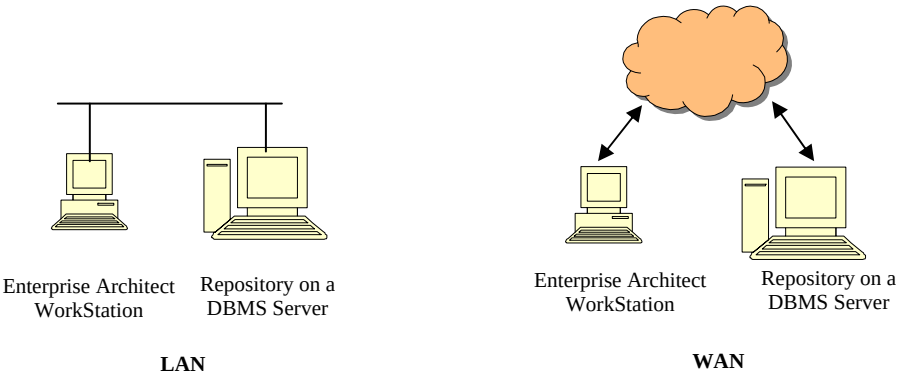
The DBMS repository offers a far more robust environment in situations where there are a large number of users.

Although DBMS repositories do not offer the replication services provided with the .eap structure, this functionality is covered by the Package Control and Version Control facilities available in the Corporate edition. See: Replication versus Package/Version Control.

For very large repositories the DBMS interface supports a "Lazy Load" option. With this enabled, when opening a repository, only the data for the visible portion of the project tree is loaded. This allows a faster load of the model, at the expense of small delays on the initial access of sub-packages.

Enterprise Architect also supports a 'WAN Optimizer', for connecting to a DBMS repository across a WAN. The Optimizer provides data compression between the main

site server and the off-site Workstations. For more information on this option see the Appendix: Optimizing Performance.



Multiple Projects

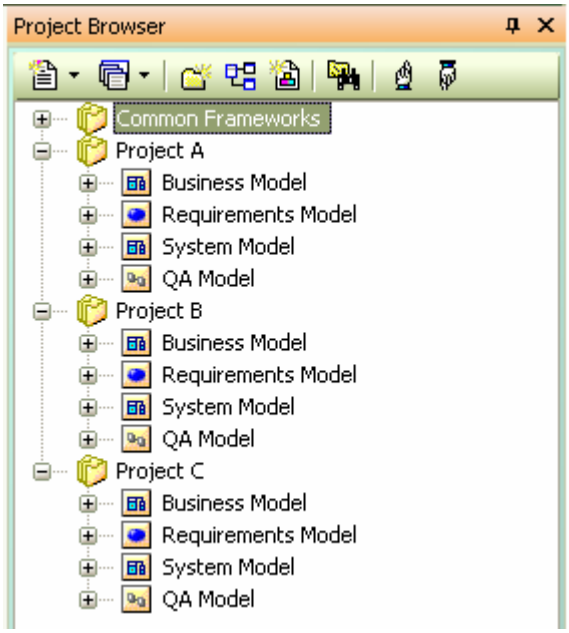
When there are multiple projects under development, it is possible that there may be core code as well as settings, reports, etc., that are common between these projects. Two approaches can accommodate such a scenario:

One Repository – Multiple Projects

Use one model repository with multiple Project Root Nodes (see figure below). Each Root Node represents a separate project. Commonality between projects (frameworks, foundation classes etc.) can also be captured under a Root Node.

Using a single repository will ensure that common code and references need only be changed once to affect all projects.

Additionally, Enterprise Architect's security can be used to restrict a user's access to certain areas (for example root nodes) that they are involved in. Elements can be accessed for use in diagrams, etc.; however, data in these Elements can only be modified by the group that has security access to the package in which they are contained.



Project Browser view of a single repository containing multiple projects.

Multiple Projects – One Repository per Project

In this scenario, each project uses a single multi-user repository. There are two types of import and export available, one to exchange packages (i.e. common frameworks) and the other to exchange the repositories reference data (i.e. type definitions and report templates etc.). It is recommended that a master repository is used to maintain a single source for the common project data.

For common packages, such as frameworks etc; simple XMI import/Export, Package Control or Version Control can be used to interchange work between the local repositories (single or multi-user) and the master repository.

The reference data on local repositories (type definitions, macros, security, report templates etc.) can be periodically updated from the master repository using the main menu option: Tools | Export Reference Data. See the appendix: Reference Data Import Export

Large Project - Branching and Merging Multiple Phases

In large "phased" development projects, the design of the next phase 'Branch' can be carried out in a copy of the 'Base' repository, with the Base model being used in development. After a phase of design work, 'Baselines' can be created against the Branch model and using 'Baseline, Difference and Merge' the Branch can then be merged back into the Base model.

For more information see the appendix: Baseline Difference & Merge

Multiple Site Scenarios

Where multiple sites are linked with a high speed WAN, Enterprise Architect can be used, as outlined above as a standard single site environment. This section focuses on sharing models across multiple sites with minimal network connectivity. Two common scenarios are:

- Multiple Sites with a low speed WAN

- Multiple sites no specific WAN

In both cases, models are shared across multiple sites using a central repository with one, or many, off-site repositories accessing and periodically updating this central repository.

Common configurations are:

1. A central .eap file with replication to and from offsite .eap files.
2. A central .eap or DBMS repository, using Citrix or Terminal Server emulation on all offsite machines.
3. A central DBMS repository with the external sites also requiring DBMS repositories.
4. A central DBMS repository with offsite users working with .eap files.

1. Central EAP file

Two possible scenarios are:

a) Small central user base with offsite users.

b) Small central site usage with multiple small sites.

In the first case replication between the main repository and offsite users is a simple solution.

Replication is a useful option in the second scenario, where there are multiple sites working on a project with defined sections, such as organizational units, and one central unit that is largely involved in compiling the organizational units and producing reports, etc. This does require that the number of users in each of the units is within a reasonable range for using the .eap files.

In order to use replication; the central unit has to set up a master repository with a set of slave repositories. For more information on this, see: Replication vs. XMI Import/Export

2. Citrix or Terminal Server Emulation

In cases where the connection speed between sites is reasonable (20–60ms latency), but the volume of throughput data is restricted (i.e., a 64k bandwidth), then a terminal server type application can be used to emulate Enterprise Architect on remote work stations.

3. A Central DBMS Repository

With DBMS repositories, replication is not available. This function is replaced with the ability to set up Package Control or Version Control.

The Package Control provides a mechanism for 'externalizing' or passing on parts of an Enterprise Architect model in the form of XMI files. Using controlled packages, it is possible to support a widely distributed development by batch import/export of these packages. The central repository can perform a batch upload (import), or a batch distribution (export), to any external repositories.

This can be further extended to include Version Control. For details on setting this up, see: Package Control & Version Control below.

Note: For offsite transfer of data, Version Control is recommended over batch control unless:

a. The sites have specific sections that are only modified by that site (e.g. one site is the code area; the other site performs testing and only modifies packages defining the testing).
b. The sites are in opposite time zones where there may be no conflict resulting from parallel changes.

For more information on this, see the whitepaper on Version Control:
http://www.sparxsystems.com.au/resources/whitepapers/index.html

4. Multiple Offsite Users

It is not uncommon to have one large site acting as Head Office with multiple external contractors working on customer sites. In this scenario, the DBMS server is used for the central repository, while it is recommended that offsite users work with the .eap repositories. In this case, both Version Control and Package Control can be used to exchange data.

Package control can be used to enable a batch process of importing and exporting data, whereas Version Control uses a locking mechanism to control the checking-out and checking-in of data.

Appendix

1: Optimizing Performance

When sharing large models over a network (LAN or WAN), some considerations should be made to optimize performance. In general, performance depends on:

- The model repository type used (EAP or remote DBMS).
- Whether version control integration is used to manage editing of shared models
- The network response time, which is based on:
  - Network load
  - WAN latency (a low latency WAN connection has under 40-50ms latency)
    Note: best used in conjunction with the WAN Optimizer (see below).

The following tables indicate which network and repository type configurations allow for high performance. The first table considers repository configurations that do no not use Enterprise Architect's version control integration capability, while the second table allows for version control.

Configurations without version control integration

Repository Type	Network Type	Users	Optimal	Comments
EAP	Local	1~10	✓	Fastest response times.
EAP	LAN	1~10	✓	Performance depends on LAN load.
EAP	WAN	1~10	✘	Not recommended. Use a DBMS instead. Otherwise, a low latency WAN is required.
EAP Replication	Local	∞ *	✓	Requires EAP files to be merged at a central point. Good for compilation of data collected offsite using local Eap files. See section on Multiple Offsite Users
DBMS	LAN	∞ *	✓	Good for large teams. LAN load varies by DBMS type. (MySQL & SQL Server models seem to cause less load than Oracle).
DBMS	WAN	∞ *	✓	Optimal WAN connection: ➢ A low latency WAN ➢ WAN load varies by DBMS type
DBMS Replication	User ⇔DBMS (LAN) DBMS ⇔ DBMS (WAN)	∞ *	✓	Local DBMS performance should be the same as the LAN/DBMS. Requires DB replication. ➢ Oracle and SQL Server both support DB replication. ➢ A DBA is required to configure the DB replication.
XMI Packages (EAP or DBMS)	LAN ⇔ LAN	∞ *	✓	Requires XMI export/import for synchronization (can be automated). See Replication vs. XMI Import/Export

***** *Concurrent user limit depends on physical capacity of database server hosting the repository*

Configurations with Version Control

When using Enterprise Architect with Version Control across a LAN or a WAN, performance is dependent on the Version Control system in use, as well as network response times. Each Version Control system has different response times; however there are a number of options available and points to consider for the best results. Below are comments on these options:

Repository Type	Version Control	Users	Optimal	Comments
EAP - Local	LAN	1~10	✓	Fastest response times.
EAP - LAN	LAN	1~10	✓	Performance depends on LAN load.
EAP - Replication	LAN or WAN	1~10	✗	Not recommended. If local EAP files are posted to a central point and merged using replication, the use of Version Control is difficult and merging may have to be done manually.
DBMS - LAN	LAN	∞ *	✓	Fast, the best option for 10+ users. See DBMS – LAN above.
EAP - LAN	WAN	∞ *	✓	Use Local EAP repositories – global updates are managed using Version Control over a WAN. 1. This is the fastest WAN option depending on Version Control provider. 2. A local EAP may be used at a single user or workgroup level for optimal performance.
DBMS - LAN	WAN	∞ *	✓	Uses shared local DBMS models (on a LAN) connecting to Version Control via WAN. This allows for distributed access to repository data across network locations: 1. This provides the best local EA performance. WAN Delays limited to Version Control updates. 2. This setup avoids excessive DB calls being made across slow links. 3. Data-Compression, if available on the WAN link, will improve the transfer of XMI Version Control files.
DBMS – WAN Replication (LAN connection to local DBMS)	LAN/WAN		✗	Product specific 'Replication' on a DBMS (SQL Server or Oracle) is not recommended as it can be difficult to use alongside Version Control. It requires an expert understanding of the DBMS replication as well as the Version Control system to be implemented.
WAN DBMS	LAN/WAN		✗	Not recommended. Recommended alternatives are: a) EAP – LAN | VC - WAN b) DBMS – LAN | VC- WAN

***** *Concurrent user limit depends on physical capacity of database server hosting the repository*

WAN Optimizer

By reducing the amount of data transmitted and the number of network calls made, the WAN Optimizer significantly improves Enterprise Architect's performance across a WAN connection to a DBMS repository. The *Optimizer* is a lightweight service installed on any server with a high speed connection to the DBMS. For more information on the WAN optimizer see:

http://www.sparxsystems.com/uml_tool_guide/uml_model_management/the_wan_optimizer.html

2: Replication vs. XMI Import/Export

To support a distributed development environment, Enterprise Architect offers two kinds of data interchange. These are:

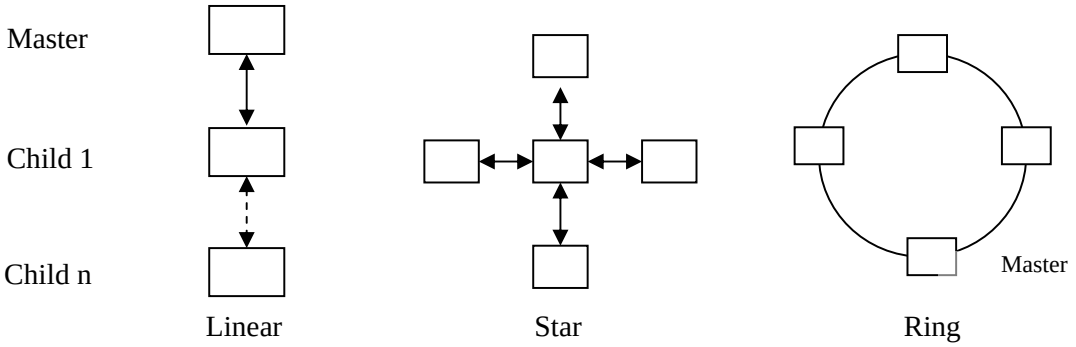
- The replication of .eap repositories
- The use of XMI for Package and Version Control

Replication

Replication is a simple means of ensuring valid interchange between .eap repositories. In practice, a master repository is set up at a central location. Replicas of the master (child repositories) are created and taken off-site for use by consultants. Geographically separated consultants can then update and modify parts of the model using these replicas. On return to the central location, these changes are merged back to the central repository.

Replication can be set up in a number of different patterns (See: MS white papers - Jet 3.5 White Paper, Replication White Paper [2]).

The replication can be in the form of a *linear, ring or star Pattern.* Some of these types of patterns are shown in the following diagram.



The typical pattern is one master with one or multiple slave repositories. See – MSDN replication[3] for more details on these patterns of replication.

For more information on setting up replication in Enterprise Architect, see Enterprise Architect's help topic: Model Sharing and Team Development | Replication.

[2] http://support.microsoft.com/default.aspx?scid=http://support.microsoft.com:80/support/kb/articles/Q164/5/53.asp&NoWebContent=1
http://support.microsoft.com/default.aspx?scid=http://support.microsoft.com:80/support/kb/articles/Q181/3/71.ASP&NoWebContent=1

[3] http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/dnaraccess/html/msdn_replicat.asp

3: XMI – Package control & Version Control

When using a DBMS (or a large .eap file) as a repository, the XMI based Import/Export can be used to export individual or grouped packages using XML. These can then be shared amongst the development team. This approach has several benefits over replication:

1. You can assemble a model from only the parts you need to get your job done.
2. You can assemble a full model if required.
3. You can assemble different package versions of a model for different purposes (eg. Sections that are: customer visible, for internal release only etc.).
4. You can roll-back parts of a model on an 'as required' basis.
5. There is less chance of 'collisions' between developers if each user works on a discrete package.

The process is more tightly controllable using a Version Control system. For more information on how to set up bulk import/export, see Package Control & Version Control below.

3.1: Package Control and Version Control

Enterprise Architect provides two methods of Version Control. These are:

- Package Control
- Version Control

Although both processes are similar in that they can be used for coordinating the sharing of packages between users, Version Control has the added feature of saving a history of changes to Enterprise Architect model packages, including the ability to retrieve previous versions. Version Control however, requires that third-party software be installed and interfaced with Enterprise Architect.

3.1.1 Package Control

Package Control can be used to import or export a batch of packages from one repository to another (i.e. a branch office to a central repository). To set this up, it is necessary to define the set of packages to be included in the batch. These are called 'Controlled Packages'.

The process for doing this is:

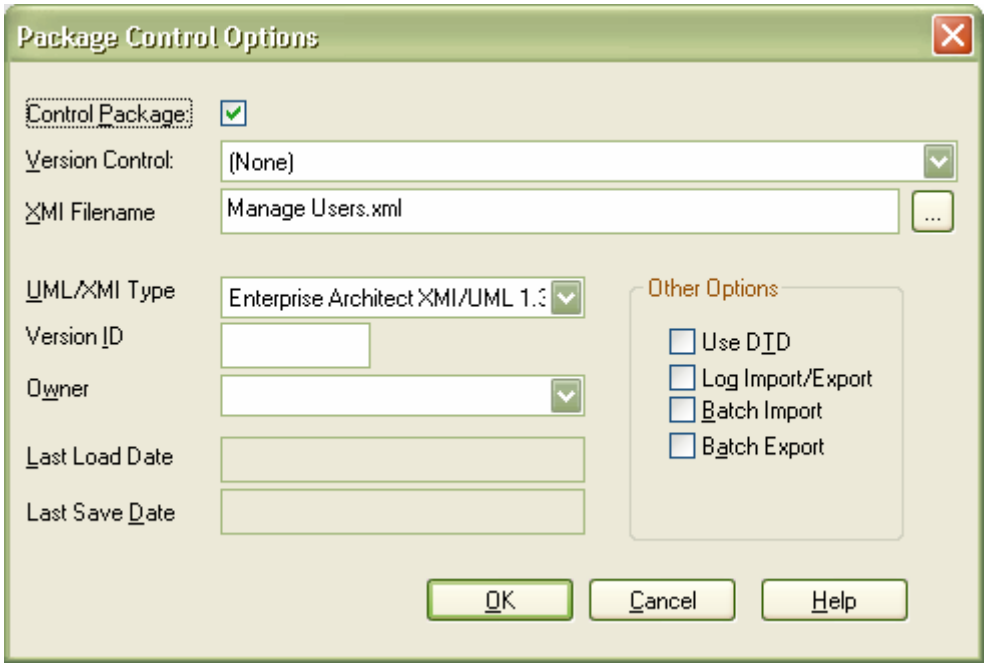
a) Configure Controlled Packages

b) Use the Batch Import/Export

To configure the control packages:

1) Select a package to add to the batch export.

2) From the main menu, select: Project | Version Control | Configure Current Package. This brings up the following window.



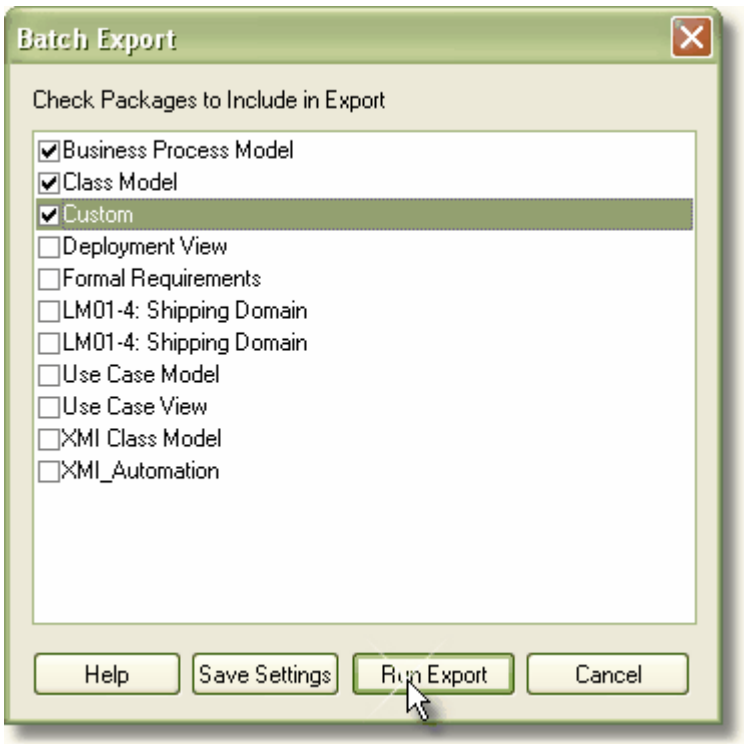
3) Tick the Control Package Option.

4) Press OK.

5) Repeat this sequence for the set of Packages to be exported as a batch.

Once this has been completed there is access to use the batch import/export as follows:

1. From the *Project* | *Import/Export* submenu, select *Batch XMI Export*.



2. In the *Batch Export* dialog, check the packages to include in this export run.

3. Press *Save Settings* if you wish to save this configuration as the default
4. Press *Run Export*.

3.1.2 Version Control

Enterprise Architect supports Version Control of packages and their component sub-packages to a central Version Control repository which is maintained by a third-party version control application. It provides two key benefits:

- Coordinating the sharing of packages between users.
- Saving a history of changes to Enterprise Architect packages, including the ability to retrieve previous versions.

It can be set up using any Version Control software that is compliant with the following standards:

- SCC standards
- CVS
- Subversion
- Microsoft TFS

There are many benefits from sharing a model using Version Control, dependant on the environment and the type of control needed - such as support for different releases and versioning.

There are four basic ways in which Version Control can be employed:

Use	Description
Shared model	Users share a central EAP file or SQL 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.
Duplicate models	An EAP file is created by a single user who configures it for Version Control. The file is then distributed to other users who connect using their own user name. Other users' packages are retrieved using the Get Package command.
Shared packages	Individual users create their own EAP files but share one or more packages by connecting to the .same SCC project and using the Get Package command
Standard packages	A company may have a standard set of packages which are broadly shared (on a read-only basis). Individual users use the Get Package command to connect to the main package.

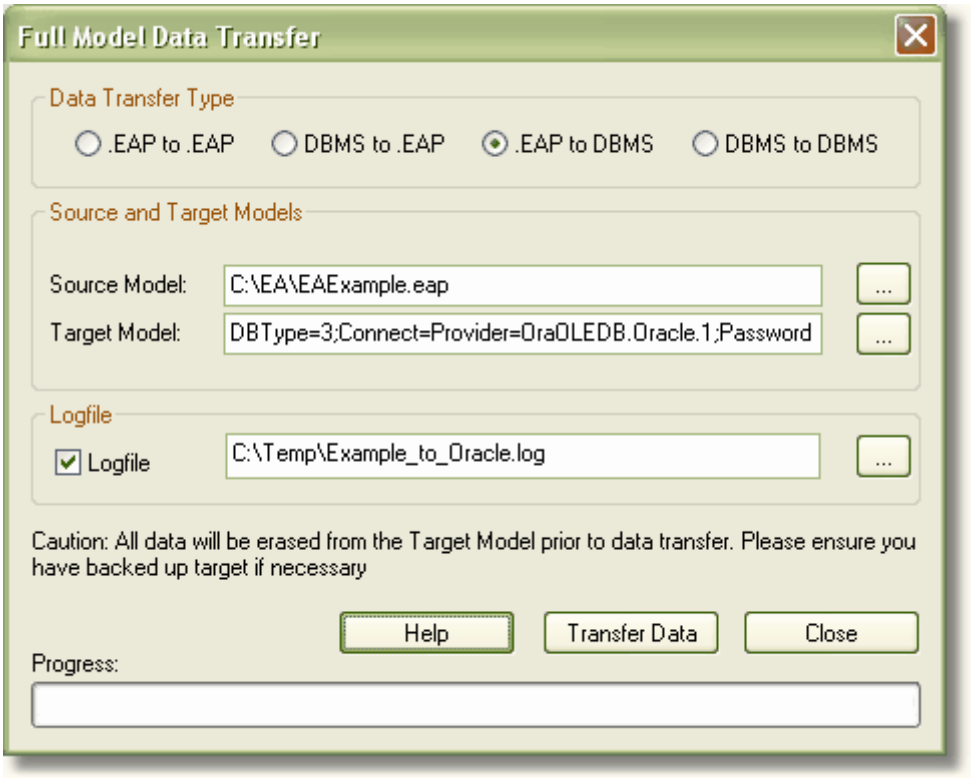
For more information on setting up of the version control systems in Enterprise Architect, refer to the help file under: Model Management | Version Control. The following whitepaper provides more detail on the use of Version Control: http://www.sparxsystems.com.au/resources/whitepapers/index.html

4: Interchanging Data between .eap Files and DBMS Repositories

Necessary to setting up offsite repositories is the ability to do a full data transfer between repositories. This is supported in Enterprise Architect with the Data Transfer function which provides for interchanging data, as follows:

- EAP ⇔ EAP
- DBMS ⇔ EAP
- DBMS ⇔ DBMS

The data transfer option can be accessed from the main menu under: Tools | Data Management | Data Transfer. This will open the *Full Model Data Transfer* dialog.



A common use of this feature is in an initial set up of a DBMS repository.

5: Baseline, Difference & Merge

A Baseline allows you to create a 'snapshot' of any part of your model. You can use Baselines to compare two snapshots of a specific part of your project, to capture the differences between them and either roll back or incorporate selected changes or all changes.

A standard process where this can be used is in creating periodic baselines and comparing the current model data against a baseline, and allowing or rolling back updates.

This functionality also supports "Branching and Merging" between models. This involves copying a 'Base' model as a 'Branch'. Then, after a phase of design work, creating a baseline against the Branch model and using 'Baseline, Difference and Merge' to merge the Branch back into the Base model.

To copy a base model – see Appendix 4: Interchanging Data between Repositories

For more information on Baseline, Difference and Merging see:
http://www.sparxsystems.com/uml_tool_guide/uml_model_management/baselinesanddifferences.html

6: *Reference Data Import Export*

There are two ways of dealing with multiple projects that use common data:

a) Keep a single repository with Project Roots for each application, as well as packages for any common functionality (i.e. foundation classes)

b) Keep separate repositories for each project, but periodically update any common reference data.

The Reference Import/Export feature is an aid for the second option. Typically a master repository is kept up-to-date and a selection of the reference data is periodically propagated to other repositories.

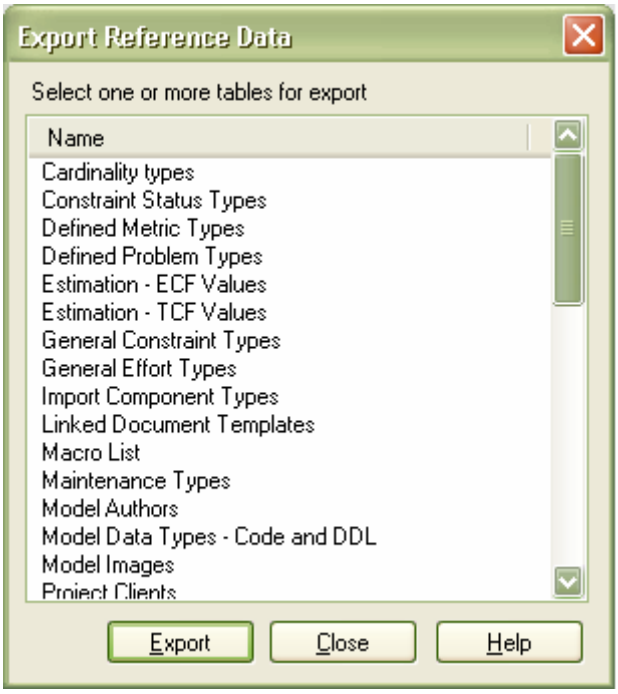
Parts of a repository that are shared include:

- Glossaries
- Type definitions (status types etc.)
- Resources, Clients, etc.
- RTF and HTML templates
- Security
- Additional stereotype profiles

When reference data is exported, Enterprise Architect writes it out to a custom XML file. This includes table information, filter information, rows and columns.

To export data, follow these steps:

1) From the *Tools* menu, select *Export Reference Data*.
This brings up the following dialog:

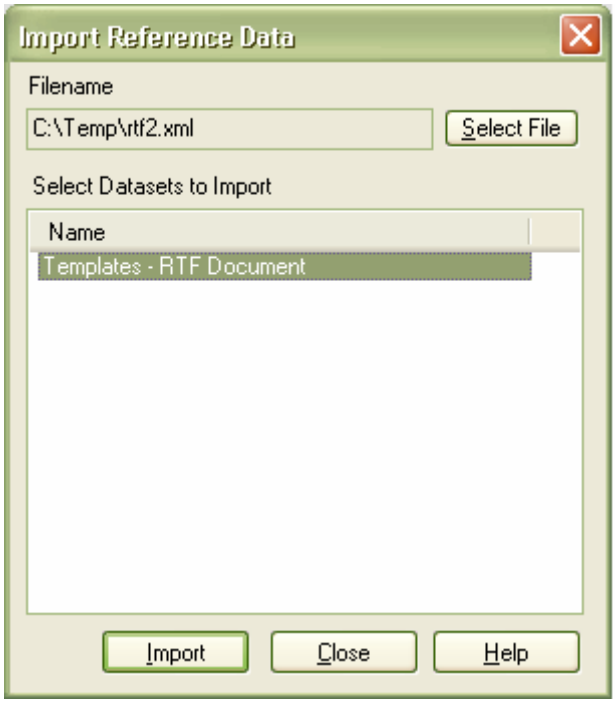


2) Select the table(s) you wish to export – you can select one or more data groups for a single file.
3) Press *Export*.
4) Enter a valid file name with an .XML extension when prompted to do so.
5) The data will be exported to the file.

To import the data into a repository:

1) From the main menu, select: *Tools | Import Reference Data*. This will bring up the dialog:

2) Use the Select File button. This opens a file explorer. Select an exported Reference-XML file.



3) Press Import to import the data.

7: Using Jet 4.0

If a Jet 4.0 database engine is required (e.g. for Unicode support), the .eap file needs to be converted to Jet 4.0 format, using MS Access, and the Jet 4.0 option needs to be enabled in Enterprise Architect. The steps to do this are as follows:

File Conversion:

a) Use MS Access 2000+ to convert the .eap file from Jet 3.5:
   a. Open the .eap file in MS Access
   b. This returns a Message "You can't make changes to the database objects in the database" – Press OK
   c. From the main menu of MS Access, select:
      Tools | Database Utilities | Convert Database | To Access 2000 Format
   d. On selection, there is a prompt to save the new file:
      i. Select the location and name the file with a 'eap' extension:
         repository.**eap**
      ii. Set: Save type as: to: "all files (*.*)"
      iii. Press OK

The new file should now be in Jet 4.0 format.

Enterprise Architect Setting:

In Enterprise Architect, the Jet 4.0 option needs to be set. To set this, from the main menu, select: Tools | Options | General | [] Use Jet 4.0. This requires a restart to enable it.

The Jet 4.0 repository is now able to be opened in Enterprise Architect.

> **Note:** There is a Jet 4.0 base model file: EABase.eap. This is available on the resources page: http://www.sparxsystems.com.au/resources/ or direct from: http://www.sparxsystems.com.au/bin/EABase_JET4.zip

8: Security

The Corporate version of Enterprise Architect provides user definable security - which allows for restrictions of user access to the model update functions. Elements may be locked on a per-user or per-group basis and a password is required to login.

Security in Enterprise Architect is not designed to prevent unauthorized access; rather it is a means of improving collaborative design and development by preventing concurrent editing, as well as limiting the possibility of inadvertent model changes by users not designated as model authors.

Enterprise Architect provides two security policies:

c) **Standard Security Model**
All elements and diagrams are considered unlocked, and a user, depending on their user-rights, may lock any element or set of elements at the user or group level as required.

d) **Rigorous Security Model**
This assumes everything is locked until explicitly checked out by a user lock. In this mode, an Enterprise Architect model is read-only until a user applies an editing lock to one or more elements.

Security allows elements to be locked for change; however users retain access to place them as linked items in diagrams where they have write permission. These elements will be shown in the diagrams, but are editable. As an example, a definition of a server is locked by the architect, but remains displayable in a diagram created by the deployment manager.

For more information on security features in Enterprise Architect, see Enterprise Architect's help under:
Model management | Team Development | User Security

9: Remote Installation of Enterprise Architect

When deploying Enterprise Architect across a network of workstations there are a number of applications and methods that can be used. Deployment can be carried out using Windows Server tools or deployment software like Microsoft SMS. The following gives a brief explanation of the method for creating an MSI file for use in this type of remote installation.

Overview
The Enterprise Architect setup executable includes an embedded windows MSI installer. On running the .exe file, the .msi file is extracted to a temporary directory. Once extracted, it can be used to carry out the remote installation of Enterprise Architect.

Some deployment software packages extract the MSI file by default. However, below outlines the manual method for doing this.

Method
To separate out the MSI for your remote installation perform the following:

1) Double click on Enterprise Architect's setup.exe (i.e. EA80_Reg.exe).
2) Go through the installation setup up to the welcome screen.
3) Using the file explorer, browse to %Temp% or C:\Documents and Settings\Administrator\Local Settings\Temp.

Note: {administrator} may be the current username – if you have a separate administrator UserName use the directory named as the UserName you have entered.

4) Find the MSI file – this is a randomly named file with a .msi extension: (it is best to sort by date and go to the latest entries).
5) Copy this MSI file to a different location.
6) You can then use this file to do the installation through Windows Server or SMS, etc.

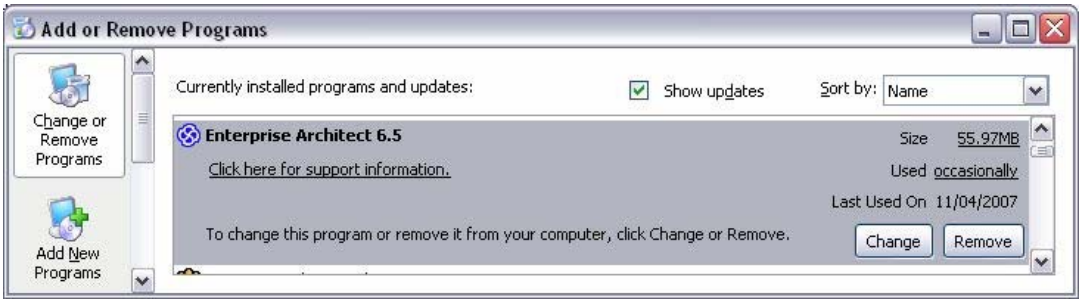
Note: By default Enterprise Architect will install for the "Current User". To install for “All Users” specify the following:

msiexec /i c:\setupfull.msi /q allusers=2

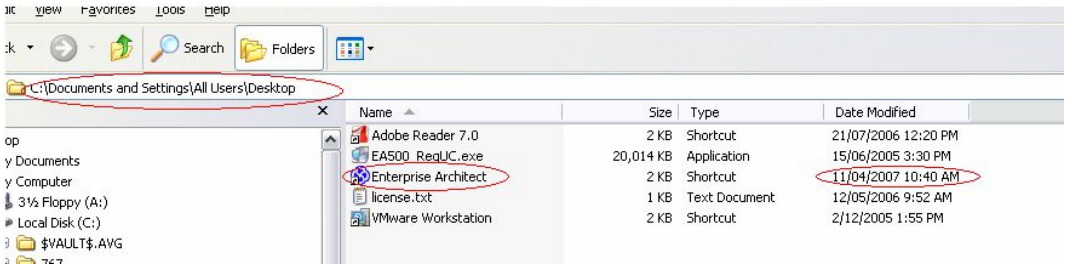
Checks Post Installation:

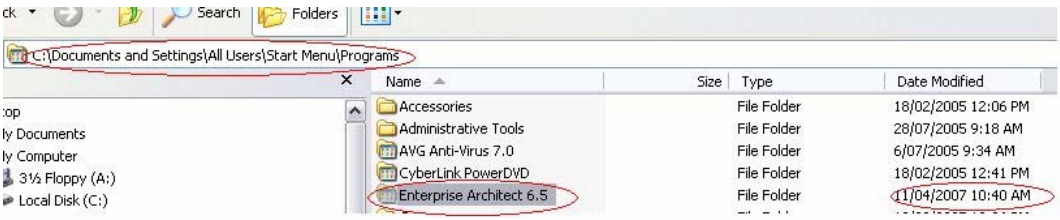
Please note that in the example below Enterprise Architect was installed for “All Users”.

1) After installation, check Windows | Add/Remove Programs for evidence that Enterprise Architect installed successfully.



2. Check the "All Users" profile for desktop and start menu items to verify that Enterprise Architect was installed successfully for all users.





Remote Floating License Installation for Enterprise Architect 8 and later

During automated installation of Enterprise Architect 8 and later, registry entries can be set for each user, giving them access to a floating license key when they start Enterprise Architect. The registry settings differ for the file based and the service based keystores as described below.

1. Example registry settings for the **file based** keystore:

```
[HKEY_CURRENT_USER\Software\Sparx Systems\EA400\EA\OPTIONS]
"SKT"=hex:00
"SharedKeyFolder"=" Y:\\Dev\\Licenses"
"AutoCheckoutEx"=hex:1a,00,00,00
```

2. Example registry settings for the **service based** keystore:

```
[HKEY_CURRENT_USER\Software\Sparx Systems\EA400\EA\OPTIONS]
"SKT"=hex:01
"SSKSAddress"="ssks://pathToKeystoreService"
"SSKSPassword"="service password (encrypted)"
"AutoCheckoutEx"=hex:1a,00,00,00
```

Key Definitions

SKT (Shared KeystoreType)	Specifies the type of keystore to obtain shared keys from. Permitted values are 0x00 (File based keystore) or 0x01 (Service based keystore).
SharedKeyFolder	This value should point to the mapped directory path, or network path, containing the shared sskeys.dat file. This setting is only used if the SKT key has a value of 0x00 (file based keystore). The above example points to a directory on a network drive: Y:\dev\licenses. **Note:** – Be aware of the double back-slashes used in the registry entry. – A full UNC path is recommended for example: "\\DevelopmentServer\EA Licenses". – Enterprise Architect users must have read and write access to this file. – If accessing a key-file on a **Novell server** the file path is case-sensitive.
SSKSAddress	The ssks address to the Shared Keystore Service endpoint. This setting is only used if the SKT key has a value of 0x01 (service based keystore).
SSKSPassword	If the shared keystore service requires a password, it may be entered into this value. Note that this value is encrypted and cannot be entered in plain text. This setting is only used if the SKT key has a value of 0x01 (service based keystore).
AutoCheckoutEx	Indicates which product keys Enterprise Architect should automatically try to obtain on start-up. Each key is represented by 4 bytes eg: 1a 00 00 00 Where bytes 1-2 are the license code (1a00) and bytes 3-4 are the license type flag (0000). The allowable values are listed below.

License codes for AutoCheckoutEx:

License	Code
Enterprise Architect Corporate	0200
Enterprise Architect Ultimate	1a00
Enterprise Architect Business & Software Engineering	1800
Enterprise Architect Systems Engineering	1900
MDG Integration for Visual Studio	0a00
MDG Integration for Eclipse	1400
MDG Integration for TcSE	2300
MDG Link for Visual Studio	0300
MDG Link for Eclipse	0800
MDG Link for Doors	0e00
MDG Technology for SysML	1000
MDG Technology for DDS	1200
MDG Technology for Zachman Framework	1600
MDG Technology for TOGAF	1d00
MDG Technology for DoDAF-MODAF	1b00
RaQuest	0c00

License types for AutoCheckoutEx:

- Full License: 0000
- Academic License: 0100

Example: If all users were to be allocated both a Corporate license and a Visual Studio Integration license, the registry key value would be:

"AutoCheckoutSharedKeyArray"=hex:02,00,00,00,0a,00,00,00

Remote Floating License Installation for Enterprise Architect 7.5 and earlier

During automated installation of Enterprise Architect, the following registry entries can be set for each user, giving them access to a floating license key when they start Enterprise Architect:

1. **SharedKeyFolder**
2. **AutoCheckoutSharedKeyArray**

The following is a registry export of the key values that need to be set:

```
[HKEY_CURRENT_USER\Software\Sparx Systems\EA400\EA\OPTIONS]
"SharedKeyFolder"="Y:\\Dev\\Licenses"
"AutoCheckoutSharedKeyArray"=hex:02
```

SharedKeyFolder

The SharedKeyFolder value should point to the mapped directory path, or network path, containing the shared sskeys.dat file.

The above example points to a directory on a network drive: Y:\dev\licenses.

Note:

- Be aware of the double back-slashes used in the registry entry.
- A full UNC path is recommended (i.e. "\\DevelopmentServer\EA Licenses").
- Enterprise Architect users must have read and write access to this file.
- If accessing a key-file on a **Novell server** the file path is case-sensitive.

AutoCheckoutSharedKeyArray

The AutoCheckoutSharedKeyArray value indicates which license types Enterprise Architect should automatically try to obtain on start-up. These values are as follows:

License	Code
Enterprise Architect Corporate	02
Enterprise Architect Ultimate	1a
Enterprise Architect Business & Software Engineering	18
Enterprise Architect Systems Engineering	19
MDG Integration for Visual Studio	0a
MDG Integration for Eclipse	14
MDG Integration for TcSE	23
MDG Link for Visual Studio	03
MDG Link for Eclipse	08
MDG Link for Doors	0e
MDG Technology for SysML	10
MDG Technology for DDS	12
MDG Technology for Zachman Framework	16
MDG Technology for TOGAF	1d
MDG Technology for DoDAF-MODAF	1b
RaQuest	0c

Example: If all users were to be allocated both a Corporate license and a Visual Studio Integration license, the registry key value would be:

```
"AutoCheckoutSharedKeyArray"=hex:02,0a
```