****

**BridgePoint and Configuration Management Using Subversion**

**Application Note**

**2011**

**Copyright** © **Mentor Graphics Corporation 2011 All rights reserved.**

This document contains information that is proprietary to Mentor Graphics Corporation. The original recipient of this document may duplicate this document in whole or in part for internal business purposes only, provided that this entire notice appears in all copies. In duplicating any part of this document, the recipient agrees to make every reasonable effort to prevent the unauthorized use and distribution of the proprietary information.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Table of Contents

[Glossary 3](#_Toc305435007)

[Notation 4](#_Toc305435008)

[BridgePoint and Configuration Management Using Subversion 5](#_Toc305435009)

[Overview of the Recommended Process 5](#_Toc305435010)

[A Configuration Management Repository 6](#_Toc305435011)

[Configuring access to the CM Repository 7](#_Toc305435012)

[Beginning a New Requirement 7](#_Toc305435013)

[Creating a Working Branch 8](#_Toc305435014)

[Sharing a New Project 10](#_Toc305435015)

[Committing Changes. 10](#_Toc305435016)

[Merging changes into TRUNK 12](#_Toc305435017)

[Key Files and Directories 13](#_Toc305435018)

# Glossary

**Requirement** A concise statement of product requirement or defect.

**Branch** Same definition as for Version below, except that the file revisions in a branch (other than the latest branch, see below) are not considered part of the mainstream of file changes. Every file in a branch (again, except the latest branch) also has a 'root' revision.

**Working Branch** A branch with a name of the form:

M<milestone#>-<Dev-INITIALS>-i<Issue#>

All work for a given issue will be done and checked into this branch.

**Latest Branch** The branch that contains the latest qualified revision of each file in the project. New projects are started from the latest branch, so it is very important that all files added to it have been fully qualified. For example, in the SVN version control tool, the latest branch is called TRUNK.

**Revision** A multiple digit number applied to a specific file. Whereas a filename and path uniquely identifies a file; a filename, path and revision uniquely identify a file's contents at a specific moment in time. Revision numbers usually take the form 'w.x'. If branched however, a file takes additional digits for example, in the SVN version control tool a file has two additional digits; 'w.x.y.z', where 'w and x' represent the simple two number revision that is the root of the branch for this file.

**Version** A named collection of files and their revision number that allows reconstitution of the state of the engineering database at a particular moment in time.

# Notation

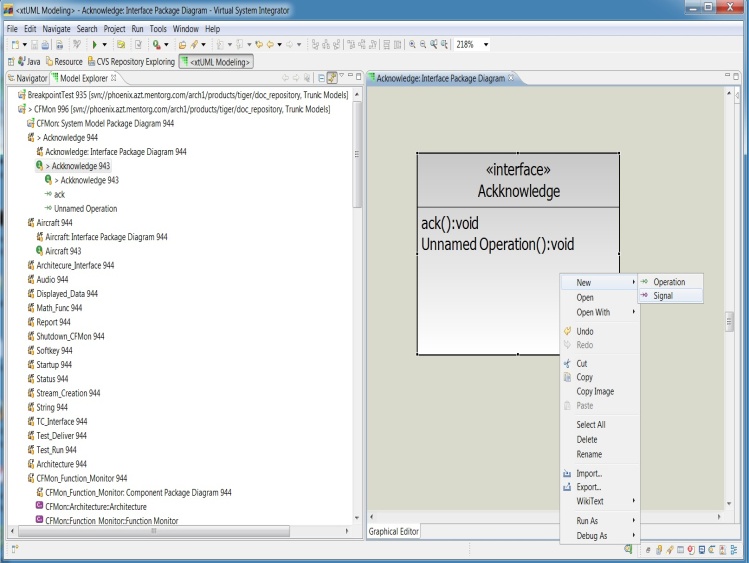
<RMB> Right-Mouse Button

**<Text-in-blue>** A sequence of menu selections

ALL CAPITALS Common Subversion term

# BridgePoint and Configuration Management Using Subversion

BridgePoint is designed to operate standalone with a single modeler or in a team environment where models are concurrently developed and shared. For small projects or individual modelers, the workspace, project and model export facilities of BridgePoint are sufficient to build an effective configuration management and archival process. However, for complex projects requiring a team of modelers, the most effective BridgePoint teams supplement the native capabilities of BridgePoint with software that provides design data management, requirement tracing and configuration management. Note, Requirements management and broader project-management topics are typically team and company specific, and therefore outside the scope of this document. However, your local BridgePoint technical representatives are trained in this area, and can offer advice on these subjects.

This application note will focus on configuration management, which is the set of practices and software that allow multiple team members to simultaneously work without interference on a common set of model elements. Although the discussion is based on the capabilities of a Subversion (SVN) repository, the process and practices described are applicable to most applications of Configuration Management (CM) software. There are a range of software solutions that provide these additional capabilities, and BridgePoint is designed to work with all solutions that support the standard eclipse framework (TEAM interface).

### Overview of the Recommended Process

Central to the Recommended Process is the idea of Requirements-driven Model development. Under this approach, a project begins based on an identified set of requirements that must be satisfied at completion. Each requirement should represent a cohesive effort, whether it’s a new capability or the repair of a defect. Often, some of these requirements will involve resolving defects or issues present in earlier versions of the project. The collection of requirements are then selected one at a time on a priority basis, and assigned to a modeler. It is at this stage, that the CM process becomes involved.

Once a requirement is assigned, the modeler will create a copy of the golden source, and place it in a work-in-progress area of the CM repository. It is common to call the golden source, the TRUNK version, and the work-in-progress copy, the BRANCH version. Modelers then introduce their changes to the BRANCH copy until the requirement is fully implemented and tested. In a team setting where multiple requirements are simultaneously being worked on, it is possible for the TRUNK version to change while other requirements are still under development. When this occurs, the working BRANCH for each of these in-progress requirements must be SYNCHRONIZED with the revised TRUNK version. This ensures that when a requirement is completed, the resulting BRANCH can be easily MERGED with the current TRUNK version.

In the sections that follow, each of the BRANCH, SYNCHRONIZE and MERGE steps will be discussed in detail.

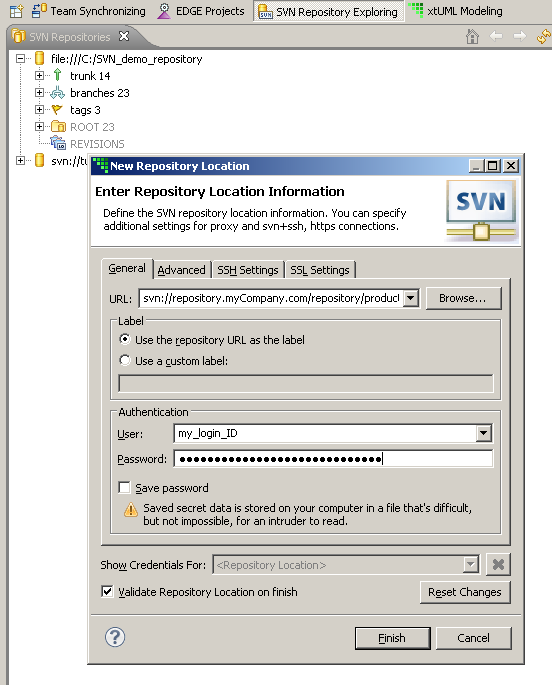
### A Configuration Management Repository

At the heart of any Configuration Management solution is the repository. The repository is a database that stores design data files, data on incremental changes to the source files, and comments documenting the designer’s intention/motivation for each change. In this article we assume that a repository server exists and has been configured to store TRUNK, BRANCH and TAG revisions. Our focus will be on the usage of TRUNK and BRANCH versions.

Note that a CM Repository can be provided either locally through a file system repository or through a remote client-server solution. In general, we find that teams employing file system repositories often create network security, multi-user access and disaster recovery issues. Consequently, we discourage the use of file system Repositories, and in this article will only focus on interactions with a client-server SVN repository.

### Configuring access to the CM Repository

When starting to work on a new requirement, it is recommended that the work be contained in a new workspace. With this new workspace, the modeler will need to configure access to the CM Repository before creating the work in progress BRANCH. To establish this connection, the following steps can be used:

* Add the SVN Repository perspective to your desktop. It is located under **Window > Open Perspectives > SVN Repository Exploring**
* Next click on the add repository location button  which will open a series of wizard windows.
* Enter the key repository information including URL of the SVN Repository server, login credentials and Structure (on the Advanced Tab)

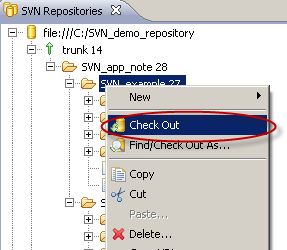
When finished, the Repository window will populate with an explorer view of the current repository contents. Some teams employ separate repositories for select information. In such a case, each repository must be added using the steps above, and will appear in the explorer view as a separate tree. The Workspace is now ready to assemble the model elements impacted by the work-in-progress.

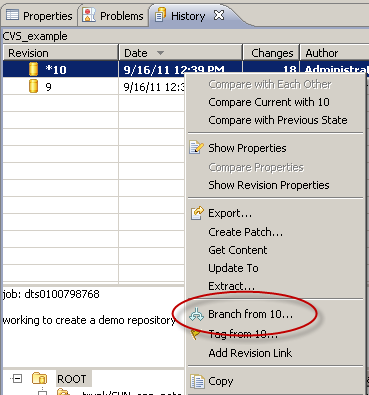
### Beginning a New Requirement

The first step in establishing the work-in-progress workspace is to create a requirement specific subfolder under the BRANCH section of the repository. The initial contents of the subfolder are a copy of the golden release contained in the TRUNK. The steps to create and populate the BRANCH subfolder are described in the section **Creating a Working Branch**. Note that many projects contain both legacy and original model content, and that each model data source uses a different series of steps to incorporate their data into the local workspace. Where legacy models are to be used, they will be CHECKED OUT from projects contained in subfolders under the BRANCH section of the repository. The steps necessary to check out these legacy models will be described in detail as part of the section titled **Branching a Project**. When it is necessary to augment legacy models with new models and projects, this new content is created in the BridgePoint workspace, and then SHARED with the repository. The steps to create and share new projects are described in the section titled **Sharing a New Project**.

### Creating a Working Branch

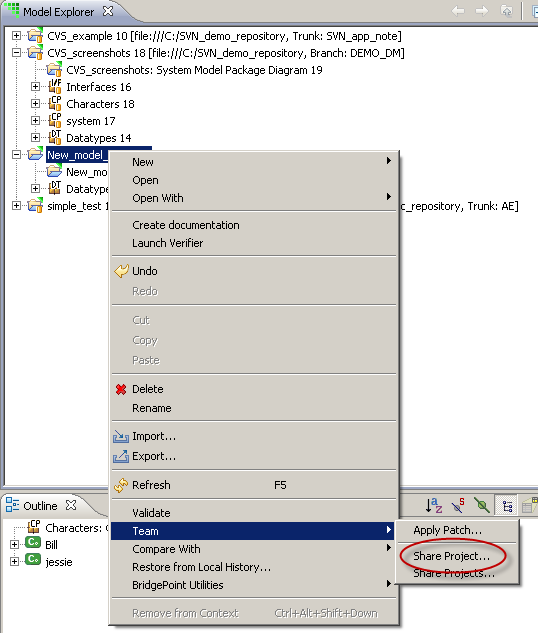
The work-in-progress BRANCH is a key mechanism to allow multiple requirements to be worked on simultaneously. A new BRANCH should be created for each requirement assigned to a developer, and the name of the BRANCH should uniquely identify the requirement and the principal Modeler. For the case where resolving a requirement requires spanning multiple releases, we recommend that the requirement be separated into multiple IDs and that each ID be unique to a release. In this way, requirement traceability can be maintained.

The steps to creating a Working branch are:

* In the SVN repository, select the golden source from the TRUNK and import it into the BridgePoint workspace using **<RMB> > Check Out**.
* Next switch to the Model Explorer view and show history of the project using **<RMB>>Team>Show History**
* (Optional) Perform acceptance testing on the project to confirm integrity.
* Create a branch in the local workspace by selecting a specific revision in the History tab (see right) and then branching using **<RMB>> Branch from** …
* Complete the Branch popup dialog by identifying the BRANCH folder name and entering a comment that documents the intent behind this change to the repository.
* These steps will be repeated for each project that is necessary for the work in progress.

In the steps above, we mention the inclusion of a comment as part of the Branch operation. Inexperienced modeling teams often overlook this important step, and fail to maintain the integrity of the comment statement. Such a failure, in our experience, leads to a process with weaker requirements traceability, lower review effectiveness, and lower model quality.

### Sharing a New Project

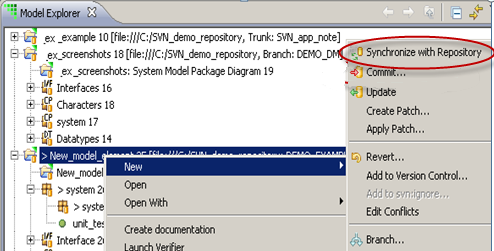
For the case where resolving a requirement requires introducing new model elements, a different procedure is required to create or extend the requirement subfolder in BRANCH. The steps for a New Project are:

* Create the new project in the local BridgePoint workspace **using File > New > xtUML Project** and then complete the entries of the New Project Wizard.
* Add the project to the BRANCH subfolder by selecting the project and **then <RMB>> Team > Share Project**
* This will bring up a window where the repository is selected, a path to the BRANCH subfolder, and a unique comment is provided.

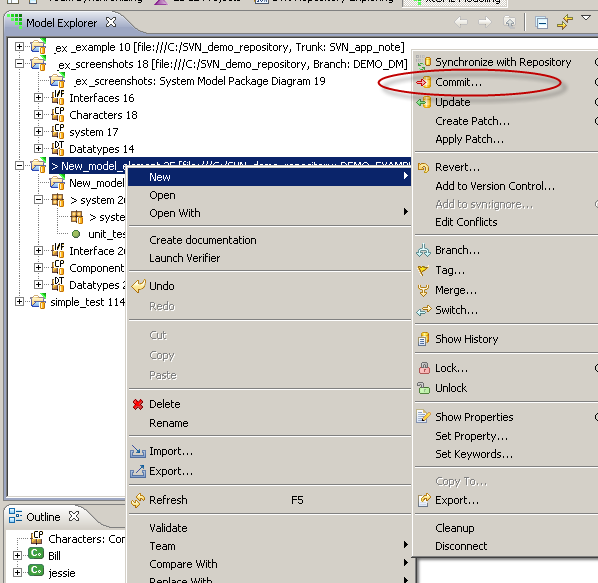
With the completion of the command, the Model Explorer view will show both the project name and the path to the SVN repository which will be used for CM control and comparison.

### Committing Changes.

Once the BRANCH folder is fully populated and the local projects under Configuration Management, the modeler can begin to enter changes that resolve the assigned requirement. As work progresses, the modeler will perform unit level testing, fixing all identified errors, and frequently COMMIT work to the BRANCH repository as part of the Disaster Prevention Plan. The frequency at which incremental changes should be committed is team specific, and often influenced by the team’s CM and Disaster Recovery practices. It is generally recommendation to commit the work when incremental change can no longer be described in three sentences or less. Also, it is a recommended practice that commits be grouped into cohesive change sets, and that priority is given to logical and functional change sets over esthetic change sets.

The steps to committing changes involve

* In the Model Explorer select the appropriate project, and then **<RMB> > Team > Synchronize with Repository**. In Team Synchronizing view a tree of update and change candidates is produced.



* If update candidates are found, then select each update analyze, and then if appropriate **<RMB>>Update**.   
  Once all updates have been performed, repeat step above.
* Review all changes that are candidates to be Committed.
* Select the root node of a change to be accepted **and <RMB>>Commit**…   
  Note that when a change is made to the project, its impact often extends across multiple files within the hierarchy. To maintain the integrity of the BridgePoint Workspace, all affected files must be committed as an atomic step.

### Merging changes into TRUNK

Eventually the modeler will commit the final set of changes that complete the requirement. At this stage, a Final Model Review should be performed, and when accepted, the BRANCH is ready to be merged into the TRUNK branch. In general, the task of merging a requirement-driven branch with the main branch requires a broad understanding of the system. While this work can be done by the original modeler, often it is better to have another team member perform this work, as this activity can then act as a further inspection of the changes as they are merged into the system. This approach can also increase cross-functional knowledge within the team.

The steps to performing a merge are very similar to the commit action described earlier. It involves using Team Synchronizing to identify differences between the BRANCH and the TRUNK version. Then select specific commits to incrementally construct a repository that contains all the changes. Note, that as with commit, it is essential that all files affected by a particular change be committed as a single group. This sequence is repeated until the BRANCH and TRUNK version no longer contain differences.

# Key Files and Directories

When using BridgePoint to do Requirements-driven development, the key components that should be placed under CM control are the markings, the model, and the eclipse support files. The generated source code does not need to be controlled, as it is always available by compiling the model. Here is a sample of the files and directories that should be placed in revision control.

* <project name>
* .externalToolBuilders/
* gen/
* models/
* .project
* .cproject

Nearly all files (\*.mark, \*.c, \*.h) under the gen/ folder should be CM controlled. There is one exception: the subdirectory code\_generation/ and its contents should not. Therefore, be sure that the code\_generation/ directory is empty before sharing, synchronizing or committing the project into a BRANCH or the TRUNK sub folder.