STOMP
Software Configuration Management Plan
Rev. 1.3

M.D. White
V.L. Freedman

June 2007

Prepared for
the U.S. Department of Energy
under Contract DE-AC06-76RLO 1830

Pacific Northwest National Laboratory
Richland, Washington 99352

This document maintained electronically at
/net/agua/files0/STOMP-QA/docs/cmp
## Revision Log

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Authors</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>October 2006</td>
<td>V. Freedman, M. White</td>
<td>Draft Software Configuration Management Plan</td>
</tr>
<tr>
<td>1.1</td>
<td>January 2007</td>
<td>V. Freedman</td>
<td>Minor updates to procedures in response to review comments</td>
</tr>
<tr>
<td>1.2</td>
<td>April 2007</td>
<td>V. Freedman</td>
<td>Updated use of revision log</td>
</tr>
<tr>
<td>1.3</td>
<td>June 2007</td>
<td>V. Freedman</td>
<td>Inserted reference to external user list documentation</td>
</tr>
</tbody>
</table>
Approvals

Approved:

__________________________________________________________________________
Mark White
Software Configuration Manager/Code Custodian

Date

Approved:

__________________________________________________________________________
Carrie Carlson
Quality Engineer

Date
# Table of Contents

1.0 INTRODUCTION ............................................................................................................................... 1

1.1 PURPOSE ........................................................................................................................................... 1
1.2 DEFINITIONS ..................................................................................................................................... 1
1.3 SOFTWARE BACKGROUND ............................................................................................................... 1
1.3.1 STOMP UTILITIES ....................................................................................................................... 2
1.4 REFERENCES ..................................................................................................................................... 2

2.0 SOFTWARE CONFIGURATION MANAGEMENT ........................................................................ 3

2.1 ORGANIZATION AND RESPONSIBILITIES .................................................................................... 3
2.1.1 SOFTWARE CONFIGURATION MANAGER (SCM) ....................................................................... 3
2.1.2 CODE CUSTODIAN (CC) ............................................................................................................... 4
2.1.3 DEVELOPERS FOR STOMP AND RELATED SOFTWARE ................................................................. 4
2.1.4 TESTERS ...................................................................................................................................... 4
2.1.5 TRAINING .................................................................................................................................... 4
2.2 ADDITIONAL CONSTRAINTS ............................................................................................................. 5

3.0 SOFTWARE CONFIGURATION MANAGEMENT ACTIVITIES ................................................. 5

3.1 SOFTWARE CONFIGURATION IDENTIFICATION ........................................................................ 5
3.1.1 IDENTIFYING CONFIGURATION ITEMS ....................................................................................... 5
3.1.2 BASELINES CHANGES ................................................................................................................... 6
3.1.3 LIFE-CYCLE MODEL ....................................................................................................................... 6
3.1.4 PROJECT FILE DIRECTORY STRUCTURE ....................................................................................... 7
3.1.5 STORAGE AND HANDLING .......................................................................................................... 8
3.2 SOFTWARE CONFIGURATION CONTROL ....................................................................................... 8
3.2.1 PLACING ITEMS UNDER SOFTWARE CONFIGURATION MANAGEMENT ................................... 8
3.2.2 PROCESSING SOFTWARE CHANGE REQUESTS .......................................................................... 9
3.2.3 DOCUMENTING RELEASES ........................................................................................................ 10
3.3 SOFTWARE CONFIGURATION STATUS ACCOUNTING .................................................................. 10
3.4 SOFTWARE CONFIGURATION MANAGEMENT ASSESSMENTS AND REVIEWS ............................ 10

4.0 SOFTWARE CONFIGURATION MANAGEMENT SCHEDULES .................................................. 11

5.0 SOFTWARE CONFIGURATION MANAGEMENT RESOURCES ................................................... 12

5.1 PERSONNEL .................................................................................................................................... 12
5.2 SOFTWARE AND HARDWARE ........................................................................................................ 12

6.0 SCMP MAINTENANCE .................................................................................................................... 12
7.0 STOMP USERS LIST ....................................................................................................................... 14

8.0 STOMP DEVELOPERS LIST .......................................................................................................... 14
List of Figures

FIGURE 1: SOFTWARE LIFE-CYCLE MODEL ........................................................................................................7
FIGURE 2: SOFTWARE CHANGE REQUEST PROCESS .................................................................ERROR! BOOKMARK NOT DEFINED.
1.0 Introduction

This document describes the software configuration management activities to be performed in support of the Subsurface Over Multiple Phases (STOMP) simulator and its associated utilities. This management plan applies to all software listed in the Software Test Plan, and is referred to as STOMP and related software in this document.

1.1 Purpose

This Software Configuration Management Plan (SCMP) provides information on the requirements and procedures necessary for the software configuration management activities of the STOMP simulator. This SCMP identifies the software, hardware, and documentation requirements for software configuration management and establishes the methodology to generate configuration identifiers, manage engineering, scientific and input/output formatting changes, maintains status accounting, and perform assessments and reviews during requirements analysis, design, development, and maintenance of the Software Configuration Items (SCIs).

1.2 Definitions

Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Code Custodian</td>
</tr>
<tr>
<td>CVS</td>
<td>Concurrent Versions System (configuration management software)</td>
</tr>
<tr>
<td>CSA</td>
<td>Configuration Status Accounting</td>
</tr>
<tr>
<td>RIDS</td>
<td>Record Inventory and Disposal System</td>
</tr>
<tr>
<td>RL</td>
<td>Revision Log</td>
</tr>
<tr>
<td>SCI</td>
<td>Software Configuration Item</td>
</tr>
<tr>
<td>SCM</td>
<td>Software Configuration Manager</td>
</tr>
<tr>
<td>SCMP</td>
<td>Software Configuration Management Plan</td>
</tr>
<tr>
<td>SCR</td>
<td>Software Change Request</td>
</tr>
<tr>
<td>SDD</td>
<td>Software Design Document</td>
</tr>
<tr>
<td>SRS</td>
<td>Software Requirements Specification</td>
</tr>
<tr>
<td>STOMP</td>
<td>Subsurface Over Multiple Phases</td>
</tr>
<tr>
<td>STP</td>
<td>Software Test Plan</td>
</tr>
</tbody>
</table>

1.3 Software Background

STOMP, an acronym for Subsurface Transport Over Multiple Phases, was developed by the Pacific Northwest National Laboratory in the framework of the VOC-Arid Soils Integration Demonstration Program, (Arid-ID) and funded by the U.S. Department of Energy (DOE), Office of Technology Development (OTD). STOMP was designed to be a general-purpose tool for simulating subsurface flow and transport. The simulator was specifically designed to provide scientists and engineers from varied disciplines with multidimensional analysis capabilities for modeling subsurface flow and transport phenomena. STOMP's target capabilities were guided by proposed or applied remediation activities at sites contaminated with volatile organic compounds and/or radioactive material. Developed with the support of the U.S. Department of Energy, Office of Environmental Restoration and Waste Management, the simulator's modeling capabilities address a variety of subsurface environments, including nonisothermal conditions, fractured media, multiple-phase systems, nonwetting fluid entrapment, soil freezing conditions, nonaqueous phase liquids, reactive transport, radioactive decay, solute transport,
dense brines, nonequilibrium dissolution, and surfactant-enhanced dissolution and mobilization of organics.

The STOMP simulator solves the partial-differential equations that describe the conservation of mass or energy quantities by employing integrated-volume finite-difference discretization to the physical domain and backward Euler discretization to the time domain. The resulting equations are nonlinear coupled algebraic equations, which are solved using Newton-Raphson iteration. The simulator has been written with a variable source code that allows the user to choose the solved governing equations (e.g., water mass, air mass, dissolved-oil mass, oil mass, salt mass, thermal energy, CO₂ gas). Depending on the chosen operational mode, the governing transport equations will be written over one to four phases (e.g., aqueous phase, gas phase, (nonaqueous phase liquid) NAPL phase, ice phase, solid phase). Solute transport, radioactive decay, and chemical reactions are solved using a direct solution technique (e.g., Patankar's power-law formulation, (total variation diminishing) TVD scheme) following the solution of the coupled flow equations. Input is directed through semi-formatted text files and output is available through a variety of user-directed formats. The simulator recognizes a number of boundary condition types and allows their specification both internally and externally to the computational domain.

The STOMP simulator is written in ANSI FORTRAN 77. A scalable version is written in FORTRAN 90. The simulator has been executed on a variety of platforms at national laboratories, government agencies, private companies, and universities. Full optimization of the simulator has been successful on Convex, Cray, Hewlett Packard, IBM, Silicon Graphics, and Sun workstations and mainframe computers. The theoretical and numerical approaches applied in the simulator have been documented in a published theory guide. Application and use of the simulator have been documented in two guide manuals. Battelle maintains the copyright to the coding and intellectual property associated with STOMP.

1.3.1 STOMP Utilities

In addition to the STOMP simulator, a STOMP preprocessor (sTeP) is used to set array sizes for STOMP compilation. Three perl scripts are used to extract STOMP output into Tecplot©, Gnuplot and Surfer formatted input files. A fourth post-processing utility is used with the scalable version of STOMP for extracting binary STOMP output into Tecplot formatted input files. All these software are maintained under CVS control in the same manner that the STOMP source code is maintained.

1.4 References


2.0 Software Configuration Management

This section describes the software configuration management organization, allocation of responsibility and authority for software configuration management activities, and references the software configuration management policies and directives pertaining to the use of the STOMP simulator and related software (i.e. STOMP utilities).

STOMP and related software used by PNNL for DOE use and its related documentation will remain under software configuration management as long as STOMP and related software is under continued development, in use for supporting Hanford Site decision making, or further use of the software is contemplated.

2.1 Organization and Responsibilities

Configuration management for the STOMP simulator and related software is designed to ensure clear lines of authority and to provide a framework within which administrative and technical control of development, enhancement and modification activities can be effectively integrated into a high-quality product for scientific research and analysis.

Primary authority for managing the STOMP software is vested in the Software Configuration Manager (SCM). Although overall responsibility for configuration management resides with the SCM, activities related to the configuration management of the software may be performed by a Code Custodian (CC).

2.1.1 Software Configuration Manager (SCM)

The general responsibilities of the Software Configuration Manager (SCM) are to control changes in the software and documentation, and to ensure testing and verification of the released versions and revisions. Emphasis is placed on providing guidance on software development, and leading other software developers.

The SCM is responsible for the following:

- maintaining management control of STOMP and related software by
  - approving or rejecting all Software Change Requests (SCRs) based on full consideration of change impacts and software needs
  - assuring that all changes made under approved SCRs are properly implemented and independently tested
  - controlling the release of SCIs.
• ensuring STOMP and related software developers are trained to the requirements, procedures, and policies of this SCMP before they can serve as change evaluators or implementers
• assisting in software configuration maintenance assessments and reviews, as required

2.1.2 Code Custodian (CC)

The general responsibilities of the Code Custodian (CC) are to maintain and track changes made to STOMP and related software. Emphasis is placed on providing a traceable history of the software and a mechanism for recreating all releases.

The CC is responsible for the following:
• ensuring that the CVS code repository is established and maintained to support software
• ensuring that SCI naming and numbering conventions, as described in Section 3.1, “Software Configuration Identification” are followed
• maintaining an SCR log to show the status of proposed, approved, and concluded changes
• maintaining the original SCRs and Revision Logs
• ensuring that the Revision Log includes, at a minimum, an inventory of the specific versions/revisions of the SCIs released with all changes identified
• configuration maintenance activities under this SCMP, and assisting developers with the use of CVS as needed
• maintaining regression tests for to be run when the baseline changes
• ensuring that backups of software, data, and records are maintained
• establishing the software configuration management system log and records
• maintaining the STOMP Users List identified in this plan

Procedures for managing change are defined in Section 3.0, “Software Configuration Management Activities.” The Software Change Request (SCR) form, Rev. 1.0, will be used to track the status and document the sequence of steps required to complete the configuration change.

2.1.3 Developers for STOMP and Related Software

Developers for STOMP and related software are responsible for ensuring that changes to SCIs managed under this plan are only undertaken in accordance with the policies, procedures, and requirements of this SCMP. Developers must receive training on this SCMP before they are allowed to perform duties as implementers of changes approved in the SCR process. Developers will implement changes for approved SCRs as assigned by the SCM. Developers may run applicable baseline tests, based on the STOMP mode they are running, but independent testing will be performed on their implementation of SCRs.

2.1.4 Testers

The general responsibilities of the Testers are to test STOMP and related software and verify that SCRs are implemented based on new requirements. Testers are responsible for creating and implementing Test Plans prior to testing. Testers are also responsible for generating Test Reports with the results of the testing and providing these documents to the CC to maintain with the STOMP records.

2.1.5 Training
Staff members charged with responsibilities under this plan will receive training to learn the requirements, procedures, and policies of this plan. The SCM is responsible for assuring that the training is completed. Completion of training will be documented by an email stating that the SCMP has been read. Copies of the email will be stored in STOMP records.

2.2 Additional Constraints

Identification of all files, documentation, media and software directories follow the procedures outlined in Section 3.1. Any additional external constraints placed on this SCMP will be identified as they arise. These include additional client requirements, additional standards, or other constraints. Client awareness of and agreement to such constraints and their impact on the project will be obtained before they are integrated into this SCMP.

3.0 Software Configuration Management Activities

This section identifies all functions and activities required to manage the software configuration of STOMP and related software products and documentation.

Software configuration management activities are grouped into four general areas:

- software configuration identification
- software configuration control
- software configuration status accounting
- software configuration management assessments and reviews.

3.1 Software Configuration Identification

Each independent source file, which may contain one or more subroutines for a particular operational mode, will be identified as a unique SCI using the scheme described in the following sections. This identification scheme will enable all project team members, including test engineers, quality advocates, and the SCM to locate each SCI quickly and easily.

3.1.1 Identifying Configuration Items

Version control and release management for all software source code will be supported using Concurrent Versions System (CVS) (Cederqvist et al., 1993) as the configuration management system. Modified source code will be checked into CVS by the CC as part of the SCR process.

A system of revision numbers will be implemented by CVS for all files, incrementing each revision number by one-tenth with each change approval by the SCM. For example, the first source file checked into CVS is assigned a revision number of 1.1. If a subroutine within the source file is revised, then the newly assigned revision number is 1.2, and is written as a header in the source file. Because revision numbers are tracked by source file, this also changes the revision number for subroutines within the same file that have been unaffected by the change. The revision number for each STOMP file accessed during execution will be printed to both the output file and screen at the end of the simulation. The new revision number will also be recorded in the SCR form during change implementation.

Because the user is responsible for creating an executable, source code is released directly to users. Complete specification of a version of STOMP requires listing the revision numbers of each source file in
addition to the date the source was released. The source release date serves as a STOMP version number, as revision numbers of individual source files may change between release dates.

STOMP source code contains a direct solver that can be used on small problems with low memory requirements. STOMP may also be executed with the SPLIB solver (Bramley and Wang 1995) and the PETSc (Balay et al. 2006). The SPLIB solver can be obtain via ftp at ftp://ftp.cs.indiana.edu/pub/bramley/splib.tar.gz, and PETSc is available at http://www.mcs.anl.gov/petsc.

3.1.1 Identifying Software Documentation

Documents will be divided into three categories: 1) principal documents, 2) addendums and errata and 3) supporting documentation. Principal documents will be identified by version numbers only. Currently, principal documents are the STOMP User’s Guide (White and Oostrom 2006), the STOMP Theory Guide (White and Oostrom 2000) and the STOMP Application Guide (Nichols et al. 1997). Addendums and errata, published or unpublished, will typically be used to document major revisions, developments or modifications. These documents will be identified with revision and version numbers. Currently, the theory guide is updated with addendums, whereas the user’s guide is updated as needed as a PNNL report.

Supporting documentation for STOMP includes Requirements and Design Documents, this SCMP, and its Project Management Plan. This documentation will be maintained in Microsoft Word with manual versioning control and maintained by the CC. As new documentation is developed, all older versions of documents will be maintained electronically, as archival material. To create a new version, the latest version will be copied; all changes accepted using the “Track Changes” features of Word, and the version number incremented in the title page and the file name. Change tracking will be used to record changes since the last version. The older version will be moved into an archive subdirectory.

3.1.2 Software Source and Documentation Storage and Preservation

Storage and preservation project software files will be the responsibility of the Code Custodian (CC). All source code will be maintained electronically on kaanapali. All documentation pertaining to STOMP and related software configuration management, user guides, software requirements, and related documents will also be maintained electronically on kaanapali. The contents of the kaanapali will be backed up by the following means:

- The directories including the CVS repository will be backed up every day to hard disk. These backups will be incremental, only backing up changes that have occurred since the last full backup. Full backups will be done on the first day of every month. A one-month retention period for all data will be maintained. All backup media used to store project files and data will be clearly labeled so that the contents may be readily identified. Labels will contain the following information: project title, software identification and date of release, date of the backup, and description of the contents backed up.

3.1.2 Baselines Changes

Each software module will be subject to configuration baselines changes as they are affected by each completed software change request implementation.

3.1.3 Life-Cycle Model
The life-cycle model is established to ensure the proper evolution of SCIs from concept to release of the software. The model shown in Figure 1 defines how each SCI progresses through the software management and development process.

![Figure 1: Software Life-Cycle Model](image)

### 3.1.4 Project File Directory Structure

A standard hierarchical directory structure will be maintained for STOMP and related software to facilitate the development and maintenance of electronic documentation and software source code. All STOMP source and documentation is stored on *kaanapali*, which is only accessible to the CC. The directory structure is shown in the following layout:

```
/Users/d3c002
 /stomp
  /code77
   /src
   /utilities
  /code90
   /src
   /utilities
 /documents
 /quality
  /scr
  /revision
 /testing
  /short_course
 /modes
```
STOMP source code written in FORTRAN 77 is stored in the /code77/src directory under the /stomp subdirectory. The scalable version of STOMP (written in FORTRAN 90) is stored in the /code90/src directory. STOMP utilities used for pre- and post-processing data are stored in the /utilities subdirectories for each version of the STOMP source code.

The /documents directory contains the STOMP user, theory and application Guides. This software documentation (e.g., manuals, bug fixes, etc.) will also be accessible through the STOMP web site at http://stomp.pnl.gov.

The /quality subdirectory stores the STOMP project management plan (PMP), requirements, design and management documents. Software Change Requests (SCRs) are archived in a subdirectory of /quality, at the same level as /revision, which archives the Revision Logs (RLs).

The /testing directory contains test cases that are divided into two categories. In the /short_course subdirectory, test problems that are used in the STOMP short course are stored and are used for testing each new build. In the /modes directory, test cases are stored that test options by operational mode and input card.

Other directories may be permitted but the directories identified in this plan are considered core directories that must be maintained.

### 3.1.5 Storage and Handling

All project SCIs will be identified to ensure that the items are properly stored for traceability, defensibility, and reproducibility. The Code Custodian will ensure that project SCIs are stored and handled appropriately. This includes storage of documents and electronic media, marking and labeling of SCIs, and retention periods. The released software and CVS archive, which can be used to recreate unique versions/revisions of the software will be maintained on the kaanapali workstation. Backups of these files onto independent media will be performed at least once a month.

### 3.2 Software Configuration Control

Software configuration management consists of establishing procedures for
- placing configuration items under software configuration management
- processing Software Change Requests (SCRs)
- creating new versions of software and documentation (RLs).

#### 3.2.1 Placing Items under Software Configuration Management

Documents are placed under configuration control upon publication release or final review of unpublished documents (e.g., letter reports, addendums, errata). Software documentation will be subject to manual version control, as described in Section 3.1.1.1 Documents are stored in electronic form under the /Release directory, and paper copies are stored in project files.

Source code modules are placed under the SCM when they become stable and have passed testing. All files related to a source code file or module, such as header and include files, are promoted along with the corresponding source code.
### 3.2.2 Processing Software Change Requests

All proposed changes that impact a SCI are recorded on one or more SCRs. All SCRs related to a revision of STOMP and related software will be tracked in a Revision Log (RL). The status and disposition of the software change request will be noted on the SCR.

An SCR is prepared to request, authorize, and track implementation of an SCI for changes approved by the SCM. Final testing and verification is performed when a user requests a release, and the outcome is recorded in the Revision Log. The process flow for managing change using SCRs and RLs is illustrated in Error! Reference source not found..

The SCR form is available electronically at [http://stomp.pnl.gov](http://stomp.pnl.gov) as file “STOMP-SCR-form.doc”. Anyone may request a STOMP and related software change, or report a STOMP and related software problem, using this form. The person requesting the change must complete the “Change Request” block of the SCR form and submit it electronically to the Code Custodian (CC).

Upon receipt of a SCR, the CC will designate it with an SCR number, and alert the SCM of the request. If the SCR is a request for a bug fix, the SCM is responsible for the SCR approval. If the SCR is a development task, the SCM may evaluate the request independently, or may opt to alert members of the STOMP Change Control Board (CCB) by email that an SCR needs to be evaluated for approval. This decision process is at the SCM’s discretion. The SCM or the members of the STOMP Change Control Board will assess the following:

- Change Evaluation: is the proposed change reasonable and practical?
- Software Design Requirements: is the proposed change consistent with the Software Requirements Specification (SRS), as outlined in Requirements Document?
- Project Management: is the proposed change in line with a project needs?
- Criticality: is the proposed change critical to completing client deliverables?
- Time Required: how much time will the proposed change require for implementation?
- Configuration Control: will this change create a necessity to constitute a new release?
- Quality Control: How will this change impact the existing code and how much regression testing will be required?

The CC will archive under the responses from either the SCM, or the STOMP CCB, and complete the “Change and Evaluation” block of the SCR form, noting if the change has been approved for implementation or not. If the change is approved, the SCM will assign it to one or more software development team members for implementation. If the change is rejected, the CC will log the SCR as closed and complete the “notification” section in the “Change and Evaluation” block of the SCR form.

The software developer(s) will make the software changes proposed in the SCR and perform unit testing as needed on the changes, and notify the CC that the code is ready to be checked in. The CC will then check the updated source code into the CVS repository tagged with a comment containing the SCR number, and a brief description of changes made to source file.

Tests identified in the STOMP software test plan are performed once a new release is requested by a user. Testing will only be performed on the operational modes that have been updated since the previous release. The SCM will verify the new source by running the new source through an archived test suite that has been developed for that particular application. For a new functionality, new test cases will be developed and verified by independent testing. If the SCM was responsible for any of the changes to the software, then a STOMP Tester will verify the testing performed by the SCM. When verification is
complete, the CC will complete the Revision Log, and the CC will release the source to the STOMP user. The date of the source release will be recorded by the CC in the RL.

When new source is released, the CC will complete the SCR/RL process by notifying users of the final status of the SCR, which may include:

- SCR originator (person who requested one of the changes implemented in the new release)
- All STOMP development team members
- All STOMP users

The date and method of notification will be recorded in the “Change Notification and Closure” block of the RL.

### 3.2.3 Documenting Releases

The information needed to reconstruct any release (that is, the state of STOMP tools on a given date) will be available through use of the CVS configuration management software. The CVS tools enable the CC to extract source code from the repository by date, which can then be used to build any STOMP executable in effect on the specified date. The full specification of a release will include the CVS revision number for each source file and the date of the entire source release.

### 3.3 Software Configuration Status Accounting

Configuration Status Accounting (CSA) is a means by which enhancements/changes and new versions/revisions of configuration items are identified and tracked. A CSA system is already established with the SCR process for the software and documentation. The status of proposed changes will then be progressively tracked through approval and implementation. These records provide traceability between versions of SCIs and associated documentation. Whenever possible, this tracking system will be automated. The CC will report the status of each SCI to the SCM, STOMP development team, and STOMP users periodically or upon request.

### 3.4 Software Configuration Management Assessments and Reviews

Software configuration management assessments and reviews will be conducted, at a minimum annually, to determine to what extent the actual software and documentation reflect the required physical and functional characteristics. These activities are formal examinations of the as-built versus the as-required software and documentation. The SCM will participate in all assessments and reviews and record the resulting action items. As each baseline is defined, a quality engineer will also assist with in-process assessments of software configuration management activities.
4.0 Software Configuration Management Schedules

Software configuration management activities will span the entire life of the software. Software development will consist of the following phases:

- Software Requirements Specification
- Design the Capability
- Assemble the Capability
The closeout phase of the software will include the archiving the source code files and documentation. During retirement, support for STOMP and related software is terminated. STOMP users and developers will be notified electronically (e.g., email, website) to terminate all STOMP and related software usage.

5.0 Software Configuration Management Resources

The SCM will ensure that an effective software configuration management program is established, supported, implemented, and maintained. The SCM or a designated authority is responsible for establishing procedures, plans, and training for the implementation of configuration management.

5.1 Personnel

The SCM is responsible for implementing this plan. He will establish and maintain the development support directories, and assist the CC in processing SCM documentation, generating status accounting reports, and preparing and distributing the project deliverables. The SCM is also responsible for ensuring that all testing activities are evaluated, documented, and reported according to the requirement of the SCMP. In addition, the SCM will participate in and provide documentation for system development assessments and reviews, if requested.

The development team members are responsible for the generation of software, electronic documentation, and other SCIs applicable to the project under the direction of the SCM.

5.2 Software and Hardware

The SCM and development team members will utilize CVS to perform identification, control, and status accounting to system SCIs. STOMP development and testing will be carried out on Linux-based platforms using the Portland Group FORTRAN compiler. Currently, primary STOMP development is carried out on a Mac OS X, 10.4.8 platform. Independent testing is primarily carried out on PCs using a RedHat Workstation OS (Linux 2.6.9-42.0.3.ELsmp #1 SMP Mon Sep 25 17:28:02 EDT 2006 i686 i686 i386 GNU/Linux). If a new compiler is implemented, and/or a new operating system is introduced, then the complete set of tests will be run to establish that STOMP functionality has not been adversely impacted, and establish a new baseline if needed.

Project records are stored and maintained as specified in the project Records Inventory and Disposition System (RIDS).

6.0 SCMP Maintenance

SCMP maintenance is necessary to document configuration management activities throughout the software’s life cycle. If any procedures defined in this document are changed, those changes will be reflected in the SCMP, as needed.

It is the SCM’s responsibility to ensure the compliance and cooperation of development team members in abiding by this plan. The Software Configuration Manager’s responsibility is to monitor compliance, and ensure that changes and updates are reflected in the SCMP, as required.
Reviews of this SCMP will occur periodically throughout the STOMP simulator’s life. At a minimum, reviews will occur at the start of each major development phase. At that time, proposed changes, if any, will be evaluated by the SCM and may be approved by the SCM in conjunction with the CCB for implementation. All changes to this plan will be communicated to the development team in a timely manner.
7.0 STOMP Users List

The following table identifies individuals at PNNL designated as STOMP users.

<table>
<thead>
<tr>
<th>Users</th>
<th>Office</th>
<th>Telephone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bacon, Diana</td>
<td>ETB/2126</td>
<td>372 – 6132</td>
</tr>
<tr>
<td>Fang, Yilin</td>
<td>ETB/2128</td>
<td>372 – 4051</td>
</tr>
<tr>
<td>Freedman, Vicky</td>
<td>ETB/2130</td>
<td>372-4067</td>
</tr>
<tr>
<td>Hammond, Glenn</td>
<td>ETB/2124</td>
<td>375-3875</td>
</tr>
<tr>
<td>Keller, Jason</td>
<td>PORTLAND/18</td>
<td>503-417-7565</td>
</tr>
<tr>
<td>Nichols, Will</td>
<td>ETB/1323</td>
<td>372-6040</td>
</tr>
<tr>
<td>Oostrom, Mart</td>
<td>ETB/2123</td>
<td>372 – 6044</td>
</tr>
<tr>
<td>Rockhold, Mark</td>
<td>ETB/2131</td>
<td>375 – 2516</td>
</tr>
<tr>
<td>Waichler, Scott</td>
<td>ETB/1322</td>
<td>372-4423</td>
</tr>
<tr>
<td>Ward, Andy</td>
<td>ETB/2133</td>
<td>372-4417</td>
</tr>
<tr>
<td>White, Mark</td>
<td>ETB/2129</td>
<td>372-6070</td>
</tr>
<tr>
<td>Williams, Mark</td>
<td>Sigma5/2121</td>
<td>376-5787</td>
</tr>
<tr>
<td>Wurstner, Signe</td>
<td>ETB/1321</td>
<td>372-6115</td>
</tr>
<tr>
<td>Yabusaki, Steve</td>
<td>ETB/2123</td>
<td>372-6095</td>
</tr>
<tr>
<td>Zhang, Fred</td>
<td>ETB/2132</td>
<td>372-4866</td>
</tr>
</tbody>
</table>


An external user list is maintained in the file stomp_license.doc and is archived in the same directory as this CMP.

8.0 STOMP Developers List

The following table identifies all individuals designated as STOMP developers.

<table>
<thead>
<tr>
<th>STOMP Developer</th>
<th>Office</th>
<th>Telephone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bacon, Diana</td>
<td>ETB/2126</td>
<td>372 – 6132</td>
</tr>
<tr>
<td>Fang, Yilin</td>
<td>ETB/2128</td>
<td>372 – 4051</td>
</tr>
<tr>
<td>Oostrom, Mart</td>
<td>ETB/2123</td>
<td>372 – 6044</td>
</tr>
<tr>
<td>Rockhold, Mark</td>
<td>ETB/2131</td>
<td>375 – 2516</td>
</tr>
<tr>
<td>White, Mark</td>
<td>ETB/2129</td>
<td>372 – 6070</td>
</tr>
<tr>
<td>Williams, Mark</td>
<td>Sigma5/2121</td>
<td>376-5787</td>
</tr>
</tbody>
</table>

Last Updated January 18, 2007.