

Software Configuration Management Plan

Highland Basic Order Tracking System

Prepared For: Highland Office Supply
Prepared By: John Steele
Elucidata

Document ID: 100-01-02
Version: 01

Copyright ©2001 Elucidata

TABLE OF CONTENTS

INTRODUCTION	4
PURPOSE & SCOPE	4
METHODOLOGY	5
REFERENCES	7
CONFIGURATION ITEM CLASSIFICATION	8
EVOLVING ITEMS	8
SOURCE ITEMS	8
SUPPORT ITEMS	9
ARCHIVE ITEMS	9
HARDWARE	9
CONFIGURATION IDENTIFICATION	10
EVOLVING ITEM IDENTIFICATION	10
SOURCE ITEM IDENTIFICATION	13
SUPPORT ITEM IDENTIFICATION	13
ARCHIVE ITEM IDENTIFICATION	13
CONFIGURATION PROCESSES	15
CONFIGURATION REVIEW BOARD.....	15
CLASSIFICATION	17
IDENTIFICATION	17
STORAGE	17
REVISION.....	18
RETRIEVAL.....	19

INTRODUCTION

This document represents the Software Configuration Management Plan for the Highland Basic Order Tracking System software development project. Software Configuration Management (SCM) is essentially the process of identifying and assuring retention of all of the various artifacts (documents, source code, executables, e-mail, etc.) generated during the Software Development Life Cycle (SDLC).

PURPOSE & SCOPE

The primary objective of the SCM process is to coordinate the use of software artifacts among the project participants, making sure everyone is working with the same versions of various artifacts (change control), and making sure that nothing gets lost (retention control).

This SCM process is tailored to fit the current software development effort, and is related to the planning and life cycle description documents for this project. This project is classified as a small to medium database development effort; large software development efforts require more rigorous controls than are described here.

This plan provides the criteria and direction necessary for the performance of the Configuration Management (CM) process during the SDLC, and pertains to the products either produced as a result of the development effort, or procured as support or reference items.

METHODOLOGY

The methodology presented here is based on the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) and the Institute for Electrical and Electronics Engineers (IEEE) standards for Information Management. This SCM plan:

- Makes use of the principal project participants as defined in the Vision & Scope chapter of the project planning document for this software development project.
- Describes the various categories of project artifacts, and how each category is to be controlled.
- Describes how the various project artifacts will be identified, and how that identification will change as each artifact evolves.
- Describes processes for approving revisions, assuring the availability of the most recent version of each artifact to all project participants, and making previous versions available upon request.

PERSONNEL ROLES AND RESPONSIBILITIES

In a small database development effort, two principal roles are defined for SCM activities:

1. Primary End-user Representative (PER)
2. Primary Developer Representative (PDR)

The PER acts as the primary point of contact for the end-user community. The PER is responsible for ensuring that significant project-related documents and communications generated by the end-user community are relayed to the PDR for storage and future reference.

The PDR acts as the primary point of contact for the development team. The PDR is responsible for the identification, control, and distribution of project configuration items (described below).

CONFIGURATION ITEMS

Project artifacts that are uniquely identified and placed under version control are known as Configuration Items in the configuration management world. In this project, configuration items fall into four general classes:

1. Evolving items, such as documents, which are subject to one or more revisions and new releases during the SDLC.
2. Source items, generally source code and object files used to build a production software application, which are generally numerous and frequently changing.

3. Support items, such as operating systems, of which the project requires certain versions for successful operation.
4. Archive items, such as SQA review forms, generally support decisions made during the SDLC, and are stored in electronic format for future reference.

Refer to the Configuration Item Classification chapter for further descriptions of configuration items.

CONFIGURATION IDENTIFICATION

The methods for uniquely identifying instances of each of the configuration item classes described above are specific to each class:

- Project artifacts classified as evolving configuration items are generally SDLC stage deliverables. These items are assigned unique identifiers that identify the project and stage in which they were originally created, along with their current revision level.
- Source items, although similar in nature to evolving items, are generally restricted to a specific identification convention by the development tools used for the project. These items are managed by object name and date via a configuration management tool that is part of, or compatible with the development platform.
- Support items are procured at specific points in the SDLC, and carry their own version numbers assigned by their developers. The software developed for this project relies on specific minimum versions of each support item for successful production operations.
- Archive items are identified by format and date at a minimum. Additional search capabilities are generally available, but specific to each format.

Refer to the Configuration Identification chapter for a further description of configuration identification.

CONFIGURATION ACTIVITIES

The processes used to manage configuration items fall into five main categories:

1. Classification: Each project artifact is classified as a member of one of the four configuration item classes.
2. Identification: Evolving configuration items are assigned appropriate identifiers.
3. Storage: Each configuration item is stored in electronic form, in a manner appropriate to its class.
4. Revision: Each configuration item is revised in a manner appropriate to its class; archive items are never revised.

5. Retrieval: The current versions of all evolving items are made available to project participants via the project home page. Other items are available via retrieval mechanisms specific to their class.

Refer to the Configuration Activities chapter for a detailed description of each configuration activity.

REFERENCES

The software development lifecycle for this project defines a series of stages; each project stage is defined as a separate operation with specific inputs and outputs. This SCM plan describes how these inputs and outputs will be controlled and managed. The complete software development lifecycle for this project is described in a separate document, available at:

<http://www.elucidata.com/refs/sdlc.pdf>

Please refer to the SDLC document for a description of the structure, inputs to and outputs from each of the stages of the SDLC, as well as a comparison to other SDLC models.

Other terms common to the software development process are defined in a Glossary of Software Engineering Terms, available at:

<http://www.elucidata.com/refs/seglossary.pdf>

Please refer to this glossary for definitions of the terms used in this document and in the SDLC document.

STANDARDS

The following standards were used as guides to develop this SCM process. The standards were reviewed and tailored to fit the specific requirements of small database projects using the referenced SDLC.

- ANSI/IEEE 828-1990: Standard for Software Configuration Management Plans
- EIA 649: National Consensus Standard for Configuration Management
- SEI CMM: Configuration Management Key Process Area

CONFIGURATION ITEM CLASSIFICATION

In the configuration management world, project artifacts such as documents, applications, and support/reference files generated or purchased during the SDLC are all referred to as configuration items. As each item is generated or purchased, it must first be classified into one of four configuration item classes: Evolving, Source, Support or Archive. The class to which an item is assigned heavily influences its subsequent treatment during configuration management.

EVOLVING ITEMS

Evolving items generally take the form of documents, help files, test data and application executables generated during the development process. These items are generally revised one or more times during the SDLC, and as such are considered to be evolving.

Evolving items are generally project stage deliverables, and are generated as a result of collaboration between participants. Although evolving items are in the substantial minority of all items generated or purchased during the SDLC, they require the most control and coordination, and are the subject of the majority of the processes described in this plan.

SOURCE ITEMS

Source items are generally source code and object files used to build a prototype or production software application. Although source items also have an evolutionary nature, they are normally used only by developers through various development tools. The storage of source items depends heavily on the selected development tool; some are stored as objects in a single large file or database table, others may be stored as independent files.

Due to the sheer quantity of these items, as well as the frequency of their revision and the possibility of multiple, simultaneous editors accessing them, source items are generally managed by specific configuration management tools. As such, it is best to utilize the identification schema and configuration management processes supported by the selected development tools. This plan defers configuration

management of source items to a configuration management tool that is part of, or compatible with the chosen development platform for this project.

SUPPORT ITEMS

Applications, operating systems, development tools and other items purchased to support the SDLC and/or production operations are termed support items. The key term here is purchased. Almost every other configuration item is created at some point during the SDLC. Support items are created by other vendors and purchased as necessary. Examples include development tools, database/application/web servers, documentation tools and operating systems. In some cases, software support items are compatible with a specific range of hardware. If this is the case, any hardware items that are bound in this manner to software support items are managed by the PDR as hardware support items.

The reason support items are tracked under this configuration management plan is that the development and/or production operations will most likely depend on their presence or availability. Furthermore, these operations may well be dependent on specific versions of the support items. For example, the application may require a compiler of a specific version or higher, and be incompatible with lower versions. Under configuration management, the compiler would be identified as a support item with a minimum acceptable version level.

ARCHIVE ITEMS

Items that are created during the SDLC, but are not evolving or source items are termed archive items. These items are retained for future reference, not revision. Examples include SQA review forms, project status presentations, copies of electronic mail correspondence, and memoranda of stage exit approval and product acceptance.

HARDWARE

Although supporting hardware may be procured for development, test and/or production operations, this plan defers responsibility for most hardware items to the respective property managers of client and developer.

Since minimum acceptable hardware configurations may be defined by the support items described above, some hardware items are managed by the PDR as support items when this relationship is identified. In this case, the PDR and the property manager coordinate with each other and with the Configuration Review Board before authorizing the upgrade or replacement of hardware upon which software support items are dependent.

CONFIGURATION IDENTIFICATION

The methods for uniquely identifying instances of each of the configuration item classes described above are specific to each class. Evolving items are assigned unique identifiers, while source item identification is managed by the selected configuration management tool. Support items carry their own identification and version numbers assigned by their developers, while archive items are identified primarily by format and date.

EVOLVING ITEM IDENTIFICATION

Evolving items fall into two classes: Documents and software executables / support files.

DOCUMENTS

Document evolving items are assigned unique identifiers that identify the project and stage in which they were originally created, along with the current revision level. The identifier consists of three parts separated by hyphens in the format 123-45-67.

The first part of the identifier is the project number; the second part identifies the stage in which the item was originally created; the third part is simply an ascending number for items generated during the stage.

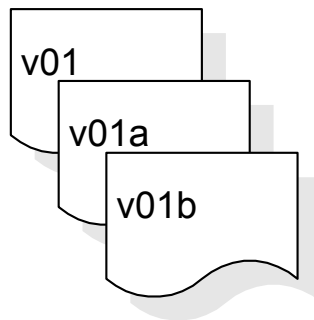
[Project] [Stage] [Sequential
Number]
217 – 02 – 02

Stage numbers are assigned as follows:

#	<i>Configuration Items</i>
00	Reference items, such as the SDLC and glossary
01	Planning stage items
02	Requirements stage items
03	Design stage items
04	Development stage items
05	Test & Acceptance stage items
06	Implementation stage items

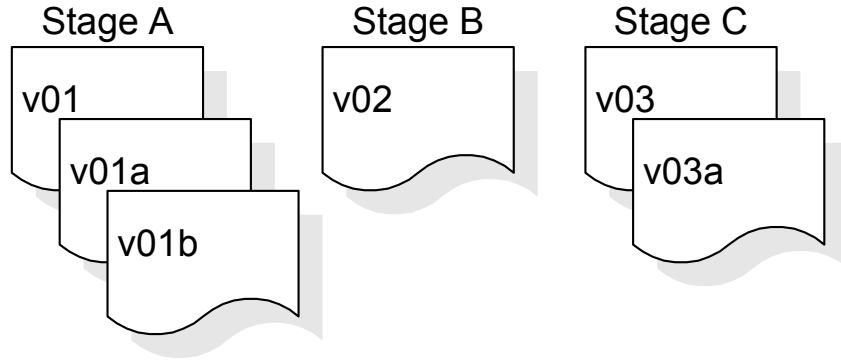
For example, the requirements traceability matrix is first created during the requirements stage, and is generally created second, after the requirements document. For project 217 the requirements traceability matrix would carry an identifier of 217-02-02. This is known as the item's primary identifier.

The version level of the item is maintained as a separate identifier. This allows the primary identifier to be used as part of a file name or URL for access to the most current version without requiring changes to all referencing items. The version level is maintained as a numeric identifier followed by an optional character string (a, b, c...), which may be appended to the version number of an item and incremented as the item is updated during the review cycle.



When an evolving item is updated within a stage, the appended character string is incremented. For example, version 01 is incremented to version 01a followed by 01b and so on within the stage. This is done to allow project participants to make sure they are working with the latest item, or to reference a previous version of the item.

When an evolving item is updated during a subsequent stage, the numeric version identifier is incremented. Version 01b would become 02, followed by 02a when revised within the new stage.

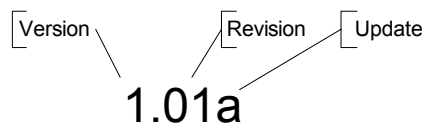


For example, the requirements traceability matrix, originally created during the requirements stage, would have been revised once during the design stage and again during the development stage. As a result, its version number would be 03. If an error was found at this point and the matrix updated during the stage, the new version number would be 03a. The complete identification for the requirements traceability matrix at this point would be 217-02-02, version 03a, which is generally shortened to 217-02-02 v03a.

This process applies to application code as well, when the code is finally released from the development cycle into production status. When the code and baseline documents are finalized and the code is approved for production the entire version ID #, complete with alpha character update identifier, becomes the final release identifier for the production code or document.

SOFTWARE EXECUTABLES AND SUPPORT FILES

Software executables and support files are generally identified by name and version number, such as “Main DB v1.01a.” The naming convention for each software evolving item is defined by the development team. The version numbering scheme consists of three components:



1. Version number, which appears to the left of the decimal,
2. Revision number, which appears to the right of the decimal, and
3. Update level, consisting of a single trailing character.

VERSION NUMBER

The Version number changes only when the core architecture of the software item changes, as when moving from one level of the development tool to another, when an application is completely overhauled, or the user interface changes fundamentally. In this case, version 1.01a would become version 2.0.

REVISION NUMBER

The Revision number changes when new features, functionality, or other content are added or significantly changed. In this case, the core architecture or user interface have not been changed, only extended or limited in some manner. The most common reason for changing the revision number is when adding a new module or other functionality to the software item. The normal sequence of revision is 1.0, 1.01, 1.02 and so on.

UPDATE CHARACTER

The Update character is appended or incremented when the only change to the software item is to correct one or more bugs, without the addition of any new functionality. Version 1.01 of the software would become v1.01a, 1.01b, and so on. This updating is over-ridden when a combination revision, involving bug fixes and new feature additions, is performed. In such a case, the software revision number is incremented and any update indicator is dropped, as in v1.01b to v1.02.

SOURCE ITEM IDENTIFICATION

As described in the previous chapter, source item identification is managed by the selected configuration management tool for this project: Visual SourceSafe. Source items are vendor supplied software; refer to the software documentation for a description of source item identification.

SUPPORT ITEM IDENTIFICATION

Support items are identified by common product name and the version number range necessary to support successful development or production operations. For example, a text editor may be upgraded during the course of the project from version 2.1 to version 2.2a; the version range for this configuration item would show 2.1 - 2.2a. This is important for after-the-fact retrieval of archived project information; documentation and source items are best handled with known compatible versions of their parent applications.

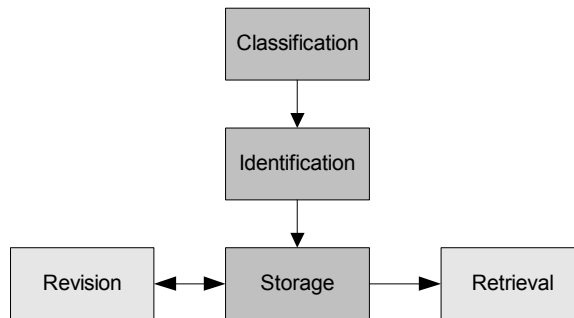
ARCHIVE ITEM IDENTIFICATION

Archive items are generally miscellaneous support documents and records of communication that are stored for later reference. These items are generally stored by document class, such as e-mail, presentation, review form or spreadsheet; the subdirectory each document is stored in provides an intrinsic subclassification for the item. Each item is identified by file name and modification date as returned by the operating system. In the event an identically named item is already present in

a target subdirectory, the new document file name will be appended with a sequential number to prevent naming conflicts.

CONFIGURATION PROCESSES

The processes used to manage configuration items include Classification, Identification, Storage, Revision and Retrieval. The first three processes are sequential and mandatory for each configuration item. The character of the last two processes changes in accordance with the classification of the configuration items being handled. The PDR is responsible for the execution of all configuration management tasks described in this plan.



CONFIGURATION REVIEW BOARD

During the Formal Iteration process for each Stage, the PER and PDR meet to evaluate the current set of issues raised during reviews. In the configuration management world, the PDR and PER are acting as the project Configuration Review Board (CRB). The project Executive Sponsor is invited, but not required to participate. The CRB may ask other Subject Matter Experts to attend, depending on the issues being discussed.

The CRB is responsible for prioritizing and selecting issues to be resolved during the SDLC. The PDR maintains a central list of issues, recording new issues as project participants raise them. The CRB then evaluates unresolved issues, assigning them to Open status if it is not appropriate to resolve them during the current SDLC stage. Unresolved issues that are selected for resolution are assigned to a project participant who then acts as the party responsible for resolving the issue. The PDR logs the responsible party and any issue status changes in the Issues Log. When examining the Issues Log, any unresolved issues without responsible parties assigned have not yet been addressed by the CRB.

The CRB is also responsible for determining the timing of support item hardware and software upgrades. Since these upgrades often impact production software viability, the CRB controls the revision cycle for these items.

ISSUES LOG

Any issues raised during the Formal Iteration process are recorded in the project Issues Log. The decision process is documented in the Issues Log along with all other CRB decisions. The CRB cannot take a formal issue under consideration until it is present in the Issues Log.

The issues log contains the following information:

Issue #	Issue numbers are automatically assigned when the Status cell is filled in.
Status	An issue may be either "Unresolved," "Resolved," or "Open."
Item	Source document or abbreviation with version number.
Title	Short title to make it easier to scan down the list.
Description	Description of the problem.
Responsibility	Name of person responsible for hunting down the answer. Most often the PER or PDR.
Resolution	Description of the solution.
Initiation Stage	Stage in which the issue was raised.
Resolution Stage	Stage in which the issue was resolved.

The current status of an issue is always in one of three states: Unresolved, Resolved, or Open. A new issue that has not yet been addressed by the CRB can be identified by the lack of a Responsible Party. Once an issue has been considered by the CRB, a Responsible Party is assigned to resolve the issue.

Open issues are issues that have been considered by the CRB, and deferred until a later stage in the SDLC. This is most often done when an issue is minor, or it is inappropriate for the issue to be addressed during the current stage.

The PDR maintains the Issues Log on the project homepage, so that all participants can examine the log to see if someone else has already recorded an issue they have discovered.

The Initiation Stage and Resolution Stage are metrics cells, used to track the pattern of error occurrence and resolution. Notice that the Initiation Stage mentions all the normal SDLC stages plus a Production stage. This allows issues to be logged against a production app and tracked to see in which version the issue was eventually resolved.

CLASSIFICATION

The PDR is responsible for classifying each configuration item as it becomes available to the project. Each item is classified as a member of one of the four configuration item classes described previously.

IDENTIFICATION

Evolving configuration items are assigned appropriate identifiers as described in the previous chapter. Source items are introduced to the source item configuration management tool, which handles their identification. Support item identifiers are recorded in the project support items log. Archive items are intrinsically identified; no further action is taken here.

STORAGE

Each configuration item class has different storage requirements. The evolving items have the most complex storage requirements, while archive items are the simplest to manage.

EVOLVING ITEMS

After identification, evolving items are stored in the subdirectory specific to the stage in which they are produced. A copy of the item is then placed on the project Web Server, and a link established from the project homepage to the new item. If the new item is an update or revision of an existing item, the old item is replaced on the Web Server by the new item under the same hyperlink. This insures that the most recent version of each project deliverable is available to all participants.

The current version, and all prior versions of each evolving item are stored in the project Configuration Management Tool: Visual SourceSafe. Standard check-in/check-out controls prevent more than one person from editing an evolving item.

SOURCE ITEMS

Source items are generally created by members of the development team and immediately introduced to the source item configuration management tool: Visual SourceSafe. The tool handles storage, identification, versioning and check-in / check-out operations. Any member of the development team, as well as the PDR, may introduce source items to the source item configuration management tool. This is the only case where the PDR does not have to be involved in the identification or storage of a specific new configuration item.

SUPPORT ITEMS

The PDR logs each support item as it is received from a vendor into the support items log for the project. After the software has been appropriately installed, the original media used to perform the installation, along with any necessary license key information, is placed into a physically secure storage area for later retrieval if necessary. In most cases, a software copy of the installation executable is stored on a central fileserver.

Support items may only be upgraded on production systems after the concurrence of the Configuration Review Board. This rule is in place to make sure the CRB is involved in the decision to upgrade software or that may impact the viability of current production software.

ARCHIVE ITEMS

Archive items are stored in the appropriate subdirectory of the archive items directory for the project. The archive items directory is used primarily by the PDR for analysis and generation of project metrics, but is available on a central fileserver. Only the PDR has write access to this directory; other project participants may make read-only copies when necessary.

REVISION

Generally only evolving items are subject to revision under this plan. Older versions of evolving items are never destroyed; they are still stored in the project folder, but are generally not available to project participants via the project homepage. This allows the project to roll back to a previous version if necessary.

Support (software and hardware) items may be upgraded at the discretion of the CRB, which is responsible for determining the impact the upgrade may have on production systems.

SDLC REVISIONS

Evolving items are commonly revised as the project moves through the SDLC. The formal process for review and release of an item is described in the Assessments and Reviews chapter of the Software Quality Assurance plan for this project.

Issues raised during the Formal Iteration process of each SDLC stage are handled by the CRB as described previously.

MAINTENANCE REVISIONS

After a software item has been placed into production, bugs and enhancement requests (often referred to as *issues*) are collected by the PDR and PER. The CRB is responsible for storing these issues and providing further analysis when necessary.

CONFIGURATION REVIEW BOARD

The purpose of the CRB in the context of production software maintenance is to prioritize and select issues to be addressed during the next software revision cycle. Once a set of issues has been selected for development, a new SDLC is initiated, beginning with the revision of the project plan and continuing forward through the entire SDLC, revising stage deliverables as necessary. In essence, a maintenance release is a mini-project, accelerated by the re-use of the stage deliverables from the original project.

RETRIEVAL

The current versions of all evolving items are made available to project participants via the project homepage. Source items are retrieved via the source items configuration management tool; access control mechanisms may be in place in the project has been partitioned among different developers. Support item installation media may be physically retrieved from secure storage by the PDR if necessary; in most cases, support items are available electronically. Read-only copies of all archive items are available to project participants on a central fileserver.