Configuration Management

CxOne Standard

CxStand_ConfigurationManagement.doc

November 4, 2002

Construx
SOFTWARE

Advancing the Art and
Science of Commercial
Software Engineering

CONTENTS

Copyright Notice

1 Introduction

This standard defines CxOne's taxonomy for the configuration management CKA, along with standardized descriptions of configuration management processes.

1.1 Overview

CxOne defines configuration management (CM) as revision control, change management, and release management for software projects. Although configuration management activities may seem somewhat disparate, there is a common thread linking them: identifying the state of project deliverables at a given point in time and ensuring project deliverables are delivered according to plan (including scope and quality goals).

This is accomplished by identifying and tracking all elements, groups of elements, and versions of those elements (revision control); ensuring that artifacts and deliverables match the scope of planned and actual work (change management); and providing releases of deliverables that have holistic integrity and support deployment goals (release management).

1.2 Goals

CxOne support for software configuration management focuses on the following goals:

- Defining a common terminology.
- Distillation of common principles into readily available checklist items.
- Provide a resource for information about existing best practices, methodologies, and techniques.
- Increase productivity by addressing issues in a uniform fashion, reducing common errors, duplication of effort, and multiple solutions for the same problem.

1.3 Background

CxOne configuration management materials are synthesized from many sources including the SWEBOK, IEEE software standards, Steve McConnell's works, other industry sources and literature, and Construx's experience with software projects, consulting, and training.

1.4 Relationship to other CKAs

Configuration management activities permeate most software engineering activities. CxOne focuses on revision control, change management, and release management for the configuration management CKA because they are core CM activities that stand on their own. CxOne has chosen to define other CM related activities in other CKAs to increase usability.

The planning and management of CM activities is defined as part of the management CKA, creating builds is part of the construction CKA, and tracing is dealt with where it is most useful (requirements, design, and quality).

1.5 Relationship to External Standards

1.5.1 SWEBOK

The CxOne configuration management area has made the following adaptations of the SWEBOK organization:

1. *Management of the SCM Process* is dealt with in the management CKA.
2. *Software Configuration Identification*, *Control*, and *Status Accounting* have been re-structured into revision control and change control.
3. *Software Configuration Auditing* is part of the quality CKA.
4. *Release Management and Delivery* is part of release management, except for software building which is dealt with as part of the construction CKA.

These changes were made to support a more pragmatic, usable taxonomy for materials.

1.5.2 IEEE Standards

IEEE 610 defines configuration management as:

> "A discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with requirements."

CxOne material explicitly and implicitly supports IEEE 610, but as noted in the SWEBOK section some CM issues are supported in other CKAs for usability reasons.

2 Configuration Management

This section covers global Configuration Management (CM) issues. The top level sections following this one provide an overview of the revision control, change management, and release management.

2.1 How to Allocate CM Activities

Since CM activities are spread across project activities, CM is a team sport. How CM activities are split up will vary depending on the needs of a project. How CM goals are accomplished is not as important is that they get done with the appropriate level of detail.

2.2 System CM vs. Project CM

Configuration management sometimes mistakenly focuses on functional system artifacts (e.g., code). In CxOne configuration management is applied to all project artifacts. As a basic example, every document and e-mail on the project should be under some form of revision control in addition to code.

2.3 How to Manage CM

Management of CM activities are usually done through a number of mechanisms, including selecting processes and assigning staff responsibilities.

Examples of how CM activities can be managed include:

- Ensuring the project planning has an appropriate level of CM structure for the project
- Explicitly delegating CM activities to appropriate leads
- Assigning a CM Lead for a project with complex CM needs
- Having one or more Change Control Boards (CCBs) and building project management workflow around the CCBs.
- Defining a versioning scheme that fits the project's lifecycle and schedule
- Creating intuitive and usable organizational structures for project information
- Using traceability to verify consistency between different types of artifacts (e.g., requirements and test cases)
- Internal and external audits

3 Revision Control

Revision control consists of the identification, storage, and management of projects artifacts and the revisions over time of those artifacts. Projects use revision control to avoid inadvertent changes, determine the state of the system or project at any point in time, enable project builds, provide support for change control, prevent tampering, etc.

Almost all aspects of revision control are supportable by modern software engineering revision control tools; it does not make sense to attempt software development without this support. CxOne requires a robust revision control tool infrastructure in place.

3.1 Identification

Projects should identify the items under revision control on the project. Identification can be done in a variety of methods and to different levels of detail. Final selections of these are left to the project team.

Examples of configuration items include:

- Product artifacts such as executable code, source code, user documentation, etc.
- Project artifacts such as project plans, work breakdown structures, specifications, test plans test cases
- Acquired elements such as software libraries, COTS packages, and test tools

Examples of identification methods include:

- Revision control tools
- Standard naming and/or numbering conventions
- Self describing organizational structures (e.g., well named folder hierarchies)
- Document maps

3.2 Storage

Projects should identify where project artifacts reside and what tools are used to control their revision history. When appropriate, the project should discuss on how to remove, add, revise, or recover elements from the library.

3.3 Versioning

Projects will normally have several levels of versioning for artifacts and groups of artifacts. Revision control software greatly facilitates this process, providing revision history for individual items, and the ability to label, branch, or otherwise identify at a unique point in time specific artifacts and versions of artifacts.

The versioning of artifacts is often tightly coupled with building the system and release management of the system.

4 Change Management

Change management consists of the systematic identification, analysis, tracking, and control of changes to project artifacts. Projects use change management to decrease feature creep, increase visibility into changes, and understand changes to project scope.

In CxOne, change management has two major components, change control and Corrective Activity Management (CAM). Change control activities deal with identifying, approving, and executing changes to artifacts. The change management portion of CAM deals with managing the details associated with proposed changes, ensuring project planning reflects changes, and managing the execution of approved changes.

This section deals primarily with change control. See ***CxPattern_CamDatabase*** and ***CxPattern_ChangeManagementDatabase*** for a description of change management in CAM.

4.1 Change Control vs. Revision Control

Revision control focuses on the use of tools that allow changes to a rapidly evolving artifact to be sequentially captured and retraced, if necessary. This allows an artifact to undergo rapid development while retaining the safety of backup copies and some measure of control. Change control focuses on procedures by which changes to an accepted artifact are carefully proposed, assessed, conditionally accepted, and applied. Change control provides a measure of stability and safety beyond that of revision control tools.

4.2 Artifact Lifecycle

Change control oversees the revision of artifacts that have achieved a certain level of completeness and are designated for change control by project plans. Change control ensures that changes during a software project occur in a defined, visible, and controlled fashion.

The artifact CM lifecycle from initial concept to final release is shown below in Figure 1-1.

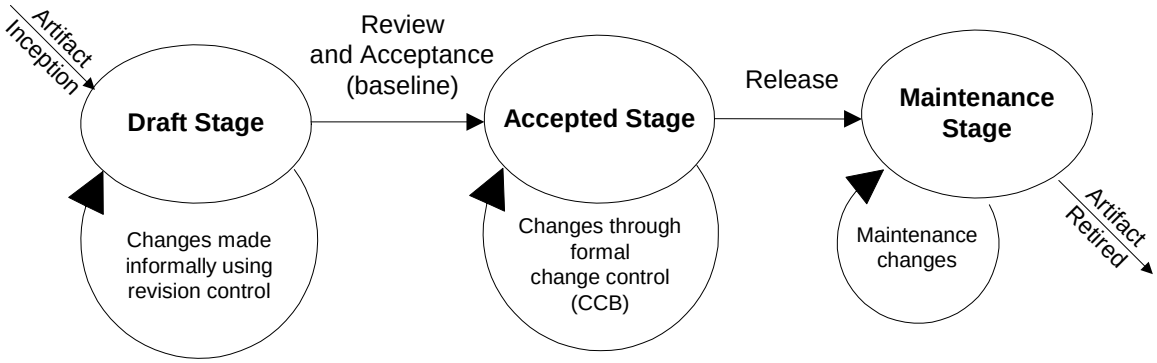

Figure 1-1: Artifact CM Lifecycle

The artifact (which may be any project artifact including a document, source code, binary component, art work, etc.) begins life in a draft stage. As the artifact develops, changes are

made informally and work progresses using revision control. When the artifact reaches an expected state of completeness, it undergoes review and acceptance. Once accepted, the artifact is considered ***baselined*** and enters the formal change control stage where all changes need to be approved by the Change Control Board (CCB).

Not all artifacts are explicitly placed under change control. Many artifacts like source code are implicitly placed under change control because the upstream artifacts they need to remain consistent with are under change control. CxOne defines ***explicit change control*** as direct control by the CCB and ***implicit change control*** as control through upstream artifacts**.**

Once released the artifact enters a maintenance stage. How the maintenance stage is handled is very dependent on the project, organization, and artifact in question. Eventually an artifact is retired, which normally means it is no longer in use (or at least no longer supported).

4.2.1 Draft Stage

During the draft stage the artifact is undergoing frequent and rapid change before it becomes stable. At this stage, it is inappropriate to apply change control to the artifact since the overhead of controlling changes merely obstructs efficient creation of the artifact. However, this stage of development may comprise a significant body of work that needs some level of identification, storage robustness and coordinate beyond sitting on someone's local storage, which is what revision control provides. Revision control provides automated support for saving and restoring versions of project artifacts such as documents and computer source code without the burden of a formal change process.

An informal determination should be made by the engineer(s) working on the artifact as to when to place the artifact under revision control. Normally the revision control overhead should be low enough to have artifacts be placed in revision control very close to their inception. Since formal change control is not occurring at this stage, it is the responsibility of each engineer to use prudent judgment and professional practice to store revisions of the artifact at appropriate intervals.

4.2.2 Accepted Stage

Once formative development is complete, the artifact will go through formal acceptance. The determination of this point in the development process will be defined in the configuration management plan. An artifact accepted into change control is often referred to as baselined. The artifact is now called a "revision" instead of a "draft." Subsequent changes to the artifact are under the control of the Change Control Board (CCB).

Acceptance includes review of the artifact to determine if it is ready for change control. The CCB will determine if anyone besides the engineer(s) is necessary to determine that the artifact is ready for change control.

Artifacts that are in the Accepted stage still utilize revision control tools and procedures.

4.3 Explicit vs. Implicit Change Control

Due to the overhead involved, it doesn't make sense to place all artifacts under control of the CCB. It often makes sense to place only up-stream, high leverage, and critical documents under the CCB, letting the CCB indirectly control changes to the downstream artifacts.

For instance, on most projects the CCB is unlikely to make direct decisions about whether changes will be made to specific source code files. However, changes to those source code files will be indirectly controlled by CCB actions on other project artifacts like the requirements or design.

The CM plan determines which artifacts will be placed directly under change control. These artifacts are said to be under ***explicit change control***. The CM plan may also call out artifacts that are being indirectly controlled by the CCB, or those artifacts may be inferred from the explicit change control artifacts. Either way, such artifacts are referred to as being under ***implicit change control***. Implicit change control usually captures most artifacts on a project, but if there are artifacts that are intended to not be under explicit or implicit change control they should be called out in the CM plan.

4.4 Change Control Board

The change control board (CCB) is responsible for processing change requests. The CCB will make the final determination of changes to artifacts under change control. The CCB will solicit input from affected stakeholders and then make prioritization decisions based on that input. The CCB will involve all appropriate stakeholders when making decisions.

Larger or more complex projects may need multiple change control boards for different areas, or a hierarchy of change control boards to deal with varying detail and impact of issues.

In all cases, the structure and use of the CCB should be driven by balancing an appropriate level of control with a minimization of overhead.

4.4.1 Defects vs. Change Requests

Defects are not normally processed by the CCB, but on some projects this may be appropriate. There can often be a blurry line between some defects and change requests, so projects should be aware of this and plan appropriately. *CxPattern_DefectManagementDatabase* describes an appeal mechanism where a defect can be appealed to the CCB.

4.4.2 Preparing Change Requests

Change requests should have as much information as possible gathered before they are presented to the CCB. CxOne recommends utilizing a database to record, store, and track change requests.

The submitter of each change request should incorporate some type of cost/benefit analysis, normally as much as the submitter is capable of providing. The level or rigor of this analysis will depend on the nature of the project and the size of the request. The estimated impact to

the project in terms of schedule, cost, risk, or other factor should be estimated, along with the estimated benefit of making the change.

The CCB will use this information as a starting point and ensure that all stakeholders are informed of the change and agree with the cost/benefit analysis.

4.4.3 Assessing Impact of Changes

Once a change request has been submitted to the CCB, the change request is circulated to those stakeholders that the CCB identifies as being impacted by the change. These stakeholders are responsible for producing or verifying estimates of the impact and benefits of implementing the proposed change, and for estimating the impact or effect of not completing the change.

The CCB may reject a proposed change out-of-hand if it determines that the cost of assessing the impact outweighs its perceived benefit. In the interest of efficiency, the CCB may decide to queue a series of change requests to be processed as a group. This will depend upon the frequency and importance of change requests as determined by the CCB.

Proposed changes should conform to *CxCheck_ChangeRequest*.

4.4.4 Accepting or Rejecting a Change Request

Once the impact of the proposed change to the entire project is assessed, the CCB must make a decision whether to accept or reject the change request. The CCB needs to reach a unanimous decision on a change request to move forward. This decision will be reached by balancing the costs and benefits of a change request.

If accepted, the change request must be prioritized in relation to other development work making the necessary trade-offs between time, function, and effort. CCB will determine where a change request's priority fits in relation to work already underway or scheduled.

4.4.5 Recording a Change Request

Regardless of whether a change is approved or rejected, the following information is recorded in the change request database and made available to the party submitting the change proposal (and any other interested parties that desire to monitor the progress of the artifact):

- The date, description, and party submitting the change request.
- The estimated impact of the change on schedule, cost, and/or risk
- The date when the change was accepted or rejected.
- If accepted, the overall impact on the project schedule, cost, and risk (which includes the effects of any mitigating strategies and their descriptions).
- If rejected, the reason for rejection.

The CCB will provide notification of meetings and their agenda and disseminate the results of its actions. Interested parties may elect to attend the CCB meeting in order to represent their interests. The CCB retains the right to evaluate and make a determination on all change requests in private before opening them up for public debate.

5 Release Management

Release management consists of the identification, packaging, and delivery of the elements of the product to an external or internal customer.

Identification determines "what" is released the customer, e.g., user documentation, executables, release notes, etc. Packaging determines the form in which it is released, e.g., zip file vs. install program. Delivery determines "how" it is released to the customer, e.g., shrink-wrap vs. web download, etc.

5.1 Types of Releases

Projects may have many different types of releases to different audiences. Releases are normally tied to project milestones. They may be consumed by internal audiences (e.g., quality control releases) or by external ones (e.g., product release at the end of a project).

5.2 What makes a Release

A release is when a project makes an explicit effort to identify, marshal, and deliver a set of artifacts. What constitutes a release is defined by the unique needs of each project.

Releases always have:

- Deliverables – These are the intended deliverables defined for the release. They could range from initial project documentation to gold master images for a product.

Releases also usually include some form of:

- Versioning – Releases normally have one or more versions that represent a particular configuration of release deliverables.
- Release Documentation – Information about the release delivered apart from the deliverables. Often used to capture deviations between the planned and actual release.
- Deployment Support – If necessary or appropriate, setup or deployment software or instructions, hardware and software requirements, resource requirements, etc.
- Release Map – Formal releases use this to describe the physical contents of the release and any other pertinent CM information related to the release.

5.3 Versioning

Versioning allows the deliverables that make up a release to be managed and for differences in system deliverable configuration over time to be communicated.

Versioning at the revision control level allows for proper identification of artifacts to be released. There may also be versioning to control different versions of sub-systems that are put together to make a final release version. There is usually a global release version which defines a snapshot of the state of all artifacts that make up a release. There may be customer

release versions that communicate interpretability to customers. There may be marketing versions that communicate product information to the marketplace.

Many software products also require internal operational versioning that allows the software to correctly interact with different versions of itself in the field, or allow for identification of deployed states as part of customer support or maintenance. This sort of product versioning needs to be identified as part of release management, but may also be tightly coupled to the behavior of the product itself (e.g., communication protocol versioning, where the software needs to make functional decisions based on the version) or to a critical support function like product marketing (e.g., how does the version of this product fit with our product families).