Delaware Child Support System (DECSS)

DELAWARE HEALTH AND SOCIAL SERVICES

Division of Child Support Services

Configuration Management Plan

Table of Contents

Figures and Tables

1. Introduction

1.1 Purpose

Configuration Management (CM) is the discipline of identifying, recording, evaluating, tracking, coordinating, reporting, and controlling configuration items by performing supporting process activities that maintain the integrity of these items throughout the lifecycle of a project.

The purpose of this Configuration Management Plan is to establish and maintain the integrity of the project assets and work products, and to define the CM processes and activities that will establish and maintain administrative control of those assets and work products. The plan will describe the CM processes and activities, including the planning of CM, configuration identification, configuration control, CM reporting, and baseline verification.

1.2 Scope

Configuration Management is a discipline that improves the overall quality of the systems, documentation, design, code, files, and data delivered on a given program. It applies technical and administrative direction and surveillance throughout the project life cycle to identify and control changes to software and documentation. CM does this through a defined approach to accomplishing configuration identification, storage, configuration change control with versioning, configuration status accounting, and configuration verification. Successful implementation of CM requires participation from the entire Project team.

The plan addresses all the tools and techniques required to maintain version control and tracking effectively. The plan does not cover hardware control. The plan also does not cover release management in detail. The plan will be updated as there are changes to the process.

1.3 Approach

The methodology outlines the process used to control project assets and work products, while supporting a change control process that is required when there is a change to a configuration item. While implementing CM tools and processes, the DECSS Team will follow the best practices and policies to avoid common configuration problems and maximize team productivity:

1. Identify and store artifacts in a secure repository. The repository is a potential central point of failure for all assets; therefore, it must be fault-tolerant and reliable. Microsoft Team Foundation Server (TFS) has the ability to revert or adjust from mismatches revealed through configuration reviews.

2. Control and review changes to artifacts - must be able to control who is allowed to modify them, as well as keep a record of what the modifications were, who made them, when they were made, and why they were made.

3. Organize versioned artifacts into versioned components - CM components are a set of related files and directories versioned, shared, built, and base lined as a single unit.

4. Organize versioned components and subsystems into versioned subsystems - To allow management of highly complex software systems and must be able to step beyond the management of individual components and group those components into subsystems.

5. Create baselines at project milestones - At key milestones in a project, all the artifacts should be base lined together. In other words, record the versions of all the artifacts and components that make up a system or subsystem at specific times in the project. At a minimum, artifacts should be base lined at each major project milestone. There are three main reasons to baseline: reproducibility, traceability, and reporting.

6. Record and track requests for change - Change request management involve tracking requests for changes to a software system. These requests can result from defects found by a testing organization, defects reported by customers, enhancement requests from the field or customers, or new ideas produced internally.

7. Organize and integrate consistent sets of versions using activities - An activity represents a unit of work performed. The grouping of file and directory versions is a change set or change package.

8. Support concurrent changes to artifacts and components - One of the key things a CM tool must support is the capability to modify the same files concurrently and to integrate or merge the changes made in parallel at the appropriate time.

9. Integrate early and often - Plan integration points early in the development life cycle and continue integrating often over the course of the project.

10. Perform Software builds and ensure correctness - Establish proper procedures and provide sufficient tracking to record the following: as to who performs the build and what went into each executable or library build.

The TFS Version Control Tool will have the provision and ability to manage all the artifacts produced for developing the software including the system artifacts and project Management artifacts effectively.

TFS Team Explorer will be used to manage and track the defects and requests for enhancements as work items.

1.4 Roles and Responsibilities

Successful implementation of the CM requires participation from the entire project organization. Below table summarizes the roles and responsibilities of members involved in CM activities.

Table 1: Roles and Responsibilities

Role	Role Description	Teams
Change Control Board (CCB)	• Responsible for change request review and approval, workitem scope and effort estimation. Approval of workitem needed to move to development phase.	DCSS, IRM, Maintenance Vendor (MV)
Vendor Project Manager	• Monitors and Oversees progress • Provide periodic reports to Management • Conducting Informal Audits to verify that Configuration Items are being correctly managed and CM procedures are working as defined • Provide ad hoc reports using CM tools to Management as needed. • Configuration Status Accounting	MV

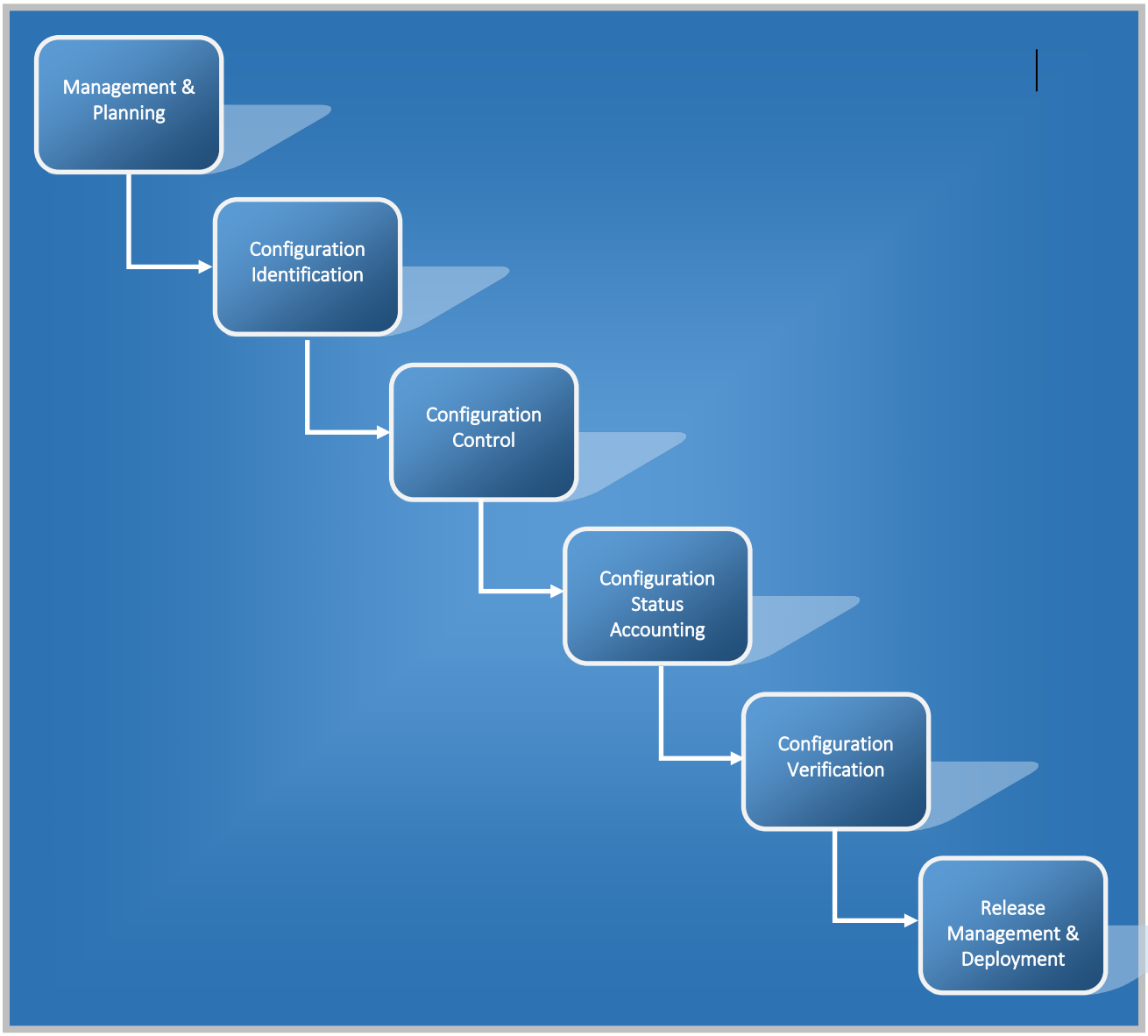
Table 1: Roles and Responsibilities		
Role	**Role Description**	**Teams**
Configuration Manager	• Executes the role of Release Manager • Overall ownership of CM tools and work related to them • Conducts the baseline verifications, as required • Deploying builds to proper environments • Follows the CM process • Makes changes to work products, as assigned • Reviews CM reports • Conducts configuration verification and reviews • Determine what goes into a release • Review management plans, as required	IRM, MV
Project Managers and Development Managers	• Coordinating with the Configuration Manager and functional team leads to determine priorities and schedule new releases • Review CM Reports • Ensure that corresponding team members follow the process and procedures • Ensure requests are sent to the CM team to promote the code to the next environment • Responsible for the overall quality of service delivery. • Participate in reviews	MV, IRM
Technical Team	• Follow the project CM plan • Make changes to work products, as assigned, according to documented requirements • Checkout and check-in the code. • Update the status in the defect tracking tool • Complete and test changes to configuration items, schedule walkthroughs and reviews of changes made, and coordinate the implementation of those changes to Unit Integration and System Test • Coordinate any changes to production batch schedule and submit Batch Change Request (BCR) as required	MV, IRM
Business Analysts	• Update or review documentation associated with formal system changes • Follow the organization and project configuration • Create test plans and execute various forms of testing in the system test environment to verify that the system has been changed as discussed and documented • Verify that end-to-end testing is successful with no impacts to current system processing upon a major change in system • Make changes to work products, as assigned, according to documented CR requirements • Participate in reviews	MV

Table 1: Roles and Responsibilities		
Role	**Role Description**	**Teams**
System Integration Testing (Testers)	• Adhere to CM process for testing and documentation • Research and verify changes based on Change Requests	MV
User Acceptance Testing (UAT) Testers	• Adhere to CM process for testing and documentation • Research and verify changes based on Change Requests	DCSS
Technical Change Advisory Board	• Serves as CM tool administrator • Administer and perform TFS Warehouse and TFS Test environment refresh • Ensure ongoing administration and support of CM tools (e.g. configuring security, adding users, ensuring backups)	IRM

2. Process

Figure 1 shows the process that is used for Configuration Management. The first step is management and planning. Next configuration identification is used along with configuration control, configuration status accounting, and configuration review to maintain the integrity of work products. Finally the work product is released and deployed.

Figure 1: Configuration Management Process



2.1 Management and Planning

The Configuration Management Plan describes the CM processes and activities that include the planning and managing of configurations, configuration identification, configuration control, CM reporting, baseline verification and release management.

The CM planning process begins with identifying and documenting the goal of the CM initiative and related organizational tools and procedures. The goal is to maintain the integrity of the project assets and work products using defined CM processes and activities that will establish and maintain administrative control.

2.2 Configuration Identification

This scope of Configuration Identification is to ***identify*** and ***name*** configuration items that will be placed under CM. Another part of configuration identification is the *creation* of a CM system for ***storing and accessing*** the configuration items.

Configuration identification describes the selection, creation, storage and specification of the following configuration items:

- Software Artifacts like source code, Database Scripts, etc.
- Deliverable Documents like DSD's, Test Cases, etc.
- Other supporting items that are used in creating and describing above work products like Project Schedule and Architecture Diagrams

A *baseline* is a set of specifications or work products that has been formally reviewed, and once agreed on, serve as the basis for further development; they can only be changed through change management control procedures. A baseline has an identifier established that represents a configuration item or a collection of configuration items and associated entities. As project evolves, several baselines may be used to control its ongoing development and testing. Currently DECSS has established baselines for Documentation, Development Source Code, UIT source code, SIT Source Code, UAT Source Code, Production Source Code. The Baseline can only be changed using the change control process with the approval of Change Control Board (CCB).

Below individual sections describe configuration identification key terms respective to DECSS:

2.2.1 Identifying Configuration Items

DECSS Project's Configuration Items include Software Artifacts, Design Documents, and other work products created for building the application or any other documents or tools that support the DECSS Project. The following are Configurable Items that are identified for Configuration Management. As the project identifies additional categories and/or items, they will be added to TFS:

Table 2: Configuration Items

Software Artifacts	• Source code (HTML, C#, ASPX, Java scripts, database stored procedures) • Metadata (data stored for creating/modifying business components, SQL Server Metadata) • Reference data (seed data, parameter tables, error message table) • Database DDL scripts (create table, alter table, create or alter index, etc.) • Batch Control Software (automated job scheduling software) • Interface code (code or scripts used for interfacing with the external systems such as MQ, or web services that interact with external agencies)

Table 2: Configuration Items	
	• Configuration files (Initialization or startup files that contain configuration parameters on web server, application server and database server) • Test scripts (scripts that are created for testing, validation and reporting) • Environment scripts (Any scripts related to Operating system or any files associated with server specific information) • Conversion scripts (scripts/programs that are created for converting the data to SQL Server from the legacy system)
Deliverable Documents	• Functional design documents • Technical design documents • Application Training documents • Management Plans
Supporting Work Products	• Requirements document • JAD documents • Logical and Physical data models • Architecture diagrams • Application and database standard documents • Project schedule • Work Items

2.2.2 Naming Configuration Items

Standard naming conventions are defined for configuration items as part of the CM process. These naming conventions specify how different versions of each are to be uniquely identified.

The naming conventions for all application-based source codes are specified in the Coding Standards section of the Application Development Plan and the Database Development Plan. These CIs when stored in the version control repository receive a number (version number) for maintenance and any record of accomplishment changes. The Version Control Tool automatically applies version numbers to CIs. The following table list all the naming conventions used in DECSS.

Table 3: Work Product Naming Standard			
Work Product	**Function**	**Naming Convention**	**Example**
Management Plans	Work products generated during the planning phase:	<Management Discipline> Management Plan	Configuration Management Plan
Execution Plans	Work products generated during the	<Project Phase> Plan	Technical Design Plan

Table 3: Work Product Naming Standard			
Work Product	**Function**	**Naming Convention**	**Example**
	planning phase:		
Joint Application Development (JAD)	Work products generated during the requirements analysis and baseline process:	<Sub System abbreviation> - <Module name> JAD Document	For the Disbursement module, a JAD document within Financial Sub System: FM – Disbursement JAD Document
Use Cases (UC)	Work products generated during design process	UC-<< Sub System abbreviation>>-<<function within subsystem>>-<<a three digit number>> - <<The description of the use case>>	UC-FIN-COL-010 Receive Batch and Process Collections
System Interface Documents	Work products generated during design process	DECSS <Interface name abbreviation> System Interface Document	DECSS DOL System Interface Document
Design Specification Documents (DSD)	Work products generated during design and baseline process	DACSES DSD - <Four Letter abbreviation for Screen or Report or the Batch process name>	CP Debit card and Direct deposit Status (CDDS) Design Specification Document: DACSES DSD - CDDS
Unit Test Plans	Work products generated during development process	<Sub System> Unit Test Plan	Security Unit Test Plan
Unit Test Results	Work products generated after Executing the Unit Test Scripts	<Screen/Notice/Report Code or batch module name>_UT	CP Debit card and Direct deposit Status (CDDS) Unit Test Results document: CDDS_UT
SIT Screen Test Case	Work products generated during System and	TC_<<Screen Name Acronym>>	TC_CPRO

Table 3: Work Product Naming Standard			
Work Product	**Function**	**Naming Convention**	**Example**
	Integration testing process		
SIT Use Case Test Case	Work products generated during System and Integration testing process	TC_<<Use Case Number>>	TC_ UC-FIN-COL-010
Functional Test Results	Work products generated after executing the Functional Test Scripts	<Screen Code> _FT	Batch Receipt Posting (BATR) Functional Test Results document: BATR_FT
Site Survey Documents	Work products generated during the network analysis	<County Name> - Site Survey	Kent County Site Survey document. Kent County - Site Survey
Solution Design Documents	work products generated during design process	<Sub System > Solution Design	Enforcement Solution Design
Batch Software Components	Work products generated during development process	BATCH_<Subsystem abbreviation>_<process name>$<procedure name>.SQL	BATCH_FIN_COLLECTIONS$SP_COLLECTIONS.SQL
Online Software Components	Work products generated during development process	<Screen code>.aspx <process name>BL.cs	Ccrt.aspx CaseCreationBL.cs

2.2.3 Storage and Accessing Control of Configuration Items

A CM system includes the storage media, the procedures, and the tools for accessing, reviewing/auditing the system and configuration items. DECSS uses Version Control Tool (TFS) as CM System. DECSS has identified the controlled TFS project/libraries for the project and describes how the code, documentation, and data of the identified baselines are physically placed under control in the appropriate TFS project/library. For each library, the format, location, documentation requirements, and access control procedures are specified. Key attributes of DECSS successful library system includes the following:

- The ability to distinguish configuration items for different work products
- A naming convention to distinguish the life cycle phase for each library
- Security to restrict access to authorized personnel
- Back-up libraries to control the recovery of prior baseline configurations
- Visibility to management and other resources
- A check-in/check-out process that restricts access to the system and ensures that only authorized personnel can move modules among the controlled libraries

The following table shows TFS projects/libraries identified in DECSS to describe how configuration items are placed under configuration management:

Table 4: TFS Projects

TFS Project	Description
DACSES Documents	This project is used to store all the documents related to but not limited to Management Plans, Procedures, Design documents, Requirements, Change Requests, Project Schedule, Testing and Training related documents
DACSESPrototype	This project is used to store artifacts related to any new prototype to be made for DECSS before implementation
DACSESWorkItem	All the workitems reside in this project including Bug, User Story, Change Request, DBCR,DSD, Issue, Maintenance Request, Requirement, Solution Design, Task, Test Case, Use Case, Test.Bug
Data Conversion	This project is used to store all conversion related script and documents
DECSS Source Dev	This project is the baseline for development code. All the developers check in their code in this project. All checked in code must be against a work item
DECSS Source UIT	This project is used to store code artifacts to release to UIT environment. After the code is tested in development project, code is promoted to this project for Unit Integration Testing
DECSS Source SIT	This project is used to store code artifacts to release to SIT environment. After the code is tested in UIT environment, code is promoted to this project for System Integration Testing
DECSS Source UAT	This project is used to store code artifacts to release to UAT environment. After the code is tested in SIT environment, code is promoted to this project for User Acceptance Testing
DECSS Source PROD	This project is used to store code artifacts to release to Production environment. After the code is tested and passed in UAT environment, code is promoted to this project for releasing to actual production environment
State_UAT	This project is for the UAT team used during design, development, and implementation phase of the DECSS project to create internal work items
DACSES	This project contains Delaware Legacy child support code

Table 4: TFS Projects	
TFS Project	**Description**
ARCSIS	This project contains code from the Arkansas child support program
NJKIDS	This project contains code from the New Jersey child support program

2.3 Configuration Control

Control is maintained over the configuration of the work product baseline. This control includes tracking the configuration of each of the configuration items, approving a new configuration, if necessary, and updating the baseline. Procedures and guidelines provided to designated project team members will show the steps needed to check out configuration items from the controlled library, revise the configuration, and check in the configuration item back into the controlled repository. These specific processes serve to maintain the baselines after they are established. The goal is to establish mechanisms that will help ensure the integrity of the baselines.

If a new CI has to be added or an existing CI has to be modified, it must be an outcome of a bug or due to a change request.

A bug is a result of any existing defects in the system or any new defect introduced by a Change Request or fix of any other defect. Bug can be identified in any environment by any team member. Once the bug is identified and verified, it has to go through regular release and testing cycle.

Change Requests are the result of the Change Request Management Process. A new Change Request is created in TFS and it has to go through regular release and testing cycle.

Process

Once a bug or Change Request is identified and approved, a TFS Work Item is created and assigned to the Development Manager. The development manager will assign these work items to a developer in the team, and the associated developer is responsible for completing the development task. The bug or Change Request work item contains a detailed description of the bug or Change Request and any supporting material such as screen shots. The developer then identifies the Software Artifacts CIs that require modification. After the Identification process and an Impact Analysis, the developer checks-out the code for modification. The developer creates/updates Software Artifact CIs in the course of their work. The developer checks-in new/updated CIs to the Source Control Tool. At check-in, the tool records a description of the change and assigns a unique revision number. The developer repeats this step until the CIs are complete. CI's then go through regular build and release process.

Database Change Request

A Database change request is a formal process to request a change in the structure of tables, views, indexes or any other database objects. Database change request are originated from either a bug or from a Change Request. Refer to the Database Development Plan for details. Implementing this request will be a coordinated effort from the DBA, Development Manager and Configuration Manager.

Physical Control

Control also includes the designation of appropriate personnel to conduct any required physical maintenance of the CM library system. Access to configuration items is restricted at a group level, and sometimes at an individual level. Permissions are managed through standards set by IRM and supporting groups.

2.4 Configuration Status Accounting

Configuration status accounting involves the recording and reporting of the configuration process. The goal of configuration status accounting is to maintain a status record of items in the baseline, thus providing traceability of all changes to the configuration items. Proper configuration status accounting answers the following questions:

- What changes have been made to the system and when were they made?
- What components were affected by this change?

Project staff document configuration status of each work product in TFS. Status information also includes the status of Change Requests that impact configuration items.

PMO is responsible for creating reports of configuration status data. Following are the reports created by PMO:

2.4.1 Weekly Status Report

This report contains the status of all in process bugs, Maintenance Requests and Change Requests. The report is generated and shared with stakeholders every week. Any changes to configuration items can be tracked using this report.

Figure 2: Sample Status Report for Bugs and Maintenance requests

For bugs and Maintenance requests, the format of the report is

ID	Title	State	External Reference	Severity	Note

Figure 3: Sample Status Report for Change Requests

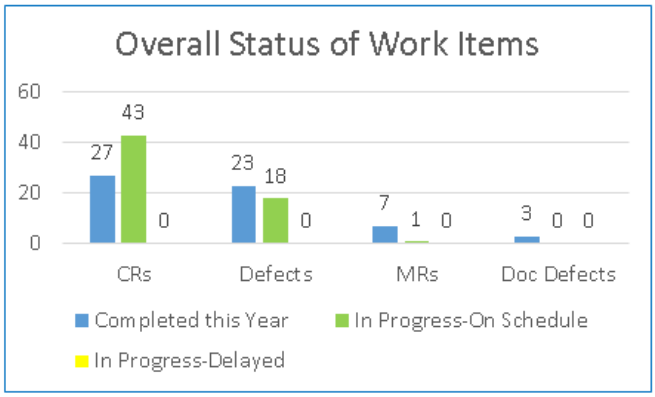
For Change Requests, the format of the report is

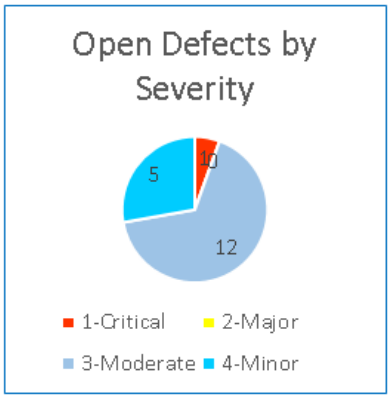
ID	Title	State	External Reference	Severity	SIT Date	UAT Date	Release Date	Note

2.4.2 DECSS Maintenance Vendor Weekly Report

This is a weekly report which contains detailed information of all the workitems in progress. This report also shows the work items completed in the reported week and work items moved across different TFS project baselines. The report also shows the completed work during last year. Below are various screenshots from the report:

Figure 4: Sample DECSS Maintenance Vendor Weekly Report





Accomplishments This Week
13 - Work items moved to SIT
6 - Work items moved to UAT

Completed Work Items This Week	
CRs	**4**
Defects	**6**

Completed Work in 2019	
CRs	**78**
Defects	**67**

Detailed Status of In-Progress Work Items

ID	Work Type	Title	State	Severity	Prod Date

2.4.3 DECSS Maintenance Vendor Team Monthly Report

This is a detailed monthly report of all the work items in any stage during past month. This report details following for the previous month

- Defects Resolved
- Completed Change Requests
- Open Work Items
- Development Work Items
- SIT Work Items
- UAT Work Items

2.4.4 DECSS Dependency List Report

This daily report is to track code dependencies between work items. If there are code dependencies between work items, it may cause release integrity issues. This report is manually ran daily by parsing all the files in TFS of work items in progress and catches all existing dependencies within code.

Figure 5: Sample DECSS Dependency List Report

DECSS Dependency List

ID	Title	Severity	State	Assigned To	SIT Date	UAT Date	Production Date	Release Order	Predecessors	Successors	Predecessors with CR#	Successors with CR#	Remarks

2.5 Configuration Verification

Configuration verification occurs after each release. Internal configuration reviews occur by checking the change requests against the code that is being verified by the configuration manager, testers, and Business Analysts. There are also continuous system checks during system testing. Reports defined above in Configuration Status Accounting section makes sure that all configuration items are properly tracked.

Configuration verifications confirm that the resulting code and documentation conform to a specified bug or change request. A configuration verification ensures that the configuration item released fulfills the requirements and is complete as delivered.

2.6 Release Management and Deployment

Release Management and Deployment includes the process of compiling work items and completing builds for the testing and production environments. It applies technical and administrative direction and surveillance throughout the project to identify and control changes to software and documentation in terms of builds.

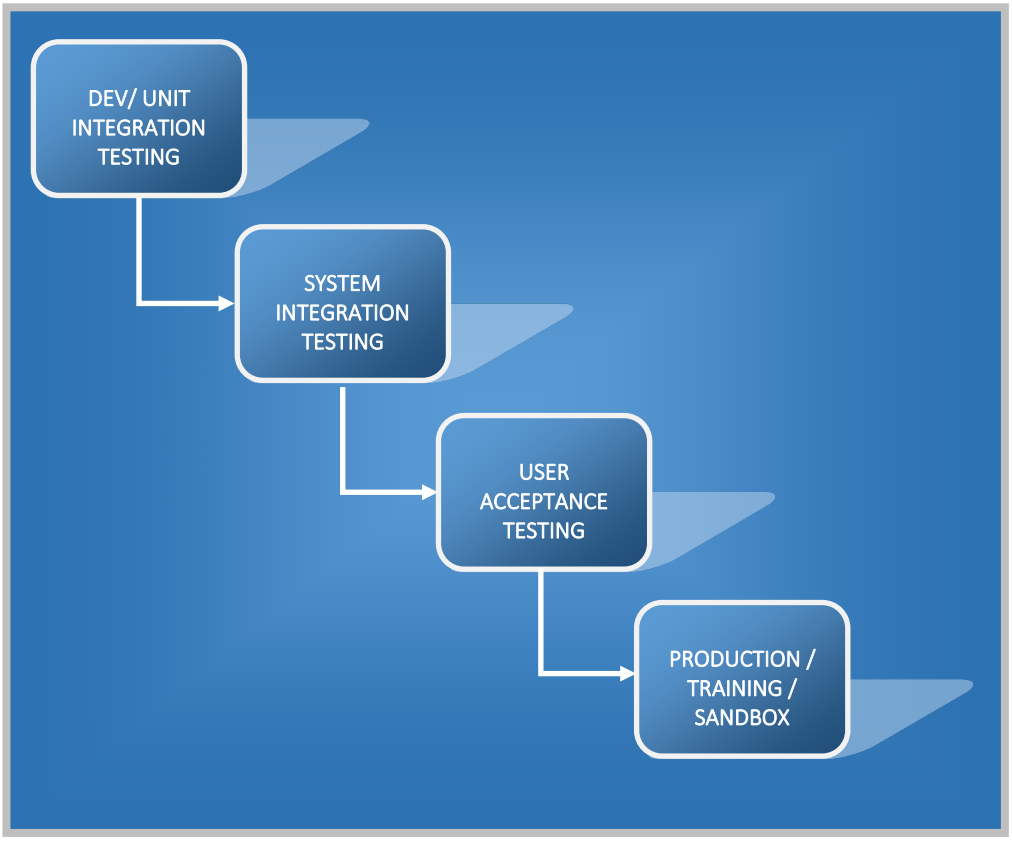
DECSS Environments

CM process should manage all the project environments. Here is the list of environments that will be under the CM control.

- Development / Unit Integration Testing
- System Integration Testing
- User Acceptance Testing / Performance Testing
- Training / Sandbox
- Production

Apart from above environments, new temporary environments are built and destroyed as per project needs such as conducting Proof of Concepts.

Figure 6: Environments and the System Promotion Model



Code Promotion Process and TFS Work Items Flow

Code Promotion is a process of promoting (transition) the Code from one environment to another environment.

Based on the forecasted release date the work item is assigned scheduled SIT, UAT and Production dates. When a developer checks in the code against a work item, a change set number gets associated with the check-in. This change set number and above-mentioned work item status are used to move CI from one environment to another environment.

Apart from regularly scheduled promotion, on-demand code promotion builds are also executed on an as needed basis.

During maintenance phase, work item types which DECSS is using for Change Management are Change Request, Maintenance Request, Database Change Request and Bug. For these work item types, Unit Integration Testing, System Integration Testing, User Acceptance Testing and informal regression testing are completed by developers, testers and users in the appropriate test environments. If problems are identified with a change, TFS Work Item is rejected and the developer is required to correct the change in the development environment, have the changes reviewed, and promote it to higher environments.

Code promotion is controlled through the TFS work item. Code promotion to various environments occurs based on the status of the TFS work item. Migration of work item are coordinated by IRM based upon approval by the User Acceptance Testers. Once approved, IRM initiates the production migration . Code promotion and configuration changes to production are segregated from the maintenance vendor and performed by IRM's Infrastructure Team.

Emergency Code Promotion

There may be situations where an identified defect in a production environment needs fixing immediately to avoid extreme situations where the application is unusable. In such situations emergency promotions may be required. During emergency promotion code may not progress through regular code promotion model, instead Configuration Manager gets required appovals from stakeholders and promote the code into production after verifying the code for correctness of the problem. This code promotion does not happen automatically through a build process; instead, it is done manually. After the emergency code fix that was promoted to production, the code will be subject to the regular testing cycle through the bug management process.

3. Tools

DECSS approach to configuration management focuses on effectively using tools and processes to control and facilitate change, that version control is met and change request requirements are verified as follows:

Software version control

Document version control

Change control management processes

The specific CM tools include check-in/check-out procedures and a configuration verification audit trail for who was responsible for the change.

3.1 Configuration Management Tools

DECSS uses following CM Tools:

3.1.1 Microsoft Team Foundation Server (TFS)

Current version control and configuration management of application system artifacts is controlled through TFS, which provides the configuration management functions needed to move and track the contents of system objects among the various environments (such as UIT, SIT, UAT, and Production). TFS provides control of configuration management secured by environment. It also allows for the flexibility of parallel development using branching. TFS tracks the system changes with timestamp, version, and user information.

TFS is used to implement and enforce the CM process. By following this controlled process, changes are defined, documented, approved, baseline, monitored, managed, and reported consistently. It provides real-time visibility of work items, allows teams to collaborate, and centrally manage assets.

Much of the Team Foundation Server user interface is offered through the Microsoft Visual Studio IDE. In addition, functionality is provided to the client through Office applications—specifically Microsoft Word, Excel and Project. There are some Web elements as well—most notably, the team projects portal and the report server. As DHSS' application lifecycle management platform is upgraded, new tools and/or versions may be implemented.

The Team Foundation Server will provide the following functionality for all managed items:

Locking and Revisions – The project needs to prevent two developers from unintentionally modifying the same file at the same time, risking the loss of one set of changes. TFS will allow developers to obtain an exclusive lock on a file through a check-in/check-out procedure.

Version Labeling – The TFS repository will permit the CM Engineer to attach a label to specific versions of CIs that comprise a baseline. A label is a string that uses the uniquely identified baseline that will be used to retrieve it. This will allow the project team to identify unique releases, repeat builds, and baseline items.

Stream - A stream is a component of a single project and a mechanism for creating and recording configurations. A stream to which a view is attached determines element(s) versions displayed in the view. A stream precisely identifies current (element) versions available for a developer to access, modify, or build.

Development Stream – Developers have their own private work area where the changes are done and Unit tested. The Development stream records project baselines. The Development stream collects all the delivered work from the Developers work area.

Branching and Merging – Branching enables CM activities for separate releases to perform in parallel. The TFS repository will allow developers to create branches of an item off the main development trunk for purposes such as enhancements and bug fixes. Once a branch is approved for inclusion in the latest release, the tool will provide developers with assistance merging branches back into the trunk by appropriately integrating the differences into a single new version.

Promotion Model – TFS will support the promotion model and allow CIs promoted as they mature.

Reporting – The tool has the ability to do code comparisons of different versions of the code, and provides the ability to generate reports on the CI's on an ad-hoc basis.

Integration with IDE – TFS has the ability to integrate with the Visual Studio IDE. This allows flexibility for the developers to have an easier checkout and check-in process. This is the tool used by the developers for development purposes.

3.1.2 Visual Studio

Visual Studio provides the tools required to design, develop, debug, and deploy Web applications, XML Web Services, and traditional client applications.

Integrated TFS environment in Visual Studio enables a wide range of change management operations to be performed directly from within the development environment, provides an integrated view of projects and increases both collaboration and team productivity.

3.1.3 TFS Team Explorer

Team Explorer is the tool used by DECSS for logging and tracking bugs. It allows the ability to log a TFS work item from creation to resolution to closure. It allows for collaboration between different teams that need to work together to implement a solution. The Team Explorer provides the following features:

Tracking TFS Work Items – The tool will allow tracking of TFS work items through a project-defined lifecycle. This includes allowing business analysts, developers and testers to add comments related to work done on a TFS work item. Additionally, it will track what people have been involved in the process.

Reporting – The tool will generate reports with details about the overall TFS work item population (number outstanding, closed, assigned to given teams, older than a certain date, etc.).

Integration– The Version control component and work item tracking component are integral features of the Team Foundation Server. This allows developers to tie back actual CI changes and version numbers to the Team Explorer work item. Therefore, it provides a mechanism for deployment builds and errorless promotion model.

Source Control Explorer and Team Explorer work together to allow developers to define and manage (as activities) changes to software assets. Through TFS's Unified Change Management (UCM) capability, file versions in Source Control Explorer are associated with work items in Team Explorer. Stakeholders perform operations directly on work items rather than on file(s) collections associated with them. This work item based approach to change and configuration management helps developers to work on the correct files and file versions. UCM allows developers to create collections of file versions, or change sets, for each individual change.

3.1.4 PowerShell

PowerShell is a configuration management and task automation tool, which consists of a command-line shell. It has its own scripting language. DECSS uses PowerShell to build automated deployment script, which builds a package from Team Foundation Server and deploy on respective environments.

4. Training

All team members are required to read the Configuration Management Plan as part of the project onboarding. Additional CM training may be conducted, as needed, throughout the life of the project to provide team members with continued instruction. The users will also be trained on any required CM tools.

5. Definitions

This section provides the definitions of all terms, acronyms, and abbreviations related specifically to the Configuration Management Plan.

Build: The process by which the Software Artifacts in a baseline generate into the system components of a release.

Bug: TFS Terminology for a Defect.

Configuration Baseline: A set of Configuration Items (CI) required for producing the system. Baselines can contain all types of CI's: Software Artifacts, Deliverables, and Work Products. Releases are generally built from baselines.

Configuration Item (CI): Any item that is versioned and managed under CM control. CI's are categorized as Software Artifacts, Deliverables, or Work Products, and are baselined at a defined point in time in the system lifecycle.

Configuration Management (CM): CM is a disciplined approach to managing and controlling the evolution of system development.

Integrated Development Environment (IDE): This is a type of computer software, which assists computer programmers in developing software. This is also known as integrated design environment and integrated debugging environment. IDE's normally consist of a source code editor, a compiler and/or interpreter, build-automation tools, and (usually) a debugger. At times, integrates a version control system and various tools to simplify the construction of a GUI.

Promote: The action of associating a specific version of a CI as part of a baseline, done by labeling.

Release: In a technical context, a grouping of software items that are made available to the user community at a point in time.

Repository: A central place where data is stored and maintained. A repository can be a place where multiple files are located for distribution over a network.

Software Artifact: An artifact is an object, which is developed for the system development that affects the execution of the system. Software Artifacts include items such as source code, configuration files, scripts, code libraries, and installed applications. Software Artifacts are one of the three categories of CIs.

Stored Procedure: This is a MS-SQL program, which is already compiled and stored or residing in the SQL Server database.

Streams: A stream is a component of a single UCM project and a mechanism for creating and recording configurations. A stream to which a view is attached determines element(s) versions displayed in the view. A stream precisely identifies current (element) versions available for a developer to access, modify, or build.

Version: The discrete state of a CI stored in the Version Control Tool at a point in time. Each CI will have one to many versions.

Work Product: Artifacts of system development that are not specifically Deliverables or Software Artifacts. Examples may include design documents, supporting documents, test scripts, test results, training materials, etc.

MV: Maintenance Vendor