New York Health Benefit Exchange

Detailed Design Review Summary for
9.4.5 Configuration Management Plan
October 9 & 10, 2012

Item Number	**Topic**
9.4.5	Configuration Management Plan

CSC

New York State
Department of Health

New York Health Exchange Project

CSC

9.4.5 Configuration Management (CM) Plan

Document Number: 528500-20120813000
Contract Number: FAU #1106211137
Release Version 2.0
Date Submitted: September 5, 2012

VERSION HISTORY

Version Number	Implemented By	Revision Date	Approved By	Approval Date	Description of Change
1.0	CSC / L. Davis, R. Davis	08/17/2012	CSC / S. Garner, C. Adams		Initial Publication
1.1	CSC / L. Davis, R. Davis	08/30/2012	CSC / S. Garner, C. Adams		Updates based on Cognosante review feedback
2.0	CSC / L. Davis, R. Davis		Carl Schell	09/05/12	DOH Approval Received

Document Distribution

Any delivered versions of the Configuration Management Plan document will be stored in the deliverable tracker section of the NY-HX SharePoint repository, and will be available to all via this medium.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

This Configuration Management (CM) Plan (CMP) defines the CM program to which the New York State Health Benefit Exchange (NY-HX) Program adheres. The plan summarizes the roles and responsibilities of CM as it relates to the various functional groups and Sprint Teams working on this project. This document will be baselined and reside in the program Process Asset Library on the NY-HX SharePoint located at https://workspace.nyhx.emedny.org/pal/.

1.1 Purpose

The overall objective of a Configuration Management (CM) Plan is to document and inform project stakeholders about CM with the project, what CM tools will be used, and how they will be applied by the project to promote success. The NY-HX CM Plan defines the project's structure and methods for:

- Identifying, defining, and baselining configuration items (CI)
- Controlling modifications and releases of CIs
- Reporting and recording status of CIs and any requested modifications
- Ensuring completeness, consistency, and correctness of CIs
- Controlling storage, handling, and delivery of the CIs

1.2 Goals

The major goals for CM implementation and management are as follows:

- To support performance and project activities outlined in the NY-HX Project Management Plan
- To optimize continued enhancement of CM and data management processes through project change and improvement activities

This plan covers the CM concepts used to achieve and maintain configured baselines through the various stages of the Design, Development, and Installation (DDI) process for the NY-HX. The Software Development Life Cycle (SDLC) for the NY-HX follows an Agile process. Agile practices include multiple builds that require packaging and deployment of code in support of those builds. This plan describes the complex interplay of hardware, code, and data configurations managed from initial design and development through to final system deployment. This includes CM interaction with the Software Development staff, technical management, testing teams, and quality assurance (QA). The plan specifies the CM tools and system utilities used to establish the controlled baselines. This plan applies to all CM functions throughout the project life cycle.

1.3 Plan Organization/Maintenance

This document is organized into the following sections:

- Section 1 – Describes the plans purpose, plan organization, cites reference documents, and describes Configuration Management Tools.
- Section 2 – Identifies the project organization for the NY-HX, describes the operational units, and the CM responsibilities shared by each.
- Section 3 – Describes the generic method for identifying and controlling configuration items (CIs) and baselines.
- Section 4 – Identifies configuration change control, references policies, generic procedures, and standards that will be developed to control established baselines and to process proposed and approved changes to established baselines (i.e., configuration controls).

- Section 5 – Describes the Configuration Status Accounting (CSA) records and reports used by the project.
- Section 6 – Describes data management identification, management, and maintenance.
- Section 7 – Describes methods for evaluating the effectiveness of the CM program (e.g., audits).
- Section 8 – References Master Project Management Deliverables Glossary & Acronym List
- Section 9 – References Master Project Management Deliverables Glossary & Acronym List.

The NY-HX CM team is responsible for the maintenance of this document and ensures that the document is reviewed and updated at least annually.

The CM manager oversees control and maintenance of this CMP. The CM manager will review and update this document at least annually throughout the life of the project. More frequent updates may be made, as needed, to accurately reflect changes on the project. Any NY-HX staff member may request a change to this CM Plan. All changes to the CM Plan should be vetted and approved through the NY-HX CSC Change Control Board (CCB).

The policies and procedures described in this document are systematically being implemented, in accordance with the Agile iterative and incremental approach, and based on alignment with the scrum team activities. Projects, teams, roles and permission have already been implemented in the Rational tools, and work items such as Epics, Stories and Tasks, are currently being entered by each scrum team. Developers are starting to place their code under source control, and a proof of concept end-to-end build and deploy has been run successfully. The CM process will continue to be updated and enhanced until it is fully implemented as described in this document.

1.4 Referenced Documents

Reference materials to support the CM efforts on this program are found in the following documentation, as shown in Table 1.

Document Name	Location	Issuance Date
Funding Availability Solicitation (FAS) for the New York State Health Benefit Exchange, Tab 6.15 Use of IBM Rational Suite	http://www.health.ny.gov/funding/rfp/1106211137/	2011
NY-HX Change Control Management Plan	https://workspace.nyhx.emedny.org/pal/Forms/group.aspx	Future Deliverable
NY-HX Scope Management Plan	https://workspace.nyhx.emedny.org/pal/Forms/group.aspx	Future Deliverable
NY-HX Project Management Plan	https://workspace.nyhx.emedny.org/pal/Forms/group.aspx	Future Deliverable
NY-HX Data Management Plan	https://workspace.nyhx.emedny.org/pal/Forms/group.aspx	Future Deliverable

Table 1: Referenced Documents

1.5 Configuration Management Tools and Processes

1.5.1 JAZZ Collaboration Environment

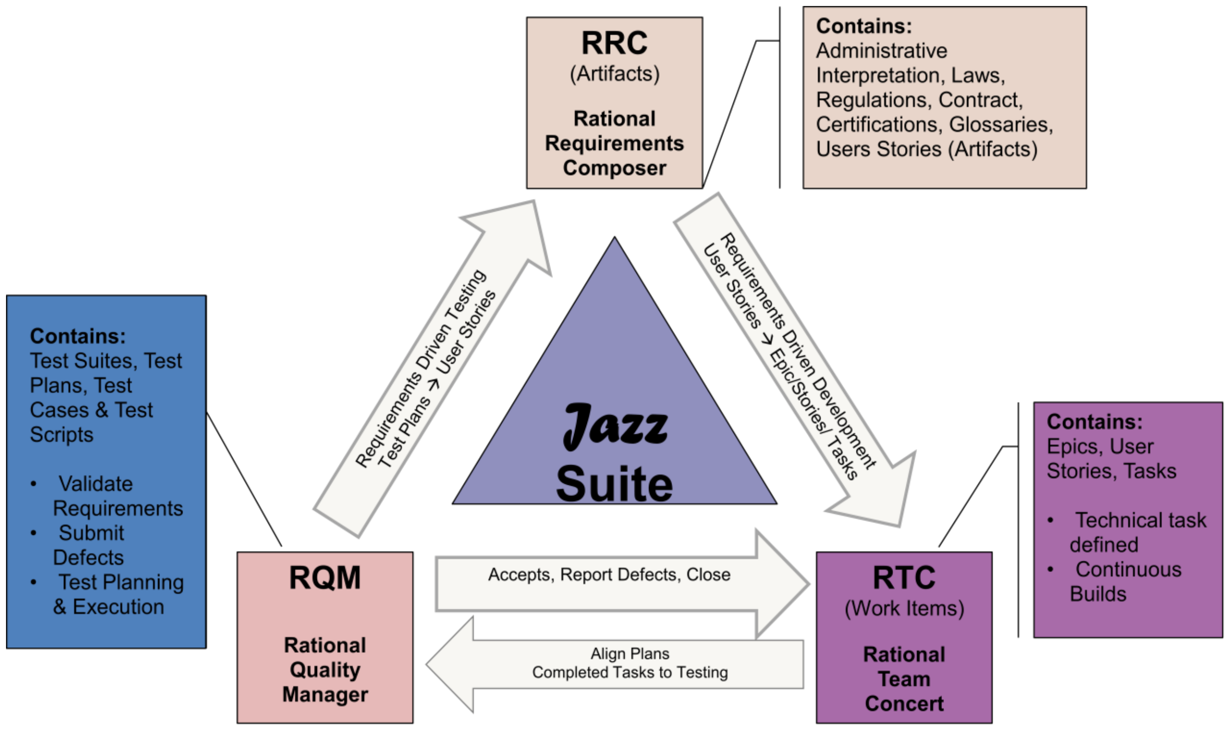

RRC
(Artifacts)
Rational Requirements Composer
Contains:
Administrative Interpretation, Laws, Regulations, Contract, Certifications, Glossaries, Users Stories (Artifacts)
Contains:
Test Suites, Test Plans, Test Cases & Test Scripts
• Validate Requirements
• Submit Defects
• Test Planning & Execution
Requirements Driven Testing
Test Plans → User Stories
Jazz Suite
Requirements Driven Development
User Stories → Epic/Stories/ Tasks
Contains:
Epics, User Stories, Tasks
• Technical task defined
• Continuous Builds
RQM
Rational Quality Manager
Accepts, Report Defects, Close
Align Plans
Completed Tasks to Testing
RTC
(Work Items)
Rational Team Concert

Figure 1: JAZZ Collaboration Environment

The Jazz Platform depicted in the above figure, provides a collaborative environment whereby the Rational Tools (RTC, RRC and RQM) are linked using a LifeCycle Project. This linkage allows for traceability of test plans and test cases in RQM to the requirements in RRC and the work items (epics, stories and tasks) in RTC. Likewise, the work items in RTC may be traced back to the requirements in RRC and the test cases in RQM. In addition to traceability, the Jazz Platform fosters communication between all users of all tools in the Rational Suite.

1.5.2 Requirement Elicitation and Prioritization

The core requirements for the NY-HX project, as outlined in the FAS and the contract between CSC and DOH, will be captured and tracked using products from the Rational tools suite. Each high level requirement will be documented in the Rational Requirement Composer (RRC), and further defined by creating a set of "user stories," which will be tracked in Rational Team Concert (RTC). Defining requirements via user stories, is an Agile approach to eliciting necessary system functions and features from a user perspective. The majority of user stories will be created by the individual scrum teams, in conjunction with their Product Owners and SME's (subject matter experts represented by NY DOH) during the initial Sprint 0 planning phase of the individual tracks.

User stories comprise the product inventory for each track of work, and detail what each scrum team will be responsible for accomplishing during the lifecycle of the NY-HX project.

Development of the user stories will occur in 2-week "sprint" cycles that will include a sprint review at the end of the 2 weeks, in order to present the Product Owner with what was completed. After the sprint review, the Product Owner and/or team may identify additional user stories that may be necessary for the product inventory. The process for managing changes to the product inventory is described in detail in the Change Control Management Plan (CCMP).

Throughout the sprint cycles, the Product Owners will be responsible for grooming the product inventory (prioritizing, adding or removing stories, and providing further details as needed), as well as ensuring the teams are working on stories that are deemed high priority and high value for the client. Each release will comprise 11 sprints, with additional sprint 0 planning sprints that will occur prior to each release. This will allow the Product Owner and scrum teams to perform further grooming, prioritization and analysis of the product inventory.

1.5.3 Rational Requirement Composer (RRC)

RRC helps teams organize, prioritize, track, and control changing requirements of a system or application. It is a source of reference for all requirements details and status for all project resources throughout the project.

As the repository for all details of each project requirement, the tool will be used for:

- Data entry of baseline requirements and approved changed requirements.
- Tracking Requirements attributes the as requirements definition evolves through the project life cycle.
- Progressive Elaboration of high level requirements into more granular requirements.
- Tracking of requirement status throughout the project life cycle.
- Aligning requirements with the project schedule in accordance with the Agile methodology

1.5.4 Rational Quality Manager (RQM)

RQM provides a collaborative application lifecycle management environment for test planning, construction, and execution. The tool will be used to:

- Enable quality assurance team to track all aspects of the quality assurance effort
- Create a dynamic test plan that contains all information pertaining to the quality assurance effort, such as goals, schedules, milestones and exit criteria as well as links to associated test cases, requirements and development work items
- Allow for manual test authoring and execution, test lab management, test execution, reporting and defect management

1.5.5 Rational Team Concert (RTC)

RTC provides a complete collaborative development environment that facilitates work item management, build management, integrated reporting, and process support. RTC has features such as change tracking, process automation, planning, reporting, testing and lifecycle traceability, which enables better insight, predictability, and control of software and systems development, along with their associated changes.

RTC's collaborative change management capabilities and benefits include:

- Rich and fully customizable work item attributes and capabilities to track and coordinate tasks and workflows for any purpose, governed by the team's selected processes
- Agile project planning capabilities to extend the traditional notion of change management from just work items, to include planning and tracking by linking work items to actionable plans
- Automated dashboards and reporting for transparency and monitoring
- Requirements change management and traceability to the project team
- A collaboration hub for a distributed decentralized enterprise change management view

1.5.6 Rational Build Forge (RBF)

Configuration Management (CM) will use Rational Build Forge, an "agnostic" software that technically can be hosted on any platform. Rational Build Forge, which is basically a script engine or automation framework, will be used for build and release management. Closely coupled with the requirements management tools is the ability to link quality testing to the shared database. By directly linking testing to requirements, the CSC team will be able to achieve a high level of customer satisfaction and greatly accelerate the build-test-debug cycle. This integrated workflow will speed the time to resolution.

1.5.7 Maven

Java Software Build Utility which dynamically downloads Java libraries and Maven plug-ins from one or more repositories. Maven provides built-in support for retrieving files from repositories, and can upload artifacts to specific repositories after a successful build. Maven is built using a plug-in-based architecture that allows it to make use of any application controllable through standard input. Maven provides Plug-ins that allow for software dependency management reporting, performing software builds, performing release management tasks such as uploading java artifacts.

1.5.8 Nexus

Java Maven Artifact Repository which stores in-house developed working and finalized copies of java archives (jars) dependencies used by java projects that utilize Maven build scripts. Provides an interface for obtaining 3rd party java archives from publicly available maven repository archives.

1.5.9 NY-HX Sharepoint

CM will store the audit artifacts in Microsoft SharePoint which is located in the NY-HX workspace.

2 PROJECT ORGANIZATION AND RESPONSIBILITIES

CM supports the functions of the software development, testing, and data center operations teams. The Configuration Manager provides oversight for these functions, and Configuration Analysts are responsible for implementing CM across the project. This includes managing, tracking, and controlling all changes to project baselines through the project life-cycle phases to ensure proper configuration and integrity at each baselined stage. Specifically, CM functions include:

- Maintaining configuration control over all appropriate NY-HX deliverables
- Maintaining controlled test environments for NY-HX testing
- Maintaining NY-HX hardware and software baselines
- Maintaining records describing the various hardware, software, and data configuration items
- Maintaining CR records and NY-HX defect reports
- Supporting QA audits of compliance with the CMP and related standards and procedures
- Auditing the hardware, software, and documentation baselines for completeness, correctness, and consistency
- Coordinating the effort with and supporting the Architecture, Development, and Test teams for the delivery and installation of the hardware components and software products to the Rensselaer Data Center (RDC)

2.1 Project Responsibilities

2.1.1 Program/Project Management

The overall responsibility of program/project management is to ensure adequate staffing, budget, facilities, and tools are made available for effective configuration management and control. Program/project management is also responsible for reviewing CM activities through weekly and monthly status reporting, and as necessary to identify and create processes, schedules, and technical issue resolution.

2.1.2 Internal and External Stakeholders

Internal and external stakeholders share equal responsibility for following and implementing CM practices and activities outlined in this document, ensuring CIs are identified and maintained, and that coordination and cooperation between functional roles at all organizational levels are maintained. The following section outlines specific roles and responsibilities for internal (project personnel) and external (DOH customer and user) stakeholders.

2.1.3 Roles & Responsibilities

The CM team will report to the PMO. This ensures the independence of CM and supports the delivery of quality products by minimizing or eliminating the introduction of uncontrolled code.

Other roles that may work in conjunction with, or impact CM for the NY-HX project, are identified in the table below, along with their associated responsibilities. In this model, the Change Owner may be any member of the scrum team, including the Product Owner that identifies a potential change.

Key: R = Responsible; performs the task A = Accountable; ultimately answerable for completion of task C = Consulted; provides information or assistance I = Informed; kept apprised of the activity	Product Owner \SME	Scrum Master	Developer	Tester	Configuration Mgmt	Architecture
Create Work Items (Epics, Stories, Tasks)	CI	RA			C	
Prioritize Work Items	C	RA				
Schedule Work Items	I	RA			C	
Implementing Work Items		A	R		C	
Building Work Items(Maven)		I	R		A	CI
Deploying Work Items(Build Forge)		I	C	C	RA	R
Testing Work Items		I	C	RA	C	
Creating Defects		I	C	RA	C	
Approving Work Item Completion	R	A				
Closing Work Items	I	RA	R	R		

Table 1: RACI Chart for CM

2.1.4 Configuration Management Personnel

The NY-HX Configuration Management (CM) team consists of the Configuration Manager and supporting Analysts. The Configuration Manager manages the definition, development, maintenance and distribution of configuration management plans and procedures. He\She guides the execution of configuration issues in a full hardware/software lifecycle. He\She supports configuration management activities for various projects including production hardware support; provides support to the software development environment. The CM Analysts are responsible for managing the hardware and software throughout the entire project life cycle. This includes ensuring that the integrity of all source code is maintained and that a valid audit trail exists for all changes to hardware and source code. The team is responsible for moving only documented and authorized source code within the testing and production environments. The Scrum team maintains the schedule for when CM is to promote each release to each environment. They are also responsible for providing CM and the Database Administrators (DBAs) with a list of the changes to be promoted into and deleted from each environment. The CM team participates in the coordination of migration activities with the DBA and Production Operations areas, and provides resources for the restoration of elements in a disaster recovery situation.

2.1.5 Training

Initial program tools training with the Rational suite, has been provided by IBM trainers. Additional NY-HX Rational training related to CM activities, will be the responsibility of the CM Team. Training will be conducted on an annual basis, or as needed to provide support to users. Training and support may include mentoring, providing training materials, Rational guides, classes and one-on-one coaching.

2.1.6 Development

The CM team supports the development teams in the configuration control and promotion of software components from the development and unit test stage through the various test stages to the operational environment. The development teams are supported by the CM team in ensuring that changes to common software components supporting multiple functions are incorporated in all applicable areas. The CM team also assists with CM tool issues encountered by the development teams.

2.1.7 Testing

The CM team supports the Testing team in establishing and maintaining test environments. The CM team also assists with the configuration of controlled elements as they are introduced into the test arenas.

2.1.8 Quality Assurance

The QA team reviews, monitors, conducts audits and raises risks as needed, to ensure the CMP guidelines are followed.

2.1.9 NY-HX Change Control Board

The NY-HX Change Control Board (CCB) serves as the coordinating body for all NY-HX defect reports and relevant Technical Baseline CRs. The NY-HX CCB may be comprised of members from CM, Architects, Developers, PMO, DOH and QA. The board will also determine the disposition for related changes to permanent baselines (i.e., functional, allocated, development, and product baselines).

2.1.10 New York State Department of Health

The New York State Department of Health (NY DOH) or its designee may review Status Accounting Reports of controlled elements and CRs to ensure that proper CM procedures are followed.

2.1.11 RDC Change Control Review Board

All hardware and software product installations at the Rensselaer Data Center (RDC) are reviewed and approved by the RDC CCRB. The board coordinates and schedules the installations to ensure that they meet all data center and customer requirements. In addition, all software deliveries to the NY-HX operational environment are coordinated and scheduled with the approval of the RDC's CCRB.

3 CONFIGURATION IDENTIFICATION

3.1 Identifying Configuration Items

3.1.1 Naming Standard

The development teams will use Rational Team Concert (RTC) for configuration management and Rational Build Forge to manage the build and release process. Naming conventions are defined for the following items:

Artifact	Convention
RTC Streams	Streams will use the Team's name for team streams and "Integration" for integration stream. Ad hoc streams are prefixed with their team name.
RTC Repository Workspaces	Begin with the user's RTC username and then may contain any user or team specific naming.
RTC Build Definition	Team builds named with team name, integration build named "Int"
Snapshots/Baselines	Snapshots are labeled with the following conventions: <Team>_<Build_Type><yyyymmddmmss> where the Team is one of {PM, EEI, EEB, FM, Int} and build type is "T" for development team builds, "I" for integration builds, and "R" for release.
Build's generated artifacts	The deployable ear file resulting from a build will be labeled <snapshot name>.ear, the same name as the snapshot used to create the Ear file. Ear files will also contain a bill of materials (BOM) listing the Ear file's contents.

Table 2: CM Naming Conventions

3.1.2 Baselines

In RTC, all CIs are stored in Components, and modifications to CIs are shared through Streams. RTC remembers configurations by creating a snapshot on a Stream. Snapshots create baselines on all the components contained within the Stream. This process allows different Streams to be working on different component baselines.

Each team and integration build will automatically create a snapshot in the team or integration Stream. Build Forge will be used to deploy release builds to the Nexus repository and on to the DEV environment as requested by the Agile development teams. Build Forge will also be used to deploy the built artifacts to other environments (e.g., SIT, UAT). CM will mark those snapshots when they are deployed to each of these other environments. CM will change the state of all work (User Stories, Defects, etc.) included in the snapshot to indicate the work items are now in DEV (or SIT, or UAT). All build & deploy history will be tracked within RTC.

3.1.3 Storage & Retention

All CIs are stored and versioned in RTC. RTC uses a DB2 database that is backed daily. The disaster recovery plan is referred to in the Environment section 5.4.3.

3.1.4 Configuration Control

During Sprints, the teams will add and modify CIs in the RTC Repository Workspace. Since the Repository Workspace is located on the server, every saved change is immediately placed under CM control and is mirrored on the redundant server.

Changes are delivered to the team's stream following the team's Agile process which may include reviews from peers (code reviews) and/or testers (informal test analysis). Team and integration builds run continuously creating snapshots to remember those configurations. The Integration build at the end of each Sprint marks the release to the DEV environment.

The teams' Product Owners are responsible for maintaining the product inventory, which includes regular review and prioritization, as well as adding and removing items (the inventory may be comprised of a combination of user stories, epics, CR's, technical stories, and spikes). The Product Owner ultimately accepts or rejects changes to the inventory. At the beginning of each Sprint, a team's Product Owner works with the team to select work the team will commit to for that Sprint.

3.1.5 Configuration Promotion

The team's Product Owner works collaboratively with the team to assesses approved changes and/or problem reports continuously throughout the development cycles.

The cross-functional development teams, following Agile best practices, include representation from development, test, architecture, DBA, CM, and Product Owner (client). The Agile development teams determine when releases move to DEV, from DEV to SIT, and from SIT to UAT.

3.1.6 Tracking & Controlling Changes to CI Baseline

All CI changes will be associated with their work item (User Story, Defect, etc.) which are implemented and built and unit tested in the team and integration builds. The changes are tracked by each build's snapshot. Snapshots are verified by the test team when deployed to the DEV and other controlled environments (SIT, UAT, etc.).

To support Agile practices, Testers (and possibly Product Owners) will have the ability to review developer's work prior to deployment to the controlled DEV environment. Team members will strive toward a test driven development (TDD) approach, whereby developers and testers work closely together to develop test cases prior to, or in conjunction with functional code. When developers complete a User Story, testers may informally exercise the new functionality by viewing the change on the developer's WAS server and provide feedback for any necessary code modifications. The goal of this collaborative approach is to reveal issues early, and adapt quickly; prior to deployment in the controlled environment.

3.1.7 Application Software Environments

CSC has selected the IBM Rational Suite to provide robust programmatic control of service development. CSC will use the IBM tools to form an integrated development environment that will manage requirements from initial concept through the entire SDLC process. Rational Team Concert and Rational Build Forge will be utilized for Configuration Management.

The CSC SDLC will be comprised of the environments listed in Figures 1 and 2. These environments contain the required hardware, software, network, and database capabilities and ensure an isolated production-ready environment.

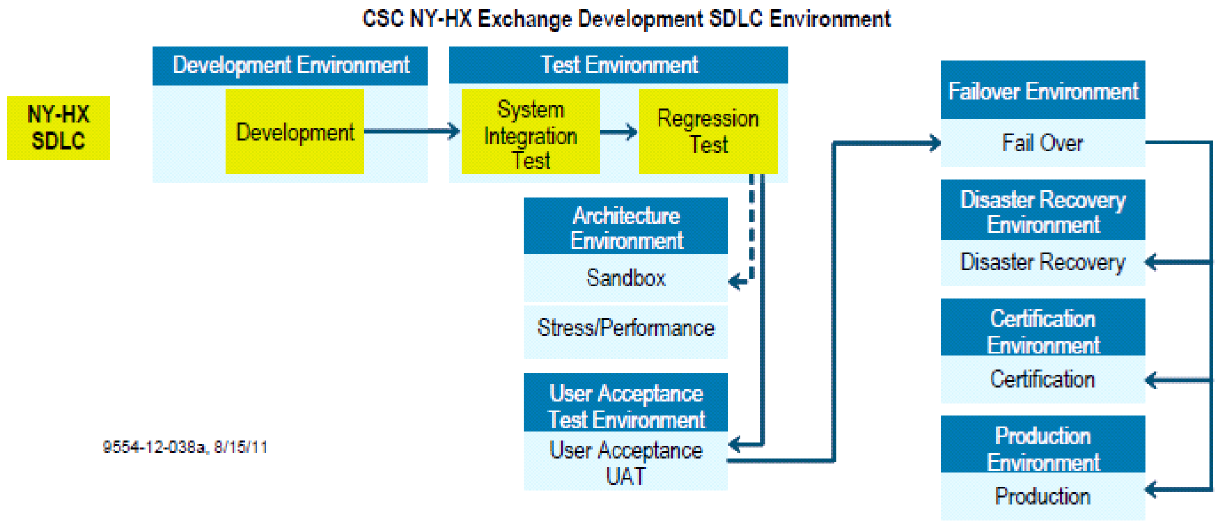
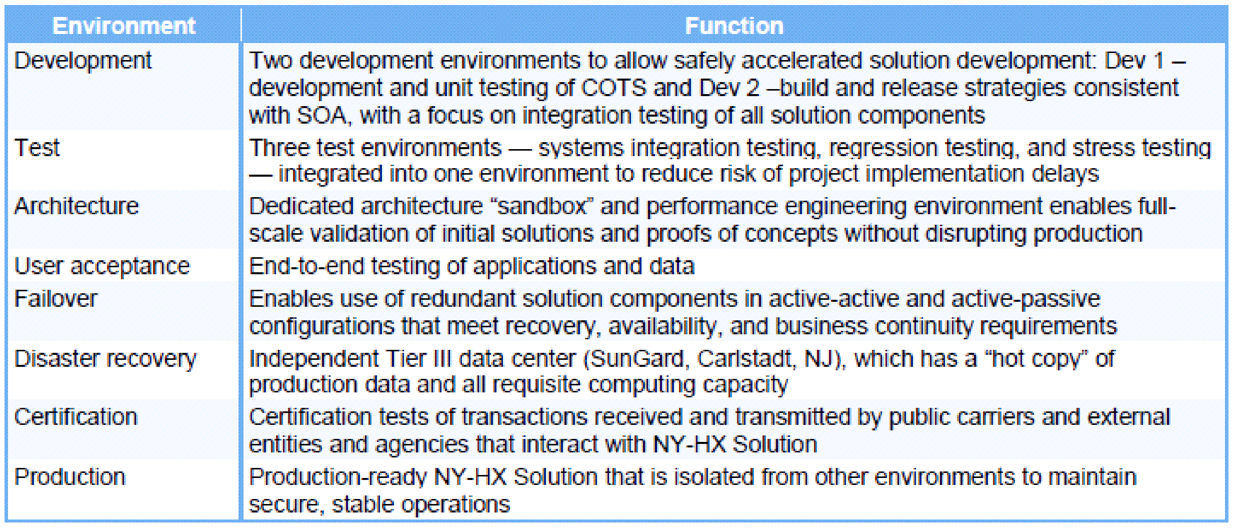


Figure 2: NY-HX Exchange SDLC Environments



Environment	Function
Development	Two development environments to allow safely accelerated solution development: Dev 1 – development and unit testing of COTS and Dev 2 –build and release strategies consistent with SOA, with a focus on integration testing of all solution components
Test	Three test environments — systems integration testing, regression testing, and stress testing — integrated into one environment to reduce risk of project implementation delays
Architecture	Dedicated architecture “sandbox” and performance engineering environment enables full-scale validation of initial solutions and proofs of concepts without disrupting production
User acceptance	End-to-end testing of applications and data
Failover	Enables use of redundant solution components in active-active and active-passive configurations that meet recovery, availability, and business continuity requirements
Disaster recovery	Independent Tier III data center (SunGard, Carlstadt, NJ), which has a “hot copy” of production data and all requisite computing capacity
Certification	Certification tests of transactions received and transmitted by public carriers and external entities and agencies that interact with NY-HX Solution
Production	Production-ready NY-HX Solution that is isolated from other environments to maintain secure, stable operations

Figure 3: NY-HX Exchange SDLC Environments Function

4 CONFIGURATION CHANGE CONTROL

4.1 Configuration Change Control

The NY-HX project currently utilizes 4 Agile development teams, one for each of the following tracks: Plan Management (PM), Eligibility Determination and Enrollment for Individuals (EEI), Small Business Health Options (SHOP), and Financial Management (FM). Agile teams are cross-functional and consist of Developers, Testers, Business Analysts, Architects, Tech Writers, CM, Scrum Masters, and Product Owners. Teams work independently, frequently sharing their changes with teammates. In addition, each team may integrate their changes with other teams one or more times each Sprint.

Each team will have their own collaboration area to share changes, which is called a Stream in RTC. The figure below shows the four team streams as well as an Integration stream. Each developer will have their private Repository Workspace that flows changes to and from the stream (Al's workspace in the figure).

To support continuous build practices, each stream (team and integration) will have a build Repository Workspace used by the build process to continuously build and test changes (the Build workspaces in the figure).

The DEV_TEST environment may be a Sprint behind the current development which is in the developers workspace in RTC. In cases where the testing team working in DEV_TEST finds a critical defect, development may need to make changes and deploy them directly to DEV_TEST rather than through the Integration stream. The Dev-Maint stream is always in sync with the system version running in DEV_TEST, and provides development team members the ability to make changes and send them directly to DEV_TEST. The Integration build sets the Dev-Maint Stream's snapshot to the version being deployed to DEV_TEST.

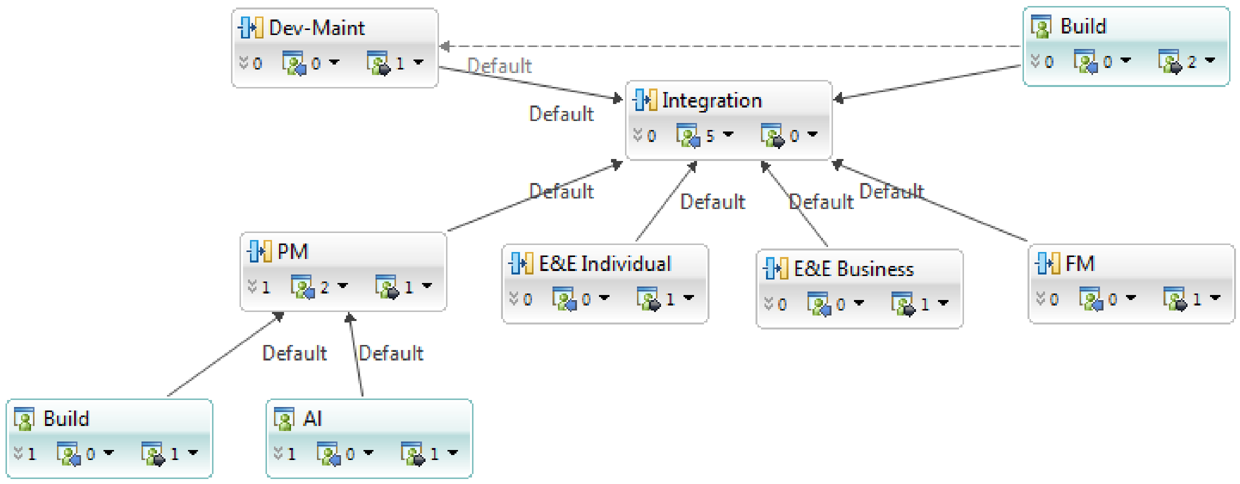


Figure 4: RTC Stream Flow

Process	Tools & Techniques
1. Developer loads team's source artifacts	RTC loads artifacts from the team's Stream to the Developers Eclipse Workspace (aka sandbox)
2. Developer receives work item (story, task, defect, etc.)	RTC presents users assigned work for the current Sprint
3. Developer makes appropriate changes to source artifacts	RTC auto-checks the files into the RTC repository as developer saves changes. RTC associates changes to the developer's current work item.
4. Prior to delivering changes to the team, developer ensures their Repository Workspace in sync with teams' Stream	RTC presents incoming changes and conflicts. Developer must accept changes and resolves conflicts (auto/manual merging) before delivering their changes to the team's Stream.
5. Prior to delivering changes, developers run team's Unit Tests	Developer runs test locally or could leverage the team build to build/test changes in the RTC Repository Workspace
6. Prior to delivering changes, testers may review the change to provide initial, informal feedback	Testers can view the developer's change by connecting to the developer's WebSphere server.
7. Developer delivers change to team's Stream	Developer delivers change set from their Repository Workspace to the team Stream in RTC
8. Developer transitions work item to next state (towards Done)	New -> Implement -> DEV_Test -> SIT --> UAT -> Done
9. Team-level continuous build includes change in the next execution	Build Forge team builds running continuously automatically detect changes since the last execution and include them in the team-level build. The build uses the team's Maven build scripts to compile, package, deploy, and unit test the team-level code.

Table 3: CM Processes for Developer Collaboration

The table below describes the team-to-team collaboration. All teams periodically integrate their team-level changes to the Integration stream and pull Integration changes back to their team streams. A developer from each team (the Integrator) will merge changes between their team and the Integration stream. The Integration stream has a continuous build that runs automatically, detecting any changes and automatically building and testing them using the Jazz Build Engine in RTC along with basic smoke tests. Build status is reported in RTC.

Process	Tools & Techniques
1. Integrator delivers team changes to the Integration stream	The Integrator ensures their Repository Workspace is current with the team's Stream and then changes the Workspace's "flow target" to the Integration Stream. RTC shows any incoming changes and conflicts from other teams. The Integrator (seeing any incoming changes indicating that another team has delivered their changes and will collaborate with that team) merges changes, resolves any conflicts, and delivers changes to the Integration stream
2. Integrator delivers other team's changes back to their team's stream	The Integrator changes their Repository Workspace (which contains the Integration Stream changes) flow target back to their team Stream and delivers the Integration changes to their team. If the team Stream has new changes, the Integrator must merge them prior to delivery.
3. Team members accept Integration changes	After the Integration delivery to the team Stream, all team member accept (and possibly merge if RTC indicates a conflict) the incoming Integration changes
4. Integration-level continuous build includes change in the next execution	The Build Forge Integration build runs continuously automatically detect changes since the last execution and including them in the integration-level build. The build uses the team's Maven build scripts as well as integration scripts to compile, package, deploy, and test the integration-level code.

Table 4: CM Processes for Team Collaboration

Process	Tools & Techniques
1. CM runs the integration build for deployment into Nexus	CM starts the automated build scripts to build and deploy to Nexus using Build Forge. The Integration build compiles, packages, tests, and (if successful) deploys the artifacts into the Nexus repository. If the build fails, the developer and scrum master will be automatically notified for resolution.
2. Deploy from Nexus to DEV_TEST	Build Forge invokes the process to deploy from the Nexus repository into the DEV_TEST environment. Build Forge updates the Dev_Maint Stream so it is in sync with the version in DEV_TEST. The DEV_TEST deployment process may require manual steps for activities such as database modifications or other environment configuration. CM updates the state in all work items included in the snapshot to indicate they moved to DEV_TEST.
3. Deploy to SIT, UAT, etc.	Step 2 will be repeated for deployment into other environments and tracked using the work item state to indicate deployment to SIT, UAT, etc.

Table 5: CM Processes for Deploy to DEV_TEST Environment

Process	Tools & Techniques
1. Development works on critical DEV_TEST fix	Developer suspends current work and changes Repository Workspace to Dev-Maint Stream.

Table 6: CM Processes for Critical Fixes to DEV_TEST Environment

4.2 Change Management

The change management diagram below, provides an overview of the Change Request workflow for Enhancement Requests, Defects, or scope changes (such as new User Stories), and how these items will be implemented, tested and tracked using RTC, once they have been approved by the CCB and/or in conjunction with the Scope Control Board (SCB). For additional information on the Change Management Process, refer to the NY-HX Change Control Management Plan or the Scope Management Plan located in the NY-HX SharePoint.

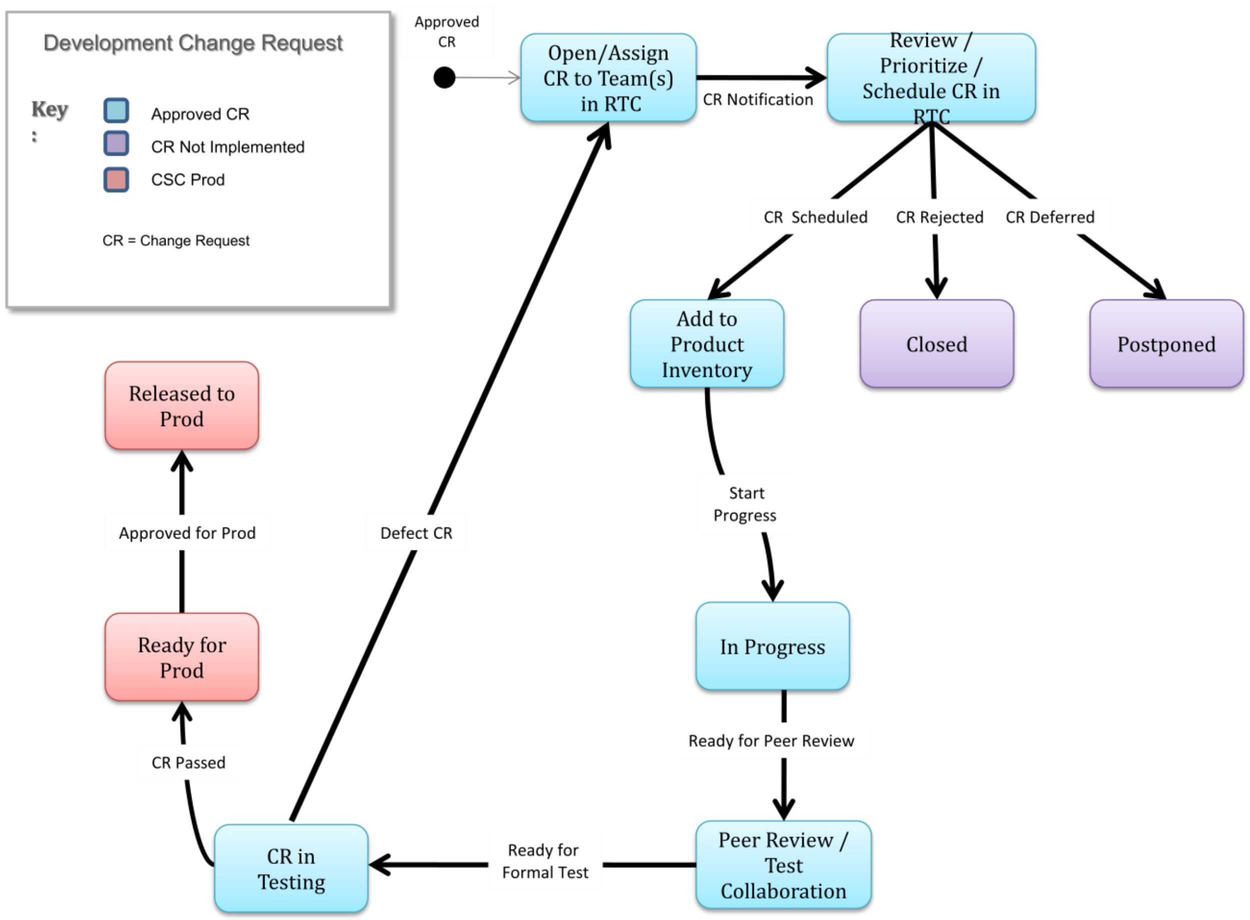


Figure 5: CM Request Workflow

5 CONFIGURATION STATUS ACCOUNTING

Configuration status accounting (CSA) is the recording and reporting of information that is needed to manage a configuration effectively, including a listing of the approved program or project artifacts, the status of proposed changes, and the implementation status of approved changes. It is concerned with relating fully approved CRs, specifications and baseline documents and their associated parts. Different baselines provide the foundation for status accounting. The objectives of CSA are to provide enhanced coordination, visibility, accountability and management. CSA reports may be used to audit the baselines.

5.1 CSA Process and Records

In terms of tracking work items and the file changes made to satisfy the work item request, RTC associates the file changes with the corresponding work item and allows users to create a query that shows this association. These queries are used to assess product quality and CM effectiveness at any given time in the program or product life cycle. Reporting on defects based on change requests provides useful quality indicators and alerts management and developers to critical areas of development. When tracking a build artifact back to the source that created it, RTC provides snapshots that associate a baseline to the source so that at any given time, the source that created the build artifact can be identified.

6 DATA MANAGEMENT

Refer to the NY-HX Data Management Plan on the NY-HX SharePoint site.

7 CONFIGURATION VERIFICATION AND AUDITS

Configuration verification and audits are accomplished to evaluate the conformance of software products and processes to applicable guidance, plans, and procedures. The audits also confirm that baselines contain all required artifacts and meet stated requirements. These activities determine the extent to which an item satisfies the required functional or physical characteristics.

7.1 Configuration Integrity

Every build (team and integration) creates a snapshot with a standard naming convention that matches the built artifact name (ear file). Auditors can trace build artifacts back to the snapshot creating it. The CIs can be identified and built artifacts can be easily recreated. In addition, reports can show all work items (User Stories, Defects) included in the snapshot as well as all CIs changed in the snapshot.

7.2 Configuration Auditing

The CM team schedules and conducts Physical Configuration Audits (PCAs) and Functional Configuration Audits (FCAs) prior to release in the production environment, to ensure the configured hardware, software, and data baselines are in accordance with the CM processes.

Appendix A GLOSSARY & ACRONYMS

Refer to the Master Project Management Deliverables Glossary and Acronym List posted in the Deliverables Section of the NY-HX SharePoint site.