



New York Health Benefit Exchange

Detailed Design Review Summary for 9.3.2 Test Plan October 9 & 10, 2012

<u>Item Number</u>	<u>Topic</u>
9.3.2	Test Plan

	Page 1	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan



New York State Department of Health

New York State Health Exchange (NY-HX)

CSC

Master Test Plan

Document Number: 528500-20120921000
Contract Deliverable Number: P-16
Contract Number: FAU# 1106211137
Release Number: 1.0
Date Submitted: September 18, 2012



Version History

Version Number	Implemented By	Revision Date	Approved By	Approval Date	Description of Change
1.0	Parampreet Sidana	09/13/2012	S. Garner, C. Adams	09/18/2012	Initial submission for delivery
1.1	Parampreet Sidana	09/17/2012			Incorporated QA comments from C. Adams
1.2	N. Graziade	09/25/2012			Incorporated data from excel sheets



Table of Contents

1	INTRODUCTION	7
1.1	PURPOSE	7
1.2	TESTING APPROACH.....	7
2	OVERVIEW	8
2.1	PROJECT OVERVIEW	8
2.2	TESTING OVERVIEW.....	8
3	TESTING ASSUMPTIONS	9
4	TESTING CONSTRAINTS	10
5	TESTING RISKS	10
5.1	TESTING REQUIREMENTS DEFINITION.....	12
5.2	MAINTENANCE OF THIS DOCUMENT	13
6	CHANGE CONTROL PROCESS	13
7	TESTING OBJECTIVES	13
8	PRINCIPLES OF TESTING	13
9	TESTING METHODOLOGY	15
9.1	AGILE METHODOLOGY	15
9.2	AGILE SCRUM FRAMEWORK	16
9.3	NY-HX SOLUTION - AGILE APPROACH	17
9.4	THE AGILE-SCRUM TEAM.....	18
10	SCOPE OF TESTING	19
10.1	TEST PLANNING AND EXECUTION	20
10.2	TEST CORRECTION.....	22
10.3	TEST ACCEPTANCE	22
11	DEFECT TRACKING AND CORRECTION	23
11.1	MANAGING REPORTED DEFECTS	24
11.2	DEFECT LIFE CYCLE/ WORKFLOW	25
12	TESTING ROLES AND RESPONSIBILITIES	26
12.1	TESTING MANAGER	26
12.2	SENIOR TEST ENGINEER/LEAD AND TEST ENGINEERS	26
12.3	BUSINESS USERS/UAT TESTERS.....	27
12.4	DATABASE ANALYST.....	27



12.5	PERFORMANCE ENGINEER.....	28
13	TESTING ENVIRONMENT.....	28
13.1	TESTING ENVIRONMENT REQUIREMENTS.....	28
13.2	SOFTWARE.....	29
13.3	HARDWARE.....	29
13.4	OTHER RESOURCE REQUIREMENTS.....	30
13.5	TESTING TOOLS	30
14	DETAILED TEST STRATEGY	30
14.1	PURPOSE	30
14.2	OVERVIEW	30
14.3	ANALYSIS	30
14.4	PLANNING.....	31
14.5	EXECUTION AND VERIFICATION.....	33
14.6	MANAGE.....	35
15	APPENDIX A – COLLABORATION OF SDLC RATIONAL TOOLS	38
16	APPENDIX B – BUSINESS AND TECHNOLOGY REFERENCE MODEL.....	39
17	APPENDIX C – DETAILED TESTING PROCESS STEPS	40
17.1	REVIEW MASTER TEST PLAN.....	40
17.2	CREATE TEST PLAN	40
17.3	SETUP TEST ENVIRONMENT	40
17.4	EXECUTE TESTS	40
17.5	REVIEW TEST RESULTS.....	41
17.6	ACCEPTANCE SIGNOFF:.....	41
18	APPENDIX D – DOCUMENT MANAGEMENT	42
18.1	WAREHOUSING OF PROGRAM ELEMENTS.....	42
18.2	WAREHOUSING OF PROGRAM DOCUMENTATION	42
19	APPENDIX E – SAMPLES FROM AGILE TEST PLANS	43



List of Figures & Tables

Figure 1: Testing Assumptions	9
Figure 2: Testing Risks	12
Figure 3: Agile Life Cycle Phases	15
Figure 4: Agile Scrum Framework	16
Figure 5: NY-HX Agile Approach	17
Figure 6: Scrum Team Structure During Sprints	18
Figure 7: Test Planning Documents.....	21
Figure 8: Test Acceptance Matrix.....	22
Figure 9: Defect Severity Level.....	23
Figure 10: Defect Turnaround Time	24
Figure 11: Defect Life Cycle/Workflow.....	25
Figure 12: Test Environment Software Requirements.....	29
Figure 13: Test Environment Hardware Requirements	29
Figure 14: Test Plan Release.....	36
Figure 15: Test Work Flow.....	37
Figure 16: Rational Tools Collaboration	38
Figure 17: Technology Reference Model	39
Figure 18: Agile Test Plan Template	43
Figure 19: PM Agile Sprint 6 Test (sample 1).....	44
Figure 20: PM Agile Sprint 6 Test (sample 2).....	45
Figure 21: PM Agile Sprint 6 (test against IE)	46



1 INTRODUCTION

1.1 Purpose

The Patient Protection and Affordable Care Act of 2010 (hereafter simply the “Affordable Care Act”) provides each state with the option to set up a state-operated health benefit Exchange, or to have a federally operated Exchange that services one or more states. New York state is planning a state-operated Health Exchange (NY-HX). The goal of NY-HX is to have an organized marketplace to help consumers and small businesses buy health insurance and, where appropriate, apply for insurance affordability programs and/or other public benefits such as Medicaid, CHIP, and other locally offered health programs. Further, New York State wants to support consumers making choices that are right for them by providing an on-line system that allows for understanding their eligibility and enrollment options, including easy comparison of available individual and small group QHP options based on price, benefits and services, and quality. Consumers seeking health care coverage will be able to go to the Health Exchanges to obtain comprehensive information on coverage options currently available and make informed health insurance choices.

The purpose of this document is to define the testing strategy and approach to testing throughout the development phases, and subsequent software releases for the New York State Health Insurance/Benefit Exchange (NY-HX) Program. This plan also defines the work products, testing procedures, roles and responsibilities, testing environments, as well as identifies assumptions, risks, and constraints for the testing effort. The Master Test Plan document is intended to be utilized by program personnel for understanding and carrying out all test activities, evaluating the quality of test activities and artifacts, and managing those activities through successful completion.

1.2 Testing Approach

This plan will focus on an Agile-based testing methodology that will be adopted during the project execution. The testing approach described herein builds upon CSC proven testing methodologies and functions in harmony with the agile approach. The testing approach also incorporates industry best practices when a definitive added value is identified to the overall strategy. Core values in the testing approach are as follows:

- Testing early and often in an iterative manner that continuously incorporates business requirements.
- Focusing on a proactively approach of building quality software up front rather than reactively waiting to respond to defects; thereby ensuring that quality is incorporated throughout the process.
- Utilizing thoughtful strategy rather than relying only on testing tools. Tools by themselves do not ensure success. Emphasizing strategic aspects of testing ensures that analysis is consistently part of our testing approach.
- Adapting the standard tools for the organization; thereby reducing risk and increasing efficiency.
- Adapting proven testing practices and methodologies that are technology-agnostic and can be utilized with a broad array of testing tools.

	Page 7	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan



2 OVERVIEW

2.1 Project Overview

The U.S. Department of Health and Human Services (HHS) requires each state to have a Health Benefit Exchange (HBE) in place by 2014. The major goal is to cover uninsured consumers, thus helping to create a healthier community and reducing overall healthcare costs. New York State (NYS), is an Early Innovator (EI) and national leader in developing an HBE serving as the key enabler for healthcare reform in United States.

Each health exchange must determine eligibility for subsidies and programs, educate consumers, provide an online storefront to evaluate, compare, and select health coverage, facilitate the enrollment process, and provide assistance for purchasers to use in evaluating options and choosing coverage. States must ensure that they are able to attract the participation of health plans to be competitive with the commercial market.

Meeting the requirement for a “no wrong door” system is critical to the success of the programs. The interface with consumers must be intuitive, well-designed, and comprehensive. An effective interface will attract consumers to the NY-HX Solution and make it easy for them to use it effectively.

The New York State Department of Health (NYS DOH) is focusing on this new venture to meet the requirements of the Affordable Care Act (ACA). Moreover, as an early innovator, NYS DOH is likely to develop an HBE that will serve as a model for other states and can interoperate smoothly and efficiently with CMS, healthcare providers, and the HBEs developed by other states. The solution should be flexible and scalable, enabling it to meet changing needs.

2.2 Testing Overview

The NY-HX program is a web-based application that supports the design, development and deployment of the front-end architecture (application, hardware, software and infrastructure) for users’ web enabling with the NY-HX solution. The architecture will include the following technical sub-systems: hCentive product, UX Enroll 2014 user Interface, the SOAP UI web-services testing tool, I-log business rules engine, DB2 LUW, Java, and UX Enroll 2014 wireframes, Java Server Pages and Magnolia content management server. Also included are the administrative functions, such as – Rational Quality Manager, Rational Requirement Composer and Rational Team Concert.

Testing for the project will follow an Agile-based methodology, where progress will occur in an iterative and incremental manner and be divided in to two-week cycles called sprints. In general within each sprint, smoke, functional (includes web-services testing, database validation,) system-integration, regression testing, retesting (general across all testing), and in some cases, User Acceptance Testing (UAT) may occur for functionality developed either during that two-week cycle, or developed during previous sprints. An initial deployment (Pilot) will be conducted in a staged manner for each release; in order to ensure the ease of identifying sources of defect. Note that the terms defect, problem or bug, have the same meaning in this document.

	Page 8	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan

3 TESTING ASSUMPTIONS

The Master Test Plan and Test Strategy is based on the following assumptions. The potential impact, if the assumption does not hold true, is also listed. This table of assumptions will be updated as additional testing assumptions are identified. The below mentioned list is not only limited to these points only:

Assumptions	Impact
All Testing environments will be established and available when needed to support the test phases and test types described in this document. This includes facilities, equipment, connectivity, set-up and ongoing support.	Scheduled testing may not be slipped and testing may not be completed. Incomplete testing could lead to production defects. Schedule delays could occur. Testing quality could be diminished.
Business users from DOH will be available to perform activities related to System-Integration, User Acceptance. And Initial Deployment (Pilot) including Test Planning and Test Execution.	Testing activities will not be completed as planned. Schedule delays could occur.
Resources from the project teams will be allocated to test efforts as outlined in this document. Testing tasks will be given a high priority and will supersede other activities during key test phases as defined in the project plan.	Planned Testing tasks activities will not be completed as per calendar. Testing quality will suffer. Schedule delays could occur.
All issues and defects identified during the testing process will be addressed in a timely manner during sprints by the assigned team member.	Testing could remain Incomplete and the end of testing phases and will lead to production defects. Schedule delays could occur. Quality could be diminished.
Scope will be effectively controlled during functional, system-integration, UAT and Initial Deployment (Pilot).	Not adhering to defined project change control procedures will allow “scope creep” to occur.
Web-enabled unit and string test cases for system-integration testing will be utilized for positive flows	If unit or string test cases will not be used, it may cause on missing some positive scenarios during testing, which could cause production defects.
Testing activities will utilize new defect management, test management, configuration management, requirements management and traceability.	Insufficient requirements tracking, development builds tracking, test execution tracking, defect tracking will impact quality of work products.
Support for testing tools (Rational Quality Manager, Rational Team Concert, Rational Requirement Composer) will be made available	Lack of technical support or availability of resource will delay the project.
All working scrum team members would be provided Laptops, so that everyone can be mobile and can work in scrum team rooms with all the team members for the respective tracks and releases.	Lack of mobility would affect the team members in working collaboratively in scrum rooms which could lead to confusion and delay in the project.
All testing activities will take place at the East Greenbush, NY CSC facility	An additional location (identification, setup) will delay project.

Figure 1: Testing Assumptions



4 TESTING CONSTRAINTS

The Integration, Initial Deployment (Pilot) and General Availability Testing efforts will be conducted given the following constraints:

1. **Dependency on Other Teams.** Successful testing is dependent on test cases being complete and available from the various development efforts (Use cases, Unit, String and Integration testing) to drive Test Planning and Execution activities.
2. **Time.** The NY - HX Program must meet the implementation time frame. The overall Planning and Execution efforts must be efficient and effective. The Conceptual Architecture and Design work of the NY-HX Program must be completed before the planning and development date of each release.
3. **Resources.** There are limited resources on the testing teams that will need support for their testing efforts from all other teams.
4. **Scope.** The scope of effort for the overall project drives Test Planning and Execution efforts. Scope changes must be closely managed as Test Planning and Execution progresses in controlled manner.
5. **Environment.** A technology refresh may limit the depth of required modifications to the data architecture and usability.

5 TESTING RISKS

Risks associated with testing will be addressed in this section of this Test Plan. For more detail on risk within the project, please refer to the Risk Management Plan. Risks associated with the testing and planning have been identified. This list is not intended to be all-inclusive since the testing team cannot always anticipate every risk at the start. The risks that are identified here are those that are foreseen at this time. This list is the starting point for the test team's risk management activities.

Critical testing risks that must be managed, tracked, and mitigated will be formally documented and reviewed. Each risk identified will be assessed in terms of the impact it could potentially have on the success of the project and a mitigation strategy will be identified which is commensurate with this potential impact.

For detailed scope and change management process, please refer 9.4.2 - Risk Register

	Page 10	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan



Risk	Potential Effect on Project Success	Risk Mitigation Strategy
Time frame for implementation is tightly time boxed.	Insufficient Unit and String testing performed by development teams. Delays in receiving components from development teams needed for Functional and System-Integration Testing during sprints.	Use strategy to prioritize and plan project efforts. Testing Team to oversee the use of appropriate standards and processes. Publish and adhere to Unit and String Test Procedures. Close coordination between the product owner, business/policy analyst, application architect, development teams and the testing team
Scope is not effectively controlled during Integration, Initial Deployment (Pilot) and General Availability Testing.	Scope creep. Schedule delays.	Adhere to defined project change control procedures. Involve project manager, requirements team, development team, architect team and testing team in determining whether a discrepancy is a defect or a change in scope or requirement change.
Lack of technical resources and tools to maintain technical environment.	Delays/interruptions in testing	Make technical resources and tools available to testing whenever needed.
Failure to allocate sufficient resources	Tools and resources vital to the testing process and other dependant areas will not be employed	Properly plan resource requirements and gain commitment to execute desired resource plan
More defects identified during the test than expected.	Fixing and regression testing takes longer than anticipated	Try to find majority of the problems as early as possible during Testing, allowing adequate time for resolution.
Fixes are not quickly turned around with high quality	Fixing and regression testing takes longer than anticipated	Let most experienced resources to work on critical fixes. Do not cut corners in unit testing fixes to minimize impact on Testing. Allow sufficient travel budget to bring JAVA expertise on board.
Test environment does not mirror production	Test results are not representative of production results.	Always test in an environment that mirrors production, and maintain the stability of the test environment
Extensive testing effort required	Additional testing will impact resources, schedule and budget.	Need to have exclusive requirements sets for each sprint, so that focused and exclusive testing can be done to deliver quality.
Delay in code delivery	Delay in code delivery on first day of each sprint would delay testing and release	Verified code should be dropped into testing environment on first day of each sprint after unit testing is completed. Proper build plan should be published and followed.



Risk	Potential Effect on Project Success	Risk Mitigation Strategy
DOH business users and product owners non-availability	Non- availability or non allocation of business users and product owners will affect overall NY-HX programs testing tasks such as Functional, System- Integration, User acceptance and Initial Deployment (Pilot) testing.	DOH has been requested to provide experts that are dedicated to NY-HX program and available all the time throughout the project
Data test bed is not controlled or non availability of test data	Lack of data refreshes or sufficient data rollback procedures or non availability of test data will delay testing	Ensure that data refreshes are well defined and executed timely. Also, verified test data is available well before test execution
Uncertainty surrounding changing the appearance and navigation of UX Enroll 2014 design.	Users unfamiliar with expected changes may impact development, designing and testing results.	Evaluate the changes prior accepting and implementing them. Include users on user design team. Test the usability design by including users in Interaction Design Mockups.
Acceptance of the User Experience work products required.	Changes to the screens after development has started will require re-work.	Decision on UX Enroll 2014 design should be taken up front prior to development and test preparation.

Figure 2: Testing Risks

5.1 Testing Requirements Definition

Testing is defined as the verification that something will perform as expected before it is implemented in production. To ensure that testing is successful for NY-HX, we must ensure the following:

- Testing strategies and plans are developed with clarity and rigor. It is critical that testing strategies and plans are comprehensive, well documented, and clearly understood by all persons involved in the testing effort and those responsible for approval of testing activities. This measure is ensured by a collaborative, communicative approach to standardizing testing strategies.
- The testing environment(s) are stable for testing. The environment(s) must be set up in such a way that testing and re-testing can be done efficiently with little or no downtime. This measure is ensured by a rigorous examination of the testing environment to be performed frequently and thoroughly.
- The testing effort is well managed. The scope of the testing effort must be properly defined, communicated, and controlled. The activities and tasks will be tracked and managed to ensure dates are met without compromising the quality of the tests. This measure is ensured by utilizing Rational Quality Manager, and core business practices which emphasize frequent checks and balances.
- Communication relating to the testing effort will need to be correct, consistent, and timely. To be efficient, the test team will need to be well informed and well coordinated. This measure will be ensured by regular status meetings within the testing team and across tracks.
- Detailed test plans (including number of cycles, test cases and timing) for each type of test will be developed at the appropriate times for each release. This measure will be ensured by keeping the test plans in sync with the builds and testing proactively and often.



5.2 Maintenance of This Document

All subsequent updates of the Master Test plan will be made and identified by revision, and only by mutual agreement between the NY-HX Program Manager, CSC Project Management, QA Manager, Development Manager, PMO and the Testing Manager. This document will be reviewed and updated on need basis and will be maintained on SharePoint.

6 CHANGE CONTROL PROCESS

Any defects, issues or requirements that are deemed changes and result in building new functionality outside the scope of baseline specifications will be incorporated into a change management process. The CSC Scope Change Review Board (SCRB) will review requests to determine whether they constitute a scope change, and/or require further review and approval by the joint DOH and CSC Scope Control Board (SCB).and Requests will be reviewed for basic understanding, impact to the current development activities, any associated risks, and will follow the scope management process documented in the NY-HX Scope and Change Control Management Plan.

For detailed scope and change management process please refer 9.4.4 NY-HX Scope and Change Control Management Plan v1.2.

7 TESTING OBJECTIVES

The general objective for testing the NY-HX program is to validate and verify any web-based portal screens supported by existing business processes and rules. Specific testing tasks to achieve this objective are stated below:

1. Define a Testing Approach that will be used by NY-HX project teams.
2. Develop Test Plans that provide the appropriate test coverage for the NY-HX project, and ensure the release is sufficiently tested prior to production.
3. Identify the test environments that will be needed to sufficiently test the web application for the NY-HX program.
4. Prepare/identify and execute Test Cases in accordance with the approved Test Plans. Document actual test results as part of the test execution activities. Obtain approval for each test that is executed successfully.
5. Identify the migration process that will be used to move the application components from one test environment to the next.
6. Identify the process(es) that will be used to document, track, report and manage issues that are identified during the testing activities.
7. Establish a repository of NY-HX test cases to be used as a benchmark for future releases.
8. Achieve an acceptable level of risk that balances cost of testing against the cost and likelihood of potential failures.

8 PRINCIPLES OF TESTING

The high level principles that will be employed on the NY-HX project, are listed below.

- **Testing can never guarantee error-free software**

It is neither physically nor economically reasonable to develop and execute an exhaustive collection of tests for each possible input condition to a component, and an exhaustive analysis of every path through the code of each component. Rather than trying to identify all errors in an effort to produce perfect software, the emphasis should be on identifying enough defects to produce software that is “good enough.”

- **Use as good test cases as possible**

	Page 13	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan

The novice's approach is often an automatic reaction to the code, using either far too many or too few test cases. Usually these tests attempt to verify that the software does what the requirements say it should. They do not test whether it does what it should not. Cultivate a testing organization that understands what types of testing are necessary in which conditions and how to develop effective test cases/scenarios. The test strategy has been defined to find as many defects as possible with as good test cases as possible.

- **Developers must fix defects as soon as they are reported and before continuing with their current tasks**

Waiting to fix bugs allows several problems to occur. The developer has time to forget the details of the problem code, increasing the time required to correct it. A developer may introduce the same defect repeatedly into the project by reusing the same incorrect approach. As a result, the project can slip n hours (where n is impossible to predict as it represents the time required to make these corrections). But management still feels that the project is on track because the developers continue to report their modules as complete. Inevitably, this leads to surprises for the management team and client when they cannot pilot or deploy the system at the end of the scheduled development period, but must instead fix all of these accumulated defects.

- **Developers should not be the only ones to test their own work**

There are at least two reasons to employ a group of minds who are separate from the developers:

- It is psychologically difficult to proofread your own work and find all of the errors.
- They may or may not have a correct or complete understanding of the requirements for which the code is written.

So, application must be tested with the collaborative efforts of all the teams and individuals included in scrum team in agile environment.

9 TESTING METHODOLOGY

NY- HX project has chosen a vigorous SDLC method which is referred as Agile methodology. It has continuous design, development, testing and client feedback approach build the NY- HX program.

9.1 Agile Methodology

Agile Methodology is based on iterative and incremental development, where requirements and solutions evolve through collaboration between self organized cross functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change.

Most software development life cycle methodologies are either iterative or follow a sequential model (as the waterfall model does). As software development becomes more complex, these models cannot efficiently adapt to the continuous and numerous changes that occur. Agile methodology is developed to respond to the changes quickly and smoothly. Agile methodology is a collection of values, principles, and practices that incorporates iterative development, test, and feedback into a new style of development. The below picture shows the phases of Agile approach (referred from Agile Scrum approach):

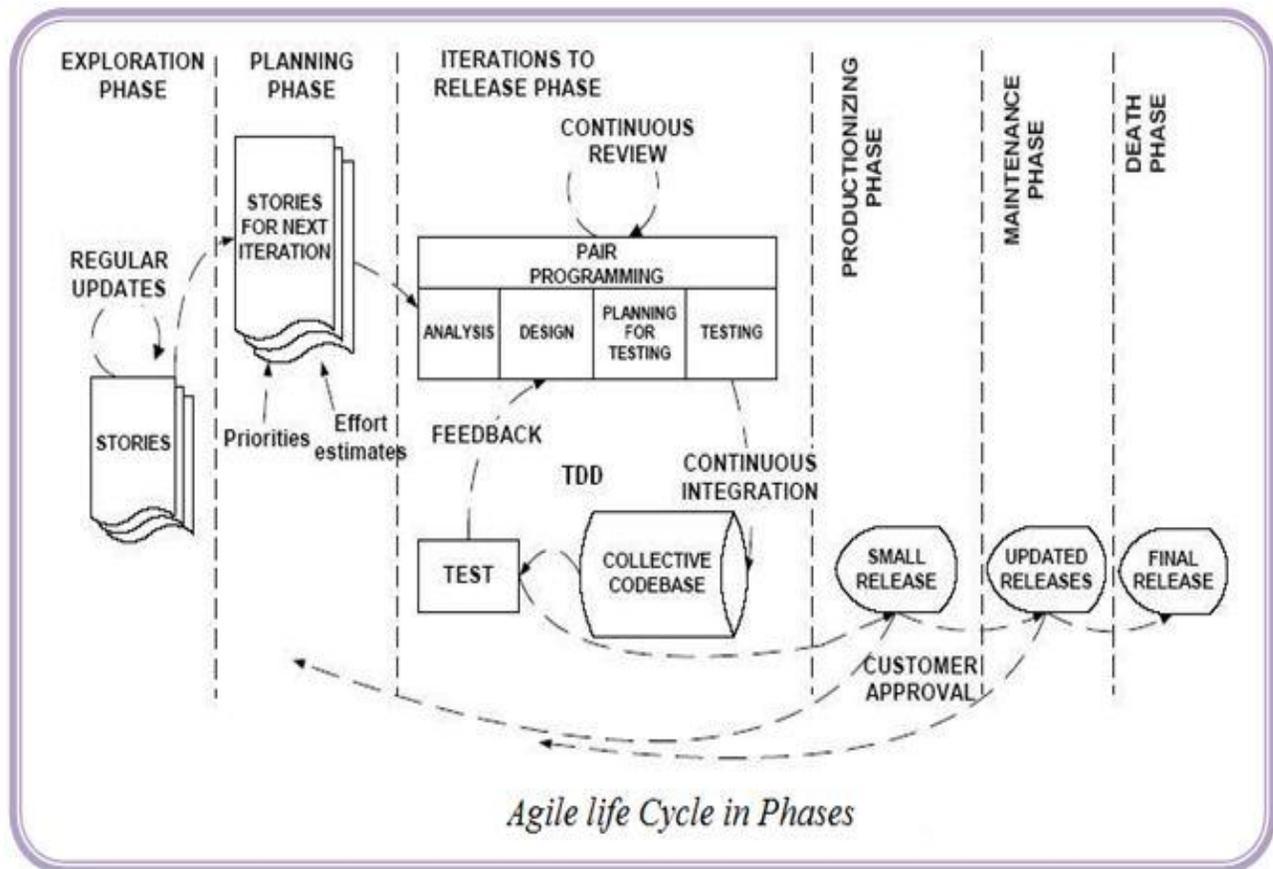


Figure 3: Agile Life Cycle Phases

	Page 15	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan

9.2 Agile Scrum Framework

Scrum is an agile framework for software development. So instead of providing complete, detailed descriptions of how everything is to be done on the project, much is left up to the software development team. This is done because the team will know best how to solve the problem they are presented. This is why, for example, a sprint planning meeting is described in terms of the desired outcome (a commitment to a set of features/functions to be developed in the next sprint) and somewhat a set of Entry criteria, Task definitions, Validation criteria, and Exit criteria (ETVX). Scrum framework is simple "inspect and adapt" framework that has majorly three roles, three ceremonies, and three artifacts designed to deliver working software in Sprints, usually 15 to 30 days iterations.

- Roles: Product Owner, Scrum Master, Scrum Team;
- Ceremonies: Sprint Planning, Sprint Review, and Daily Scrum Meeting;
- Artifacts: Product Inventory, Sprint Inventory, and Burn-up/down Chart

The below picture shows the Agile – Scrum framework at high level:

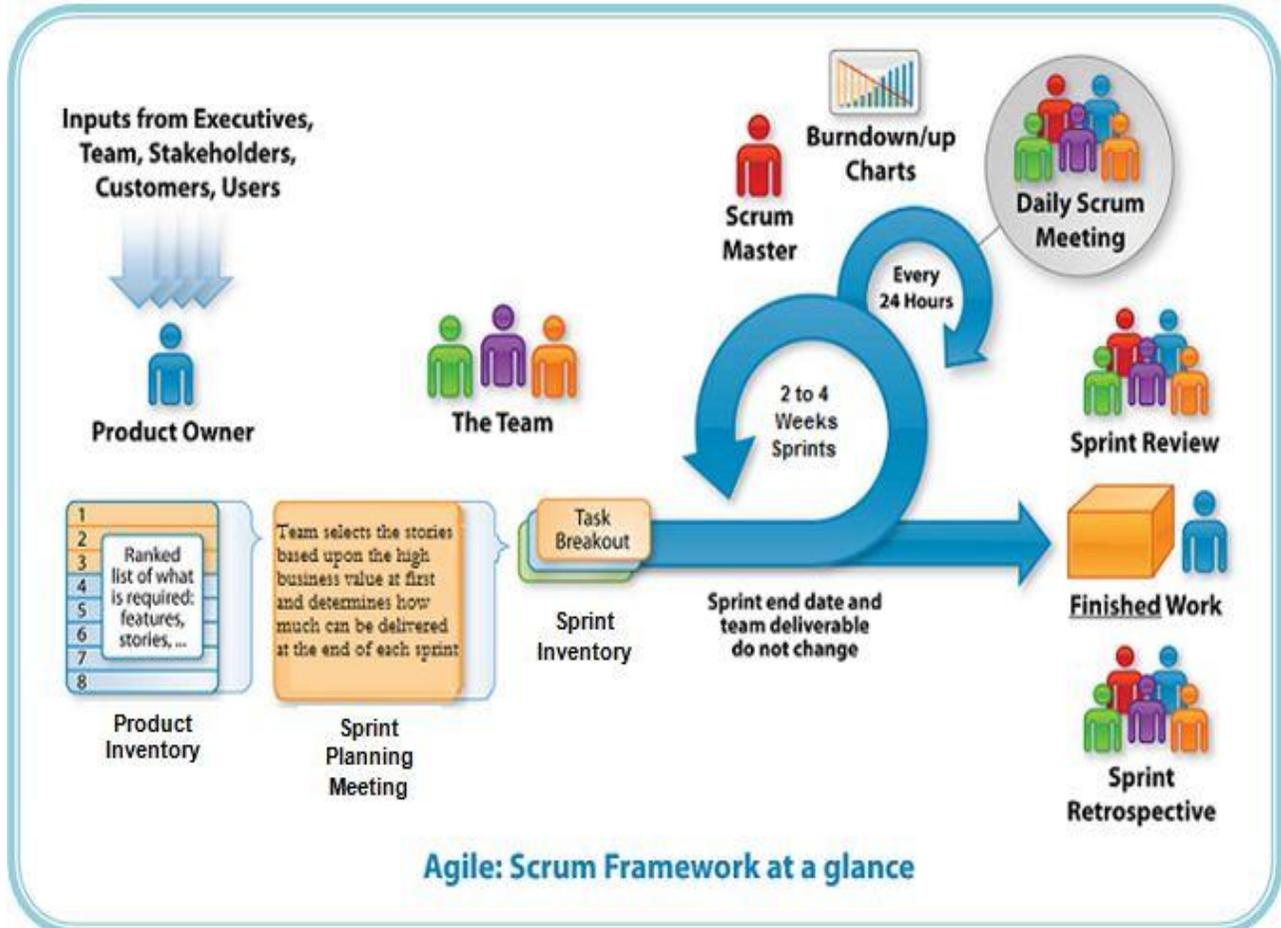


Figure 4: Agile Scrum Framework

	Page 16	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan

9.3 NY-HX Solution - Agile Approach

- **Project** - For an Agile development project to be successful, it must operate in a project context that maintains its adaptive nature. NY-HX overall program comes at project level which consists of multiple tracks, releases and sprints.
- **Track** - Agile development also focuses on establishing and maintaining the direction of Tracks of work comprised of multiple Releases. The NY-HX project Tracks are comprised of major sets of business functionality and will result in operational readiness review deliverables for each specific track. NY-HX program is initially scheduled and divided in five tracks, which are:
 - a) Eligibility & Enrollment 1 (Individual)
 - b) Eligibility & Enrollment 2 (SHOP)
 - c) Plan Management (PM)
 - d) Financial Management (FM)
 - e) Communication and Consumer Assistance (CA)
 - f) Post Prod Data Aggregation
- **Release** - The Agile development approach focuses on the execution of Releases comprised of multiple Sprints and a definitive set of functionality. Each release will result in completed requirements and design deliverables, as well as the completed test results for that particular set of functionality. Each release has 11 sprints from sprint 0 (planning), sprint 1 to 8 for development, designing and testing. And sprint 9 and 10 for testing, defect fixing and closing/reporting of release.
- **Sprint** - The core of the Agile development approach are Sprints that enable small teams to pull a batches of work from requirements through to a deployable product increment within a time box. Each sprint is defined for 2 weeks each.

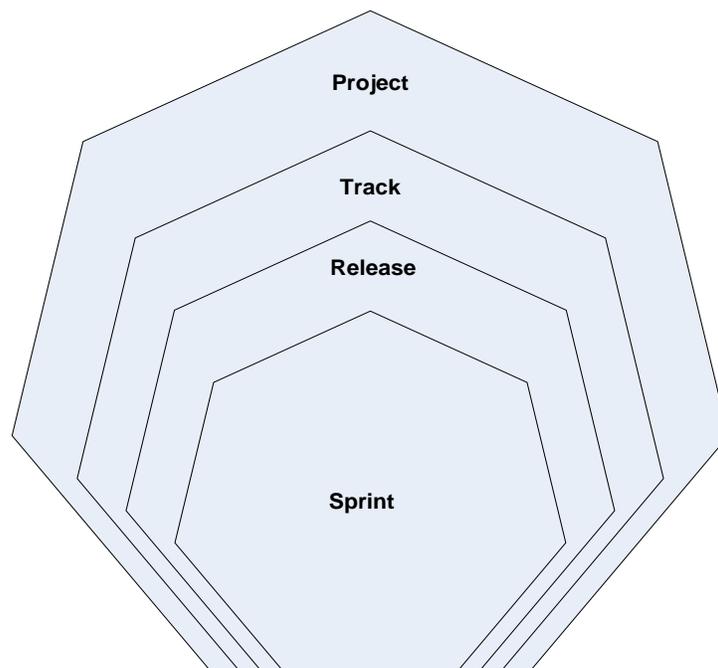


Figure 5: NY-HX Agile Approach

	Page 17	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan

9.4 The Agile-Scrum Team

Scrum relies on self-organizing, cross-functional team. The Scrum team is self-organizing in that there is no overall team leader who decides which person will do which task or how a problem will be solved. Those are issues that are decided by the team as a whole. The scrum team is cross-functional so that everyone necessary to take a feature from idea to implementation is involved.

These agile development teams are supported by two specific individuals: a Scrum Master and a Product Owner. The Scrum Master can be thought of as a coach for the team, helping team members use the Scrum framework to perform at their highest level. The product owner represents the business, customers or users and guides the team toward building the right product.

Projects using the scrum framework make progress in a series of sprints, which are time boxed iterations no more than a month long. At the start of a sprint, team members commit to delivering some number of features that were listed on the project's scrum product inventory. At the end of the sprint, these features are *done*--they are coded, tested, and integrated into the evolving product or system. At the end of the sprint a sprint-review is conducted during which the team demonstrates the new functionality to the product owner and other interested stakeholders who provide feedback that could influence the next sprint.

Scrum Teams may include: Product Owners, Business Analysts, a Scrum Master, Policy Analysts, Technical Designers, Developers, Testers, Tech Writers, Database Analysts, Technical Architects, Quality Analysts, Business Users and Subject Matter Experts.

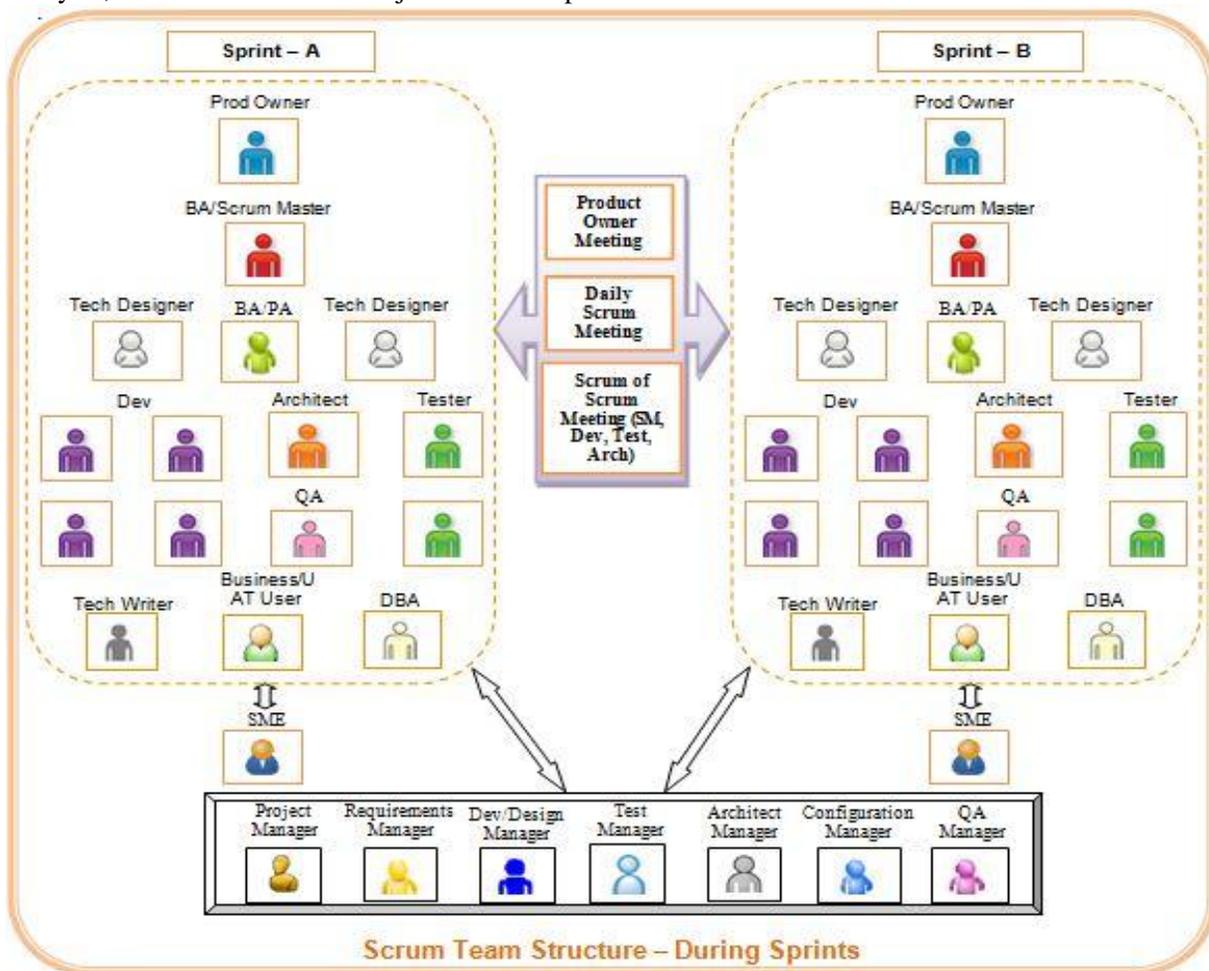


Figure 6: Scrum Team Structure During Sprints

	Page 18	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan



10 SCOPE OF TESTING

The scope of testing for each release in the NY-HX program comprises the following types of testing:

1. **Unit Testing** (as part of the development work) - tests a logical unit/component of work (LUW) such as a component, data access object (DAO) or a screen (GUI).
2. **String/Component Integration Testing** (as part of the development work) - tests the integration of logical groups of unit tests. All testing at this level is within the teams responsible for creating the units or strings.
3. **Smoke Testing** – tests the basic health check (as navigation, login, basic high level functionality, DB connectivity etc) of the builds, so that functional or system-integration testing can begin.
4. **Functional testing** – tests the individual piece of the functionality mapped to each granular level of requirements or tasks, which are defined for each user story. It includes middle layer (web services) and database testing during sprints. Web services testing is done to test the service calls, which carries the data to and fro through the databases to front end (presentation and application UI layer). Database testing includes the testing of the data structure (tables, columns-rows and fields), data validation and verification within the database.
5. **System-Integration Testing** – tests system level integrated functionality, including navigational flow, data, and business functionality by testing multiple components strung together as over all integrated system. These tests are intended to flush out errors in interfaces between components and will include limited external links. System Integration Testing also confirms that the system as a whole meets its requirements, and tests the integration of individual work products across all project teams, modules, and interfaces. It also includes the web-services testing and database testing.
6. **Regression Testing** – tests application functionality on previously tested portions of the system after changes have been made.
7. **Retesting** – tests all the bug fixes for the reported defects, and whether the fixes satisfy the requirements. Retesting occurs continuously and is an integral part component of testing across all teams.
8. **Performance Testing (Non-Functional)**– tests general throughput and response times for the application in a full-volume environment with max number of users. (performed by performance/architecture team)
9. **User Acceptance Testing** (performed by business users) – system is made generally available to a designated number of business users to verify functionality and check if all the requirements have met.
10. **Initial Deployment (Pilot)** – system users or their designated representatives confirm that the components have been built or configured according to the defined specifications. In other words, the users verify that the system can be used to help them run their business. Training activities will also begin in this phase.

A key aspect of the test plan is that each testing phase leverages the work that was performed during prior stages. This incremental approach facilitates early identification and removal of major defects while they are relatively less difficult to find and correct. It also helps to establish a progressive level of confidence in the project’s functional adequacy and stability as the various components are developed and deployed.

Another key aspect of the test plan is that it incorporates a holistic approach in order to satisfy the overall business objectives. Within each release, testing will be performed to address standalone component and interoperability tests for all screens.

The following sub-sections define the scope and types of testing that will be performed.

	Page 19	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan



10.1 Test Planning and Execution

Test Planning will be conducted during sprint 0 and prior to each sprint as well. During Test Planning, test conditions, test data, test cases and expected results will be documented. Planning for User Acceptance Testing (UAT) and Initial Deployment (Pilot) Testing also includes defining acceptance criteria, which are the rules used to determine when the work has been successfully completed.

For Test Planning, the following table documents the roles primarily involved in planning, the major activities, and the work products for each type of test.

Test Type	Responsibility	Major Activities	Work Products
Unit	Developer	Construct unit test cases Identify data requirements Conduct unit testing of module	Unit test cases with expected and actual results
String/ Component	Developer	Construct string test cases Identify data requirements Conduct string testing of modules	String test cases with expected and actual results
Smoke and Functional	Business Users(s) Testing Engineer External Entity Rep	Identify test cases Identify/validate data requirements Determine timing of test cases Execute test cases	Smoke test cases and actual results for basic health check of builds during beginning of sprints Functional test cases for each functionality/tasks derived from user stories and actual results Business User/Product owner sign off
System - Integration	Business User(s) Testing Engineer Developer/Fixer External Entity Rep	Identify test cases Identify/validate data requirements Determine timing of test cases Execute test cases	Over all system-integration test cases System-Integration test cases with expected and actual results Business User/Product owner sign off
Regression	Testing Engineer Configuration Manager	Compare results before and after a module or configuration change	Result comparisons Testing Lead/Manager Sign off
Performance/ Volume	Performance Engineer	Develop Performance/Volume Test Plan Execute Performance/Volume Test	Performance/Volume Test Plan Performance/Volume Test results Performance Mgr. Sign-Off

Test Type	Responsibility	Major Activities	Work Products
User Acceptance Testing	Business User(s)	Identify acceptance test cases	UAT results
	Testing Engineer	Identify/validate data requirements	Business User/Product owner sign off
	Developer/Fixer	Determine timing of test cases	
	External Entity Rep	Execute test cases	
Initial Deployment (Pilot)	Business User(s)	Identify test scenarios	Initial Deployment (Pilot) results
	Testing Engineer	Identify/validate data requirements	Business User/Product owner sign off
	Developer/Fixer	Determine timing of test cases	
	External Entity Rep	Execute test cases	

Figure 7: Test Planning Documents

Test cases are created for a majority of ‘logical units of work (LUW)’. LUW is a unit of work that has a distinct starting point (an event triggering the process) and a distinct end point (result from the process). In Agile process, each defined task from user stories may have n number of test cases for verification and validation. The business use cases developed during business design can be a starting point for validating existing test cases. For each case, acceptance criteria, usually in the form of expected results, are also defined. For System-Integration, User acceptance and Initial Deployment (Pilot) tests, business users should approve these expected results as noted above.

Execution, regardless of the test type, and testing proceeds according to a general process as follows (See the Appendices at the end of this document for detailed process steps):

1. Utilize Test Cases/Scripts and associated Test Plans from the business scenarios.
2. Prepare the Testing Environment.
3. Prepare/review detailed Test Data.
4. Run tests according to the test plan, creating Test Logs.
5. Analyze the test results in detail. Investigate in detail any discrepancies noted, creating and log defects with details.
6. Analyze defects and resolve, as needed, by working with the application architect, the development team, and the test team.
7. Retest, as needed, once the defect is fixed and changes are completed.
8. If defect is not fixed, defect is reopened and assigned back to relative team member
9. Perform regression testing.
10. Obtain sign-off (where applicable).

10.2 Test Correction

All defects (including modules, functionality, technical components etc.) encountered during the Functional, System-Integration and Initial Deployment (Pilot) testing phases and will go through the defined Defect Tracking and Correction procedures. Any defects whose severity impacts the ability to continue testing will be raised to the scrum team/project management team and addressed as a critical issue. A decision will be made in a timely manner to determine the next steps so that the testing effort can move forward.

10.3 Test Acceptance

As the Test Planning and Test Execution efforts for each type of test are completed, key project personnel and/or business users will review and approve the work products. Sign-off indicates that the work products are accepted, and may include any concerns, issues, or modifications deemed necessary for acceptance. Sign-offs on test execution work products also indicate that the acceptance criteria have been satisfied.

The Test Acceptance matrix below identifies when each of the Test Planning and Test Execution work products is estimated to be approved.

Phase	Test Type	Sprint Planning Sign-off	Sprint Execution Sign-off
Unit Testing	Component Tests	No signoff necessary	No signoff necessary
String Testing	Component Assembly Tests	No signoff necessary	No signoff necessary
Functional Testing (during sprints)	Sprint Review	After Every Sprint 0 and 1	After every Sprint 2 to 10
System Integration Testing	System Integration Review	As per release (Gantt) chart	As per release (Gantt) chart
User Acceptance Testing	User Review	As per release (Gantt) chart	As per release (Gantt) chart
Regression	Confidence Review	No signoff necessary	No signoff necessary
Performance/Volume	Performance/Volume Review	As per release (Gantt) chart	As per release (Gantt) chart
Initial Deployment (Pilot)	Pilot Review	As per release (Gantt) chart	As per release (Gantt) chart

Figure 8: Test Acceptance Matrix

The System-Integration, Performance/Volume, User Acceptance and Initial Deployment (Pilot) tests results require formal review.

If the results of any testing effort be deemed unacceptable, detailed documentation should be developed and presented to the Testing Lead explaining the test results and why they are unacceptable. A joint decision will be made regarding the next steps for resolving the issue.

11 DEFECT TRACKING AND CORRECTION

Identifying defects within business processes and application functionality is one of the primary reasons for Functional (during sprints), System-Integration, User Acceptance and Initial Deployment (Pilot) Testing. Defect Tracking and Correction procedures will be used to prioritize, categorize, assign, and correct defects found during testing.

In all levels of testing it is important to distinguish between defects and changes. Simply defined:

- A *defect* occurs when a project component's (module or technical infrastructure) behavior departs from that prescribed in baseline specifications.
- A *change* occurs when a project component behaves according to the baseline specification but something else is wanted. This is true even if the specification is obviously wrong.

Defects are fixed as a normal part of testing. However, changes must be submitted as a change of scope, following the project's change control procedures. If the change is approved, it will be added to the system. Differentiating between defects and changes helps prevent new functions from being introduced during testing. This helps keep testing on schedule and within budget.

For NY-HX project IBM Rational requirements management, test management and defect tracking tools will be utilized to track test case execution, defects and changes identified during the Functional, System-Integration, User Acceptance and Initial Deployment (Pilot) testing phases.

To avoid misunderstandings within the project team or between the project team and the business users, it is critical that everyone knows how to identify the severity of defects, and how severity levels affect delivery of the system. The current NY-HX project Defect Tracking system uses the following severity codes:

Defect Severity Level	Definition
1-Critical	<ul style="list-style-type: none"> • Defect that results in a system crash or critical business function failure, without an acceptable alternative workaround • Prevents the system from meeting business requirements • Affects functionality, tester workaround is not available • Testing cannot continue at pace. Very slow performance during testing
2-High	<ul style="list-style-type: none"> • Some loss in functionality in a component of the application, while application as a whole is still functional. • Prevents the system from meeting business requirements and alternative workaround is very cumbersome and/or time consuming • Has a High impact on the project • Needs to be corrected for sign-off • Jeopardizes data integrity
3-Medium	<ul style="list-style-type: none"> • Impact can be corrected at a routine schedule release • Contained within the user interface • Affects functionality, however a tester workaround is available • Problem or error has an operational impact but a workaround exists and testing can proceed
4-Low	<ul style="list-style-type: none"> • Indicates that the problem is a low priority and can be fixed at any time • Does not affect system performance • Does not affect functional or non functional requirements • Usually a cosmetic issue or has a minor operational impact • Testing can proceed without interruption

Figure 9: Defect Severity Level

These codes are aligned with the best practices that use the following definitions for each of the severity levels:

- **Critical (1)** – When software results in a system crash, database crash or critical business function failed and there is no acceptable workaround available for the problem. Critical defects are blockers and must be fixed / corrected immediately, so that testing can be resumed.
- **High (2)** - When software logic or layout does not meet specifications (for example, it does not contain required fields) and there is very cumbersome workaround available for the problem. High defects must be corrected before the code is going to be freeze for final release.
- **Medium (3)** - When software logic does not meet specifications, but a workaround is available. For example, if you click the button, the code does not work, but if you choose the command from the menu, it works fine. This severity level may also be used for failure to comply with user interface standards or inappropriate scheduling of batch jobs. These defects do not significantly impact the execution or performance of the software and need not necessarily be corrected before the code is released to production. However, for the NY-HX project these defects should also be corrected before code is released to production.
- **Low (4) - For cosmetic details. Low-level defects do not need to be corrected before release to production.**

The UAT and Initial Deployment (Pilot) fixers will strive to fix all the possible defects (including low) before the system is moved to production. Critical and High defects will receive the highest priority. *However, the presence of low severity defects will not prevent the system from being implemented in production.*

Defect Turnaround Time should be abide by the development teams to fix the defects, so that all the possible fixes can be retested and the best quality product can be deliver within the given fixed timeframe. Defect Turnaround Time to fix the defect, is defined in below table:

Defect Severity Level	Defect Turnaround Time
1-Critical	Max 24 hours
2-High	Within the Sprint
3-Medium	Within the Release
4-Low	Within the Release/Track

Figure 10: Defect Turnaround Time

11.1 Managing Reported Defects

11.1.1 Classifying Defects

In light of the scale of the NY-HX project, it has been determined valuable to establish a scheme for classifying and organizing defects when they are reported. Defects will be classified where defects are located through the use of system geography such as batch vs. online, subsystem name, window name, or interface name.

11.1.2 Recording Defects

When defects are found, it is important to record them in Defect tracking tool, so that they can be tracked, retested and closed accordingly. This record may include the following items depending upon defect tracking tool's configuration as: defect ID number, a descriptive headline (summary), the release/version/build number of the application when the defect is found, date found, severity level, tester's name, affected components, classification data, and a description of the defect. Supporting evidence of the defect may be in the form of a screen snapshot or a copy of an event log.

11.1.3 Resolving Defects

Once reported, defects must be assigned to a developer, tracked, fixed, and finally promoted to a clean build. The development, architecture, design and database teams must follow defect turnaround time protocols, so that fixes can be retested as early as possible. Whenever possible, the original developer of the code should make all corrections. As corrections are made, the release, version, build and branch containing the corrected code, the code module version, and resolution must all be recorded. A record of defect corrections will be kept to provide a solution set for future fixes. Successful fixes will be re-implemented.

11.1.4 Tracking Software Quality

The Test Team will report on progress on a daily/weekly basis. Reporting will include:

- Defect counts – by priority, subsystem, test type
- Disapproved, Completed (Pass/Fail) Test Cases/Problem Logs - by test type, priority
- Testing progress - # test cases identified, created, executed
- Additional metrics as requested

11.2 Defect Life Cycle/ Workflow

Defects will be logged during verification and validation of the test cases through Rational Quality Manager (RQM); and defects are stored as work items in Rational Team Concert (RTC) tool. The customizations of the statuses, fields and so on are performed in RTC. To manage the defects, the defect life cycle will be followed according to a defined collaboration within the RQM and RTC tools. The defect life cycle allows and enables test management to follow the defects/issues until their closure. The value of handling defects according to the defect life cycle will be emphasized within and across all teams. They will be providing or selecting appropriate values such as summary, description, steps to reproduce, attachments, severity, priority, dates to be fixed / delivered and so on. The defect work flow will also help team members, Scrum Masters Product Owners, and project management to handle and control the quality of the NY-HX project.

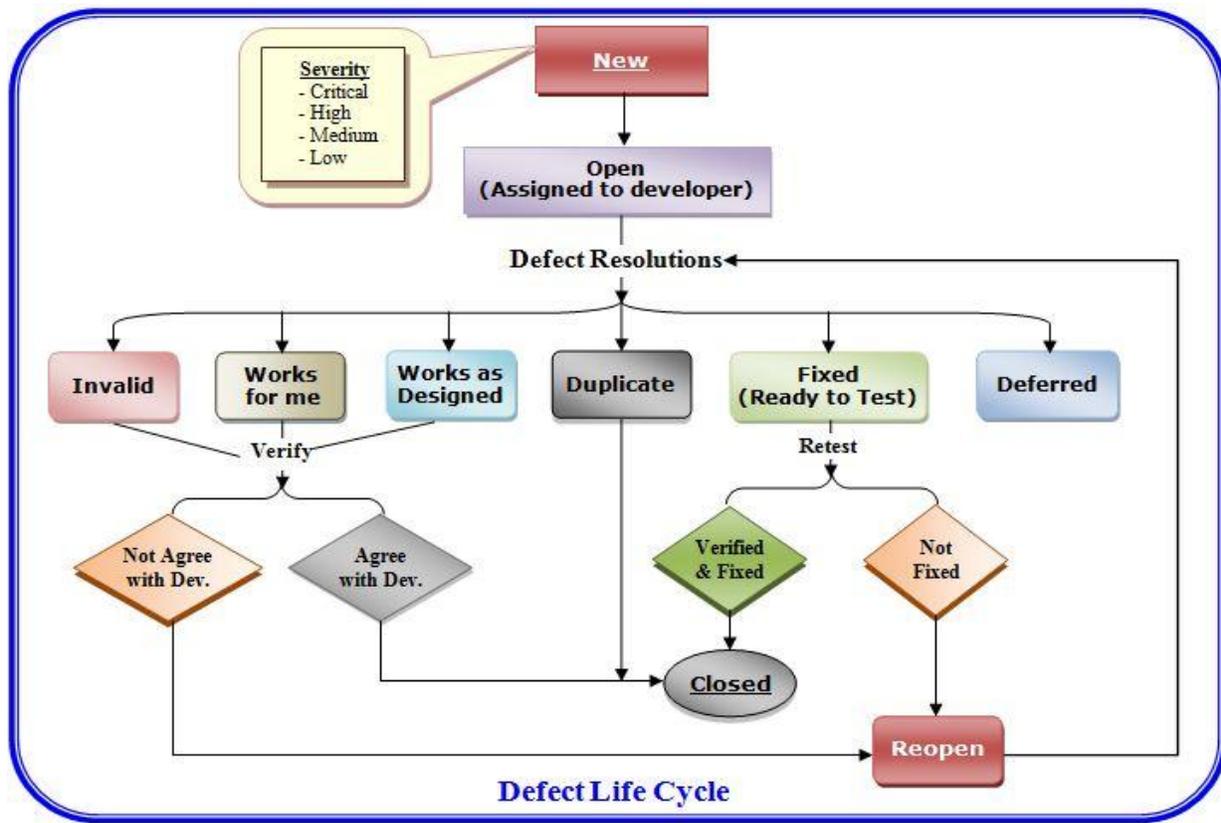


Figure 11: Defect Life Cycle/Workflow

	Page 25	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan



12 TESTING ROLES AND RESPONSIBILITIES

The following is a breakdown of responsibilities by role for the testing activities associated with the NY-HX program. More than one person may fill a role or a person may fulfill more than one role.

12.1 Testing Manager

The Testing Manager will provide overall coordination for the testing activities for a release during; the unit/string testing being conducted by the dev teams during the development phase and functional, system-integration conducted by testing team and UAT testing conducted by business user. Responsibilities include:

- Providing overall leadership and direction for testing efforts across the entire *program*
- Developing an overall master test plan, test strategy and test approach
- Providing standard templates for work products
- Identifying and notifying appropriate project management of test team requirements for staffing, space, hardware, supplies, etc.
- Directing and maintaining testing project plans and oversee testing effort
- Managing a tightly controlled testing environment by working with the Technical Support Team
- Monitoring testing activities and compiling testing statistics and work on Issue resolution
- Managing testing discrepancies and making sure they come to closure in a timely manner
- Escalating high priority defects, issues, or changes, bringing them to closure on a priority basis
- Keeping project management informed of testing status on a regular basis
- Monitoring and providing quality assurance reviews for all testing work products
- Ensuring the formal defined QA/testing standards and processes are being followed
- Guiding testing team members in logging issues and following Defect life cycle
- Meeting with all QA/testing teams after their individual scrum meetings and understand the issues/blocker and work on resolution and JAD sessions
- Documenting and monitoring test issues and track to closure
- Reporting issues and blocker items to appropriate teams and managers
- Coordinating test planning, execution activities, and results reviews for the various test types (e.g. Functional, System-Integration, Initial Deployment (Pilot), and UAT)

12.2 Senior Test Engineer/Lead and Test Engineers

A Sr. Test Engineer/Lead and Test Engineer is responsible for the development and documentation of test cases/scripts/procedure-steps and scenarios, execution of tests and logging of results to ensure all the acceptance criteria is satisfied. The Engineer will utilize automated test tools (if available) and appropriate manual templates, processes and procedures. Specific responsibilities include:

- Working with the Sr. Testing Engineer/Lead to coordinate test planning and execution activities for all test types and Ensure testing is conducted per the test plan
- Assisting the Sr. Testing Engineer/Lead in defining and managing testing environments and working with technical support personnel responsible for environment maintenance and support
- Identifying testable, non testable, front end, middle layer, database requirements for testing
- Identify requirements and work closely with Business Analyst and development engineers to understand the requirements, behavior of the system and accordingly write the test plan, test cases to complete test coverage
- Determine and acquire needed test data, test environment and support
- Create RTM and ensure and update the Test tracking tool (RQM and Excel)
- Recognize Issues/Problems/defects ahead of time and Issues tracking tool (RQM and Excel) and escalating high priority defects or changes
- Develop test results, defect results and test summary reports

	Page 26	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan

- Perform different types of testing like smoke, functional, system- integration, Regression, defects retesting and support business users during user acceptance testing
- Identifying ,addressing and assigning severity and priority for defects and changes
- Reviewing testing results and work products for completion and Obtaining final sign-off for tests as required
- Identifying and resolving priority conflicts between pre-implementation tasks and testing tasks
- Act as the “go to” resource for manual and automated scenario and case/script development and in the use of the automated tools for key users and business resources involved in test planning and execution
- Designing and maintaining the technical and functional structure of the automated testing tools (if applicable) and Maintaining library of scenarios and cases/scripts (manual or automated)
- Executing tests and compiling test results and statistics
- Communicating testing discrepancies and facilitating closure in a timely manner
- Conducting and attending life-cycle peer reviews, Test Readiness Reviews (TRR) and daily Scrum meetings and JAD sessions

12.3 Business Users/UAT Testers

The Business Users are business’ Subject Matter Experts (SME’s) from the user community. These individuals are responsible for System-Integration, User Acceptance and Initial Deployment (Pilot) test planning and execution of tests. It is anticipated that for some business users/UAT testers, their role during test execution is at times, similar to that of the Sr. Testing Engineer. Some testing responsibilities include but not limited to:

- Performing assigned test planning and execution activities
- Creating business test cases/scenarios and defining acceptance criteria with expected results
- Developing (ad hoc) test plans and work products
- Ensuring the test plans include the appropriate types of tests and level of detail
- Executing business test cases to meet the defined acceptance criteria and expected results
- Executing (ad hoc) test scenarios and cases/scripts as planned
- Identifying and logging defects, issues, changes in scope/design, and enhancements
- Facilitating the resolution of defects, issues, changes, and other discrepancies, as assigned
- Participating in testing review sessions
- Approving and signing off on test cases/scenarios and testing results produced during the functional, system-integration, UAT and initial development (pilot) testing effort

12.4 Database Analyst

The Database Analyst supports the testing process by providing direction on data structures, relationships, data placements, system interfaces and database technology. Few testing responsibilities include:

- Assisting with test execution
- Assisting with issue resolution
- Working with the Configuration Manager on back-up and restore procedures used to support all types of testing
- Preparing the testing database(s) for test execution (overall environment and automated tools)
- Tracking data used from back-up to back-up
- Coordinating back-ups and restores with the Technical Infrastructure Team
- Developing queries needed for test execution validation and verification

	Page 27	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan

12.5 Performance Engineer

The Performance Engineer is responsible for ensuring the web-site performs under load and assists with technical tests that verify the various technical components. Specific testing responsibilities include:

- Performing load tests as necessary
- Performing network connectivity tests
- Performing crash testing activities
- Assisting in development and testing activities as required
- Assisting with software deployment as necessary

13 TESTING ENVIRONMENT

13.1 Testing Environment Requirements

The environments used to support Unit, String, Integration, Performance/Volume and Initial Deployment (Pilot) testing activities are described below. The environments should be available and verified prior to testing the start of testing activities.

- **Development** – Used for Unit and String Testing work performed by the Development Teams. This will consist of physical environment and data environments for unit and string testing. This may be done at local machines of development teams.
- **Functional Testing** – Used for functional testing during sprints. The testing team uses this environment to perform functional testing to verify defined tasks from user stories during sprints.
- **System-Integration Testing** – Used for System-Integration testing. The test team to perform system-integration testing to assess potential functionality or system defects against a full size database will use this environment. This environment will have the continuous Integration of the builds which are tested, verified and passed during functional testing.
- **Performance Testing** – Used for Performance/Stress/Volume Testing. Performance team will be performing performance/stress testing to uncover potential performance issues for whole application with a certain number of users and data. It is expected that this environment will have adequate space to support full volume databases.
- **User Acceptance Testing** – Used for business users’ acceptance testing. This environment will be used for acceptance criteria validation and final approvals and sign off for pilot testing.
- **Pilot Testing (Staging)** – Used for the Initial Deployment (Pilot) testing. This environment will also be used for user training, final acceptance and sign off for production.
- **Production** - This environment will be the production environment, which will be used by live users for live transactions. Also, production support will use it verify productions issues and apply fixes for them.

13.2 Software

The following table outlines the software requirements for the NY-HX program in the test environment:

Desktop/Laptop	Desktop/Laptop Requirement Detail
Operating System	Windows 7
Applications	MS SharePoint
	Rational Requirements Composer
	Rational Team Concert
	Rational Quality Manager
	MS Office (Word, Excel, Visio, PowerPoint)
	IE 8.0 or 9.0
	Shared Drive mapping
	Lotus Notes
	SOAP UI
	DB2/LUW database access
	DB Visualizer tool to query database
	Active Risk Manager
	Oracle BPA
	Magnolia
	Activity Workflow
	iLog Business Rules Engine
	FileNet

Figure 12: Test Environment Software Requirements

13.3 Hardware

The following table outlines the minimum Laptop/Desktop hardware specifications required operating the NY-HX program in the test environment:

Laptop/Desktop Requirement	Minimum Laptop/Desktop Requirement Detail
Processor	Intel i5 Core processor
Memory	4 GB
Hard Drive	40 GB
Network Interface Card (NIC)	100 MBPS
Monitor	17" LCD with 1084x720 resolution and 16 M colors

Figure 13: Test Environment Hardware Requirements



13.4 Other Resource Requirements

Datasets connection strings for each environment will be pre-determined and loaded and fresh 'n' clean data schema is created before each test phase begins. Data backup and refresh procedures will be executed according to individual test plans.

13.5 Testing Tools

To assist in Functional, System-Integration Testing and User Acceptance testing; developers will follow standard coding techniques and tools in creating unit and string test cases. Developers and Test Engineers will also utilize appropriate tools where applicable.

The Test Strategy best practice is to utilize automated testing tools where appropriate. The following tools are available for testing.

- JUnit™ for unit and string testing
- Rational Quality Manager for functional, system, SIT and Release Integration testing and defect management
- Rational Functional Tester for automated regression testing
- Rational Requirement composer for maintaining and uploading requirements
- Rational Team Concert for configuration managements (code and builds and defect tracking)
- I-Log as business rules engine
- Content management server for static and dynamic contents
- Soap UI for (middle layer) web services testing
- MS SharePoint for all deliverables documents; common collaboration site
- DB2/LUW for (data layer) database testing

14 DETAILED TEST STRATEGY

14.1 Purpose

The purpose of this detail test strategy is to document the standards that will be used for analyzing, planning, executing, verifying, and managing the Functional, System-Integration, User Acceptance and Initial Deployment (Pilot) testing.

14.2 Overview

The test strategy is divided into five phases: Analysis, Planning, Execution and Verification, and Management. Each of the phases is further divided into description, tasks, and work products. While each task documented in this appendix should be completed, the work products are "suggestions". If documents already exist with the relevant information, the existing documents can be substituted for the work products. However, the information on the existing or created documents must be complete.

14.3 Analysis

14.3.1 Description of Analysis Phase

Initial exploration will be conducted to determine what requirements gap analysis has been completed to-date and what documentation exists. If additional information/clarifications are needed, all the teams will try to get the clarifications and collect the necessary information. Next, identify and document the scope of the specific testing activities. This scope should be reviewed with the project team prior to starting the planning activity to make sure the analysis information is current and accurate. This exercise should provide final approved product inventory which includes all the user stories.

	Page 30	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan



14.3.2 Analysis Tasks

Identify and document the business requirements and/or IT functions/rules that will be verified during the test execution. This information may exist. If not, then JAD sessions, business requirements documents, technical design documents and design/architecture documents will typically be a good place to get started.

- Determine scope of testing effort. The scope should be identified, documented, approved and agreed upon by the Project team.
- Estimate the time and duration needed to complete the testing effort during the sprints. Compare this estimate to the estimate that exists in the Testing Project Plan. Work with the Testing Lead to clearly understand the work effort for the specific test.

14.3.3 Analysis Input Work Products

- Defined Functional Decompositions (User Stories and tasks breakdown)
- Updated Business Requirement Documentation (BRD)
- Updated Technical Design Documentation (TDD) {includes application architecture}
- Overall product inventory according to defined scope
- Scope for testing effort

14.4 Planning

14.4.1 Description of Planning Phase

There are three major activities in the planning process: creating the sprint test plans, preparing environment, and identifying, preparing and gathering test data. All three of these activities must be successful for the planning effort to be successful. If any of the three activities are incomplete or incorrect, the test execution will take much longer and will be more difficult to complete.

14.4.2 Planning Tasks

14.4.2.1 Create Sprint Test Plan

Creating the sprint test plan involves translating the business requirements/user stories that were identified during the "analysis" phase into a list of task breakdown and test cases that can be designed, developed, executed and verified. The test plan will identify specific test cases/scenarios that will be executed during associated cycles.

- Identify test scenarios/cases based on the selected product inventory (user stories/requirements) and business area tracks that were identified in the gap analysis phase. The test scenarios/cases should typically represent a unit of work performed by end user.
- Identify and define acceptance criteria for each user story and scenarios, so that effective test cases can be written, executed and verified.
- Generate the test plan. The test plan will include each test case that will be executed during the specific sprint cycle.
- The test plan will need to be reviewed and approved by the product owner and business users.
- Test cases can be prioritized to ensure the critical ones are tested first, if suggested by product owner/business users. Mostly test cases would be prioritized according to prioritization of user story.

	Page 31	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan

14.4.2.2 *Create Test Environment*

Preparing the environment involves the acquisition and building of the hardware and software components that make up the system to test or validate it completely. Special care must be taken to ensure the test environments are properly established.

- Identify hardware and software that is needed to build a proper test environment. The technical support team, information system support team, network support team and/or the production support team should build, set and provide the environment.
- Set up the hardware, software, and application components that supports the software. The technical support team, information system support team, network support team and/or the production support team will need to complete the setup.
- Ensure that a mechanism is in place to control versions and backup of software and its movement to different test environments. Failure to control versions and backup can result in delays and rework with the ‘correct’ version.
- Execute a preliminary test to ensure all hardware, software, and application components have been set up properly and fully functional.

14.4.2.3 *Prepare Test Data*

Preparing test data includes the identification and set up of data that will be required to support different cycles of testing. Data may be created manually, extracted, copied or generated. Careful thought must be given in data preparation. Test data preparation, validation and verification of data should be supported or generated/created/populated/extracted by database analysts, technical designers or developers and should be supplied to testing team well before test execution. If data is not properly set up, the testing will not integrate properly between processes.

- Identify the source of the data that will be used for testing. Determine if the data will be created, extracted, copied, or generated.
- Identify the volume of data that will be used for testing. The volume of test data must be good enough to be executed quickly, yet comprehensive enough to represent production data.
- Create data manually as required. Data may be created using various tools (i.e., online portion of the application, editing tools, queries, etc.).
- Extract data as needed. Data may be extracted using various tools (i.e., extraction program, utility program, purges program, etc.).
- Generate data as needed. Data may be generated with one of the various data generation tools on the market.
- Execute a preliminary test to ensure the data has been properly established.

14.4.3 *Planning Work Products*

- Test Plan
- Test Scenarios/Cases
- Test Environment
- Test Data



14.5 Execution and Verification

14.5.1 Description of Execution Phase

Execution and Verification are kind of corresponding activities. The execution process is where the planning activities are integrated into a single process. The test cases that were documented will be executed in the test environment using the data that was set up. For test cases, screen prints would be generated. If required, for the test cases, the results will need to be printed or saved to tape as proof that execution of the test sets was completed and passed.

14.5.2 Description of Verification Phase

Verification is simultaneously done while executing each test step/case. The results of the test execution will need to be reviewed to determine if the test was successful. If the test was successful, the results will be saved as "proof" and the test plan will be updated to indicate the test was successful. If the test was not successful, an issue will be generated that describes the problem and the test plan will be updated to indicate the test was not successful.

14.5.3 Execution Tasks

- Execute application test cases using the test environment. Testing team will execute each test case documented on the test plan and whenever required, scrum team members can also help in completion of test execution. When possible, utilize job-scheduling tools to reduce the potential for error and increase efficiency.
- Generate and label test case results according to sprint cycle and release wise.
- For the test cases, screen prints should be generated for all screens with input data and the screens that contain confirmation messages. If a test case can be "proved" by browsing information, the browse screen print should be generated as well.
- Print any summary reports, file prints, file compares, queries, etc. that provides the information as the test cases were successfully completed and passed. Typically, "before" and "after" documents should be created to prove changes to data. Each of the documents should be clearly labeled with the system test date and test case number.
- Neatly package the results of the test and turn the test plan along with the test results, over to the Scrum Master and Product Owner that will be verifying and approving the tests.
- Report defects as they occur during test execution. The defects will be reviewed and fixed as appropriate. Testing must be repeated until successful for all the issues reported. If the application team cannot resolve any issue, it should be elevated to the Scrum Master or Project Manager.
- Update test plan to indicate which test cases were executed successfully and which test cases were not successful. Provide a copy of the updated test plan to the Testing Lead.

14.5.4 Verification Tasks

- Compare results that were generated during the test execution with the acceptance criteria that was defined by business users.
- Document and resolve issues as they are identified during the execution and verification process. Anytime actual and expected results do not match, an issue must be generated. The scrum team will need to know when issues are identified. If the scrum team cannot resolve the issue, it should be elevated to the Scrum Master or Project Manager.

	Page 33	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan



- Issues/defects must be prioritized and assigned to relative team member to fix and resolve them. The Scrum Master will work with the development and design team to prioritize and assign issues to corresponding team members for resolution.
- Update the test plan to indicate which test cases were successful (expected results match with the actual results) and which test cases were not successful. Provide a copy of the updated test plan to the Scrum Master and Product Owner.
- After the verification is complete, organize and package the test plan and test results. The original test documentation will be provided to the Scrum Master and Product Owner. The scrum team may retain a copy if they so desire.

14.5.5 Execution and Verification Work Products

- Revised Test Plan
- Test Results (test summary report, screen prints, reports, file prints, file compares, queries, etc.)
- Issue/Defects Report

14.5.6 Entry and Exit Criteria

14.5.6.1 Entry Criteria

Testing of application is planned in multiple iterations. The entry criterion is specified below:

- Updated Unit and String test cases and unit/string Test Results (test summary report, screen prints, reports, file prints, file compares, queries, etc.).
- Unit testing Issues/Defects report and all unit testing issues/defects are closed.
- Stable test environment (web servers, application servers and database servers and instances) is ready prior to testing.
- Test case document is ready and covers all the functionalities for the particular Sprint.
- User Access rights and ID’s have been created in the test environment
- Appropriate access rights are set up in the Defect Management tool for testers/managers.
- Code delivery to QA for each sprint is accompanied by documentation indicating areas of functionality present in the application. Checklist is also expected to provide:
 - SQL queries used to retrieve data, if any.
 - Detailing of the Services to be tested and the basic infrastructure for the same.
 - Data formats expected for the inputs in the edit fields.
- Application has undergone sufficient Unit Testing to verify stability of the build.
- Smoke/Sanity testing is successful
- All Severity 1 and 2 defects from previous cycle are fixed, delivered, tested and closed

14.5.6.2 Exit Criteria

The exit criteria for product release will be when the product has met the expected functionality defined in the test requirements of this release. The following conditions should be met:

- All Severity 1 and 2 defects from previous cycle are fixed, delivered and tested.
- All defects are either closed or deferred and test cases related to deferred defects should be mapped to defect ids and remains failed
- Product Owner should agree to sign off on all open defects as tolerable for product launch.
- A test plan with execution report should be published after release and should be reviewed by concerned stakeholders
- All test cases addressing to critical and high scenarios should be passed

	Page 34	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan

14.6 Manage

14.6.1 Description of Management Phase

The analysis, planning, and execution and verification phases occur one after another. The management phase is an ongoing phase that will start with the analysis phase and will continue after the verification phase for every sprint and release.

The scrum teams will report milestones and status to the Scrum Master, Development Manager and Testing Manager. The Scrum Master, Development Manager and Testing Manager will consolidate the information and will report milestones and status report to Sr. Management, PMO and Client.

14.6.2 Management Tasks

- Identify and acquire the required resources to effectively analyze, plan, execute and verify the tasks.
- Report milestones and status to the Project Manager.
- Track and manage issues that relate to the application team.
- Assist with the risk management documentation.
- Support the testers with analysis, planning, execution, and verification efforts.
- Obtain Signoffs for completed tests.
- Package documentation at the completion of each test.

14.6.3 Management Work Products

- Release Plan
- Issues/Defects Report (If necessary)
- Status Reports
- Testing deliverable Signoffs (if process is defined, it will be delivered)

14.6.4 Test Suites/Cases Management

Test Case Management will be done at Test Plan, Test Suite, Test Case, Test Script level according to the workflow of RQM tool. There will be different types of test cases, which will be managed in Test plans and Test Suites. Testing will be behind development at least one sprint. So, during each sprint test cases will be written and executed according to selected user stories during sprint planning.

- Test Plan will be created at Release level for each track
- Test Suite will be created at Sprint level and Test Suites will be under Test Plan
- Functional Test Cases/Scripts will be created at Sprint level and test cases/scripts will be under test suite
- SIT Test Cases/Scripts will be maintained in SIT test suite under Test Plan for each release
- Regression Test cases will be maintained in Regression test suite under Test Plan for each release

Below diagram shows the folder structure which will be managed and maintained during testing:

	Page 35	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan

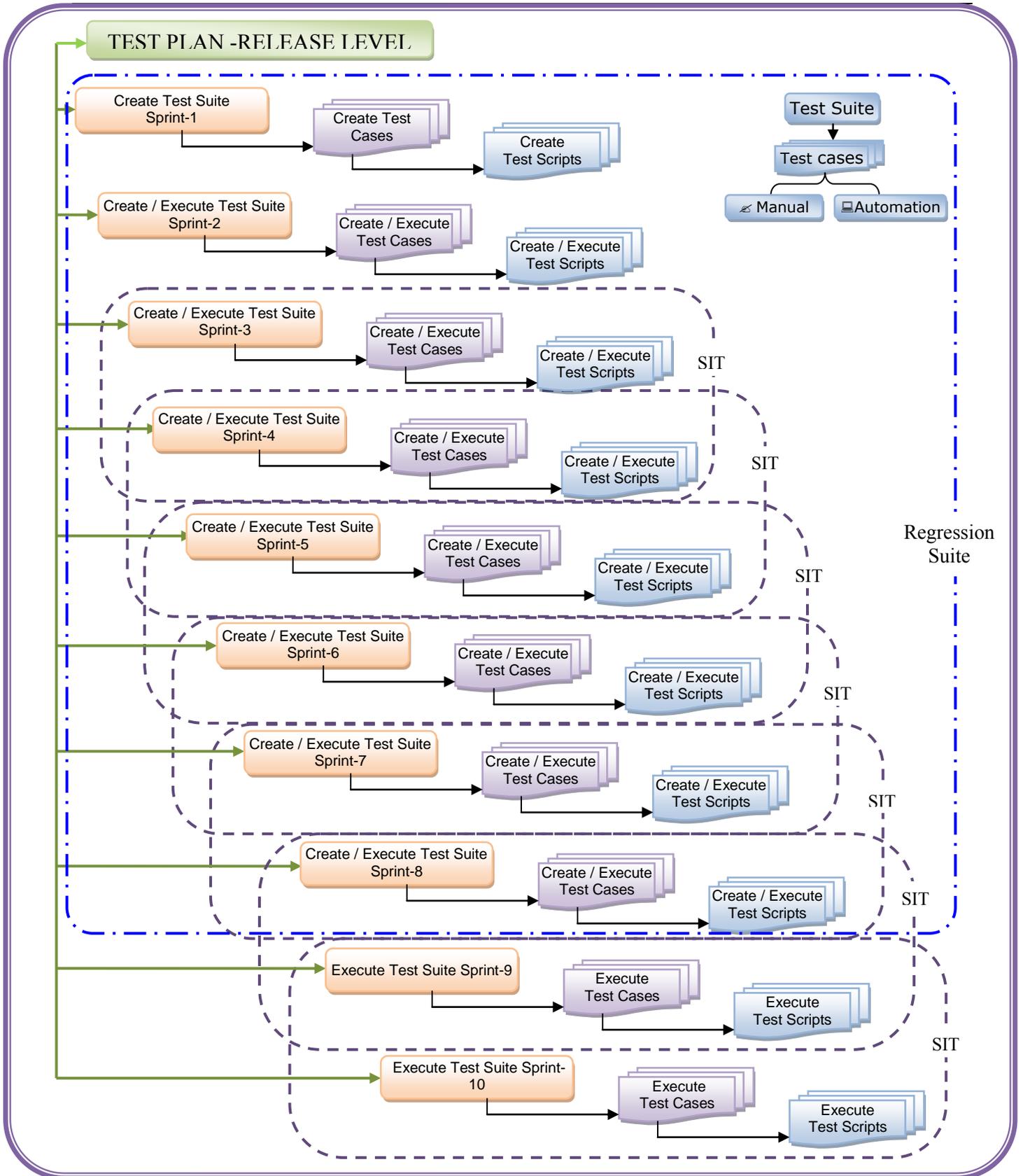


Figure 14: Test Plan Release

14.6.5 Test Work Flow

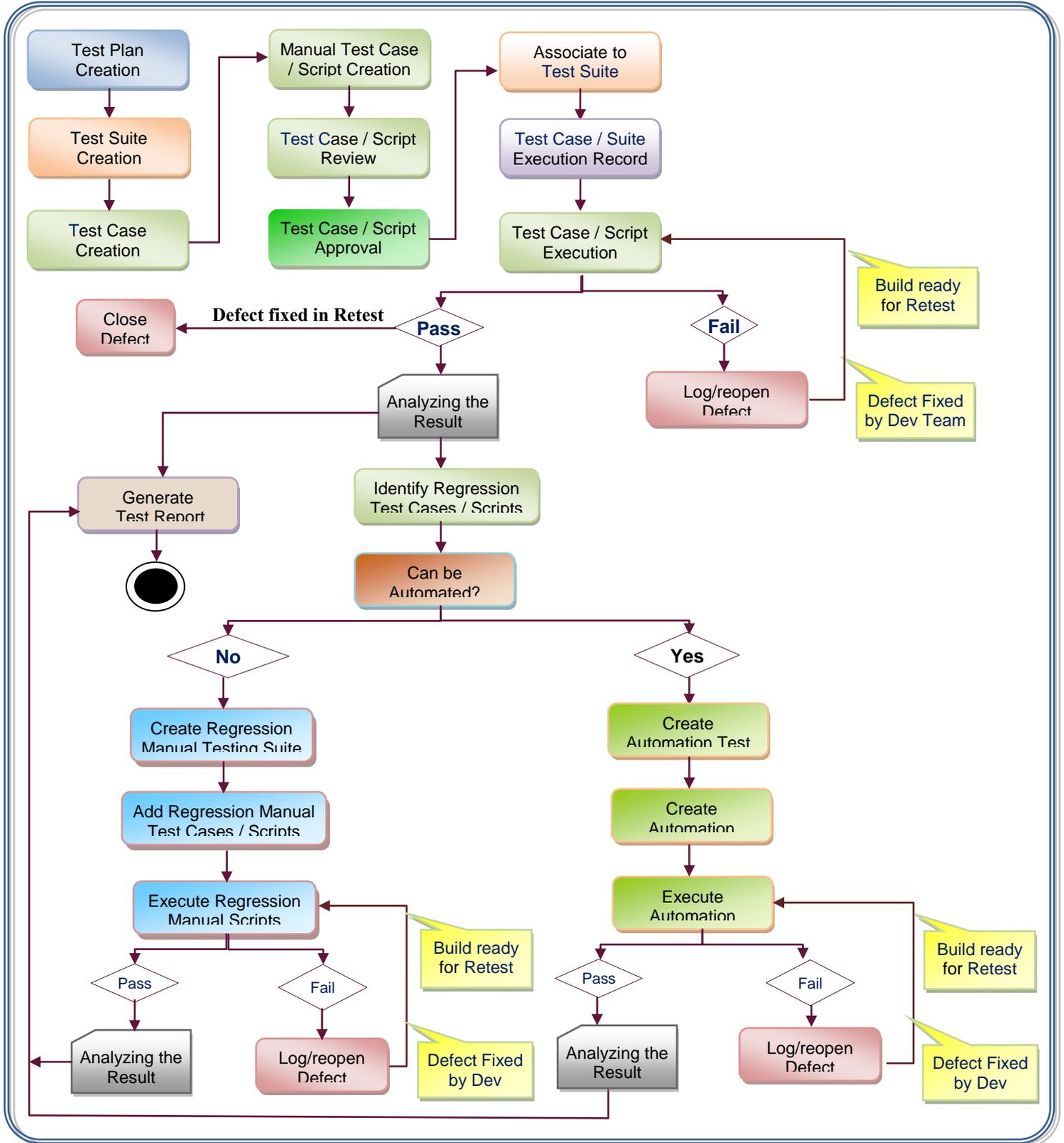


Figure 15: Test Work Flow

15 APPENDIX A – COLLABORATION OF SDLC RATIONAL TOOLS

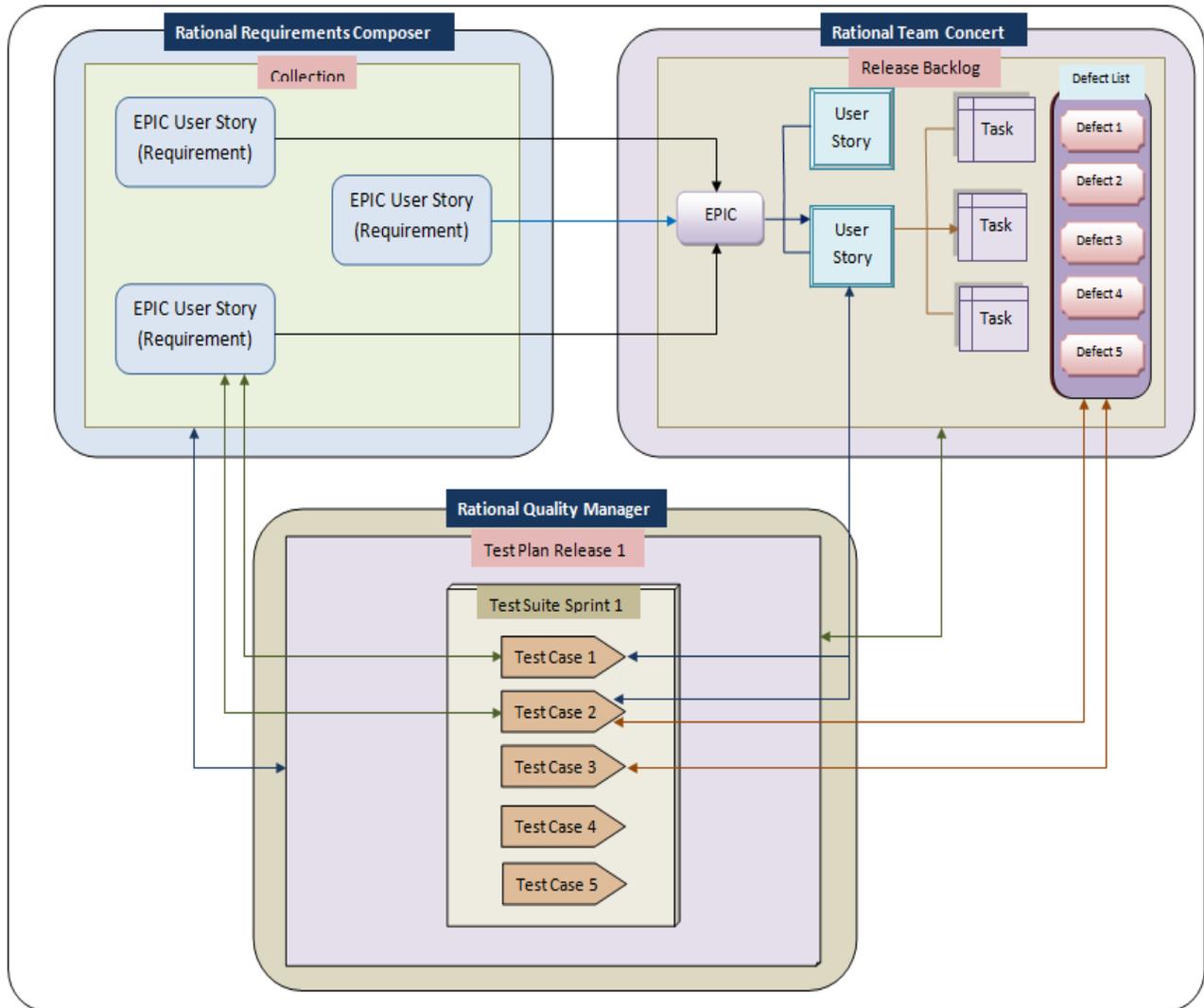


Figure 16: Rational Tools Collaboration



16 APPENDIX B – BUSINESS AND TECHNOLOGY REFERENCE MODEL

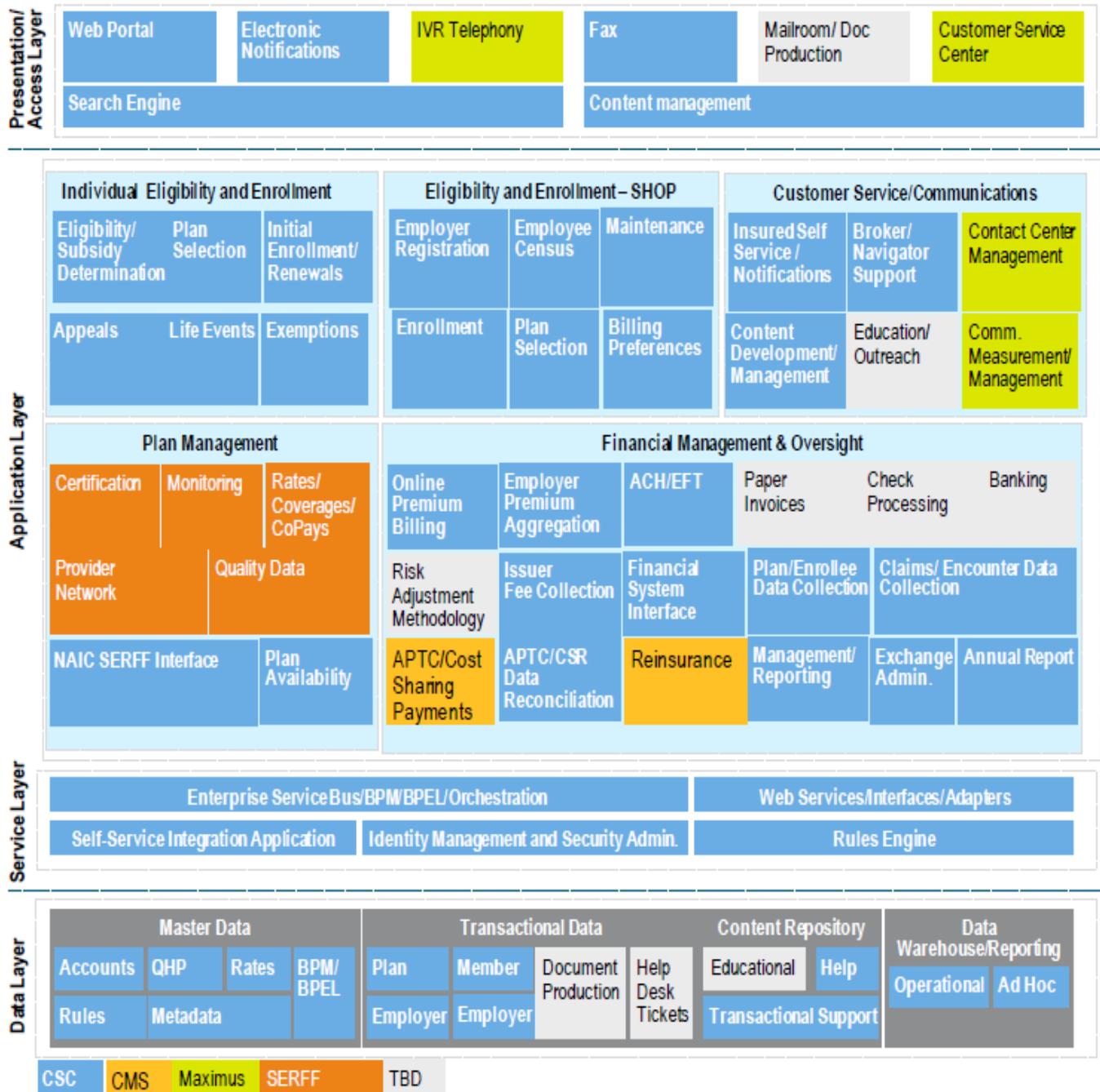


Figure 17: Technology Reference Model



17 APPENDIX C – DETAILED TESTING PROCESS STEPS

17.1 Review Master Test Plan

Actors: Project Manager, Test Manager, QA Manager, Development Manager, PMO, and Business Architect

- Create: Identify Approach, Resources, Tools (defect/issue management, testing) and Procedures
- Review: Schedule Meeting, Obtain feedback
- Finalize: Apply revisions
- Acceptance Signoff: Review and signoff

17.2 Create Test Plan

Actors: Scrum Master, Testing team, Business Users, Development team, Performance team, Configuration Manager, Technical Architect, DBA

- Create: Design Requirements, Business and Technical Inputs, Test Plan Repository, Test Cases/Scripts (Identify or Create)
- Review: Schedule Meeting, Obtain feedback
- Finalize: Apply Revisions
- Acceptance Signoff: Review and obtain signoff

17.3 Setup Test Environment

Actors: Information technology system support team, Network Support team, Development team, Technical Architect, Tech designers, Performance Manager, Configuration Manager, Project Manager, DBA, Test Manager, and Scrum Master

- Define: Identify Tools, Infrastructure Requirements, Data Requirements (All environments)
- Create: Install tools, extract/load data, initial backup
- Configure: Setup environment preferences
- Validate readiness: Execute mock test case
- Readiness Signoff: Review and signoff

17.4 Execute Tests

Actors: Scrum Team consists of Test Engineers, Sr. Test Engineers/Lead, Performance Test Engineers, Business Users, Developers, Development Lead, Tech Designers, and DBA

- Initial Environment: Backup or refresh data
- Configure: Checkout test plans, test cases/scripts
- Test: Execute steps, capture results, capture test plan changes
- Finalize: Log test results, check-in documented results

	Page 40	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan

17.5 Review Test Results

Actors: Product Owner, Business Users, Scrum Master, Project Manager, Test Manager, Test Lead, Development Manager, Development Lead, Scrum Team, Quality Assurance Manager and PMO

- Prep Work: Prepare for review, determine additional participants, individual review of daily testing results
- Schedule Meeting: Schedule time and place for daily meetings
- Review: Validate results, review issues, log issues, assess change requests to test plans, assess and prioritize change requests due to defects
- Finalize: Log results for status reporting

17.6 Acceptance Signoff:

Actors: Product Owner, Business Users, Scrum Master, Project Manager, Quality Assurance Manager

- Prep Work: Prepare for final review of test results and issue resolution, individual review of test phase results
- Schedule: Schedule time and place for meeting
- Review: Review final test status report or identify rework
- Finalize: Document acceptance or rework until acceptance is achieved



18 APPENDIX D – DOCUMENT MANAGEMENT

18.1 Warehousing of Program Elements

All development elements will follow standard configuration management procedures as documented in the NY-HX Configuration Management Plan. This includes code, on-line screens, technical components and objects, wireframes, business rules and data where appropriate. A folder structure will be created within the version control system for capturing these items for proper archival, backup, approval and signoffs.

For detailed configuration management, please refer 9.4.5 NY-HX Configuration Management Plan.

18.2 Warehousing of Program Documentation

All agreed upon work products will be properly checked in (checked out) of version control system or SharePoint for change control purposes. This includes the Functional Design, Technical Design, Test Plans and Test Cases, and Technical and Environmental Setups. A folder structure will be created within SharePoint for capturing these items for proper archival, backup, approval and sign offs.

For all the CMS deliverable documents, please visit the SharePoint link:
<https://workspace.nyhx.emedny.org/pm/design/Forms/AllItems.aspx>

	Page 42	New York State Health Insurance Exchange (NY-HX Program)
		9.3.2 Test Plan



19 APPENDIX E – SAMPLES FROM AGILE TEST PLANS

Agile Test Plan							
Track Name			Approved By	Date	Signature/Initials	Sign Off	
Release #							
Sprint #							
Resource Name	% Committed		Assumptions	Clarifications/Risks	Dependencies	Comments	Out of Scope
Test Plan Version #	Date	Reviewed by	Comments/Updates				

Figure 18: Agile Test Plan Template

Agile Test Plan			
Overview			
The test plan is designed to describe the scope of the overall test effort and provides a record of the test planning process. This test plan identifies the scope to be tested, what is to be included in the testing effort, environment used, test scenarios & test data, the break down of the test cases and identifying risk areas, assumptions and dependencies.			
In Scope	Category	User Story Name	Comments
		174: As a developer, I want to create the physical data model for NY-HX, comparing with the SERFF data dictionary	
		386: As a developer, I want to create prototype for an Exchange Regulator, so that the team can review and finalize the validation screens for an Exchange Regulator	
		387: As a developer, I want to create prototype for an Issuer, so that the team can review and finalize the validation screens for an Issuer	
		1519: Validate the data against the technical and business rules (Validation Framework)	

Figure 19: PM Agile Sprint 6 Test (sample 1)



User Story Name	Type of Test	Test Scenario	Test Data	Test Case Number	Test Complexity	To be Included in Regression	Manual or Automated	Testing Phase/Test Environment
174	Data Validation	Issuer Table with valid data elements and records		PM_RI_SP6_US174_TC001_Verify that the Issuer table exists in the database	2: Medium Complexity	No	Manual	Functional (System) Test
	Data Validation	Plan Table with valid data elements and records		PM_RI_SP6_US174_TC002_Verify that the Plan table exists in the database	2: Medium Complexity	No	Manual	Functional (System) Test
	Data Validation	Plan_Service_Area Table with valid data elements and records		PM_RI_SP6_US174_TC003_Verify that the Plan_Service_Area table exists in the database	2: Medium Complexity	No	Manual	Functional (System) Test
	Data Validation	Plan_Accreditation table with valid data elements and records		PM_RI_SP6_US174_TC004_Verify that the Plan_Accreditation table exists in the database	2: Medium Complexity	No	Manual	Functional (System) Test
	Data Validation	Plan_Premium_Data Table with valid data elements and records		PM_RI_SP6_US174_TC005_Verify that the Plan_Premium_Data table exists in the database	2: Medium Complexity	No	Manual	Functional (System) Test
	Data Validation	Formulary_Data Table with valid data elements and records		PM_RI_SP6_US174_TC006_Verify that the Formulary_Data table exists in the database	2: Medium Complexity	No	Manual	Functional (System) Test
	Data Validation	Benefit_Services Table with valid data elements and records		PM_RI_SP6_US174_TC007_Verify that the Benefit_Services table exists in the database	2: Medium Complexity	No	Manual	Functional (System) Test
	Data Validation	Benefit_Service_Cost_Sharing Table with valid data elements and records		PM_RI_SP6_US174_TC008_Verify that the Benefit_Service_Cost_Sharing table exists in the database	2: Medium Complexity	No	Manual	Functional (System) Test
	Data Validation	SBC_Scenario_Results_Data Table with valid data elements and records		PM_RI_SP6_US174_TC009_Verify that the SBC_Scenario_Results_Data table exists in the database	2: Medium Complexity	No	Manual	Functional (System) Test
	Data Validation	Service_Area Table with valid data elements and records		PM_RI_SP6_US174_TC010_Verify that the Service_Area table exists in the database	2: Medium Complexity	No	Manual	Functional (System) Test
	Data Validation	Zip_Service_Area Table with valid data elements and records		PM_RI_SP6_US174_TC011_Verify that the Zip_Service_Area table exists in the database	2: Medium Complexity	No	Manual	Functional (System) Test
	Data Validation	PM_Master_Data with valid data elements and records		PM_RI_SP6_US174_TC012_Verify that the PM_Master_Data table exists in the database	2: Medium Complexity	No	Manual	Functional (System) Test
	Data Validation	Provider_Network_Physician_Data Table with valid data elements and records		PM_RI_SP6_US174_TC013_Verify that the Provider_Network_Physician_Data table exists in the database	2: Medium Complexity	No	Manual	Functional (System) Test
	Data Validation	Provider_Network_Ancillary_Data Table with valid data elements and records		PM_RI_SP6_US174_TC014_Verify that the Ancillary_Network_Physician_Data table exists in the database	2: Medium Complexity	No	Manual	Functional (System) Test

Figure 20: PM Agile Sprint 6 Test (sample 2)

User Story Name	Test Pre Conditions	Test Scenario	Test Case Number & Name	Step number	Test Step name	Expected Results
174	The Issuer table must be present in the database	Issuer Table with valid data elements and records	PM_R1_SP2_US174_TC001_Verify that the Issuer table exists in the database	1	Login the given database with valid User ID and Password	User should be able to login the database
				2	Run the query to verify the data types and field size of Issuer table: DESC Issuer	All the column names with their data types and field size from Issuer table are displayed: Issuer_LegalName, Alpha-Numeric, X(1000) Federal_Identifier, Alpha-Numeric, X(9) NAIC_Company_Code, Alpha-Numeric, 9(5) NAIC_Group_Code, Alpha-Numeric, 9(5) HHS_Issuer_ID, Numeric, 9(5) Issuer_Holding_Company_Name, Alpha-Numeric, X(1000) Issuer_Address_1, Alpha-Numeric, X(200) Issuer_Address_2, Alpha-Numeric, X(200) Issuer_Address_3, Alpha-Numeric, X(200) Issuer_Address_City, Alpha-Numeric, X(150) Issuer_Address_State, Alpha-Numeric, X(2) Issuer_Address_Zip, Alpha-Numeric, X(9) Issuer_Id, Alpha-Numeric, X(50) Issuer_website, Alpha-Numeric, X(250) Issuer_state, Alpha-Numeric, X(2) Issuer_Third_Party_Filer, Boolean, X(1) Issuer_Phone, Numeric, X(15) Issuer_Phone_Extension, Numeric, X(5) Issuer_State_ID, Alpha-Numeric, X(50) Issuer_Consumer_Facing_Website, Alpha-Numeric, X(250) Insert_By, Alpha-Numeric, X(20) Insert_Date, Date, Update_By, Alpha-Numeric, X(20) Update_Date, Date,
				3	Run the query to verify the number of records in Issuer table equals the records obtained from Data Source: Select Count (*) from Issuer	The function must return the number of records in Issuer table
				4	Run the query to verify that primary key, "Issuer_Id" has unique values: Select Issuer_Id, Count (Issuer_Id) From Issuer Group By Issuer_Id Having Count (Issuer_Id) > 1	The query should not return any records

Figure 21: PM Agile Sprint 6 (test against IE)