Test Plan for Paychex Out of Office Application

Version 2.0

Prepared by:
Ian Dann
Daquanne Dwight
Tom Eiffert
Elysia Haight

Rochester Institute of Technology
Paychex

March 10, 2013

Revision History

Version	Revision Date	Changes Made	Justification	Author
1.0	2/14/13	Initial Revision	Creation of template	Ian Dann Daquanne Dwight Tom Eiffert Elysia Haight
1.1	2/18/13	Completed missing sections	Completed test plan document	Elysia Haight
2.0	3/10/13	Updated to correct defects from bug tracker	Some out-of-date information contained	Elysia Haight

1. Introduction

This test approach document describes the appropriate strategies, processes, workflows and methodologies used to plan, organize, execute and manage testing of the Paychex Out of Office (OOO) application.

1.1 Scope

1.1.1 In Scope

The Paychex OOO Test Plan defines the unit, integration, system, regression, and Client Acceptance testing approach. The test scope includes the following:

- Testing of all functional, application performance, security and use case requirements listed in the Requirements Document [1]
- Heuristic evaluations for user interface
- Cognitive walkthroughs of user interface

1.1.2 Out of Scope

The following are considered out of scope for Paychex OOO application Test Plan and testing scope:

- Functional requirements testing for systems outside Paychex OOO
    - LDAP
    - Email service
- Acceptance testing on devices other than those operating Android 4.0 or iOS 4.
- Acceptance testing on tablets
- Testing while accessing Paychex's LDAP server

1.2 Quality Objective

1.2.1 Primary Objective

A primary objective of testing application systems is to: *assure that the system meets the full requirements, including quality requirements (AKA: Non-functional requirements) and fit metrics for each quality requirement and satisfies the use case scenarios and maintains the quality of the product.* At the end of the project development cycle, the sponsors should find that the project has met or exceeded all of their expectations as detailed in the requirements.

Any changes, additions, or deletions to the requirements document and the use case description will be documented and tested at the highest level of quality allowed within the remaining time of the project.

1.2.2 Secondary Objective

The secondary objective of testing application systems will be to: *identify and expose all issues and associated risks, communicate all known issues to the project team, and ensure that all issues are addressed in an appropriate matter before release.* As an objective, this requires careful and methodical testing of the application to first ensure all areas of the system are scrutinized and, consequently, all issues (bugs) found are dealt with appropriately.

1.3 Roles and Responsibilities

1.3.1 Project Team Members

Software Engineering students at Rochester Institute of Technology participating in Senior Project and undertake development activities for the Paychex OOO Application. Responsible to:

(a) Develop the system/application

(b) Develop Use cases and requirements in collaboration with the Project Sponsors

(c) Conduct Unit, system, regression and integration testing

(d) Support user acceptance testing

1.3.2 Project Sponsors

Paychex employees representing Paychex as the customer. Responsible to:

(a) Contribute to use case and requirement development through review

(b) Conduct full user acceptance and end-to-end testing at completion of the product; this includes reporting results

1.4 Assumptions for Test Execution

Below are some minimum assumptions made in order to test the Paychex OOO Application:

- For User Acceptance testing, the Project Team members have completed unit, system, and integration testing and met all the Requirement's based on the Requirements Document.
- User Acceptance testing will be conducted by the Project Sponsors
- Project Team Members will support and provide appropriate guidance to the Project Sponsors to conduct testing.

1.5 Constraints for Test Execution

Below are some identified constraints for test execution:

- Project Sponsors should clearly understand on test procedures and recording a defect or enhancement.
- YouTrack is available on the team server
- Developer will support ongoing testing activities based on priorities
- Test scripts must be approved by regression testing lead
- The acceptance testing lead cannot execute developed test plan. This must be done by other team members and the project sponsors.

2. Test Methodology

2.1 Purpose

2.1.1 Overview

The purpose of the Test Plan is to achieve the following:

- Define testing strategies for testing all the of functional and quality (non-functional) requirements outlined for the product
- Define bug-tracking procedures
- Identify testing risks
- Identify required resources and related information
- Provide testing schedule

2.1.2 Build Tests

2.1.2.1 Unit Testing (Multiple)

Project Team members will develop and perform unit tests of controller and model components during the development of each component in order to ensure that each class abides by the interface and communication expectations. The unit testing procedure will follow the Test-Driven Development (TDD) model, in that unit tests will be created and run before controller and model classes are completed. Each change to a class will require running of the unit tests.

2.1.3 Milestone Tests

2.1.3.1 Regression Testing

At the completion of acceptance tests for each release, automated tests are to be developed and run at regular intervals to ensure that subsequent modifications to the system do not introduce regressions.

2.1.4 Release Tests

2.1.4.1 UsabilityTesting

To ensure the system is usable, before submission of the completed user interface (UI), cognitive walkthroughs and heuristic evaluations will be performed.

A cognitive walkthrough involves defining a set of tasks that a user should accomplish through the user interface, with simple steps to accomplish and expectations of outcome. A collection of unaffiliated people representing the end user groups are given the task and a wireframe representation of the UI, and asked to complete the task without assistance. The UI is evaluated based on the time it takes for users to complete the tasks, and how many needed assistance.

Heuristic evaluations require the team to collaborate on evaluating the designed user interface given the following heuristics:

- **Visibility of system status**:
  - The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
- **Match between system and the real world**:

    - The system should speak the user's language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- **User control and freedom**:
    - Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
- **Consistency and standards**:
    - Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- **Error prevention**:
    - Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
- **Recognition rather than recall**:
    - Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- **Flexibility and efficiency of use**:
    - Accelerators—unseen by the novice user—may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- **Aesthetic and minimalist design**:
    - Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
- **Help users recognize, diagnose, and recover from errors**:
    - Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
- **Help and documentation**:
    - Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

2.1.4.2 Acceptance Testing

Acceptable testing is to be preformed by the Team as well as the Project Sponsors based on requirements documented in the Requirements Document and use cases for each release, after the system has been demonstrated by the Project Team members. The Project Sponsors will provide feedback during the regular sponsor meetings and over email so that all test results can be documented and addressed.

2.1.5 Testing Completeness Criteria

A test is considered to have been completed successfully if the result of test meets the expected results as specified in the Requirements Document.

2.3 Bug Regression

Bug regression testing will be conducted throughout each of the four releases. When a bug is found in a release and fixed it will still be re-tested on all of the following releases. If a fixed bug becomes active again, it will be documented and set as a top priority to fix. Development team will equally split the bug regression testing across the system. All recurring bugs will be designated to the individual on the development team with the most knowledge regarding the bug.

2.4 Bug Triage

Bug Triaging will be performed weekly at each Project Team member meeting. All Project Team members should be involved in the triage meetings. The purpose of the triage is to determine the type of resolution for each bug and to prioritize and determine a schedule for all "To Be Fixed Bugs'. Team members will then assign the bugs to the appropriate person for fixing and report the resolution of each bug back into the bug tracker system. The Project Coordinator will be responsible for tracking and reporting on the status of all bug resolutions.

2.5 Test Completeness

Testing will be considered complete when the following conditions have been met:

2.5.1 Standard Conditions

- When Project Team members and the Project Sponsors agree that testing is complete, the application is stable, and agree that the application meets the functional requirements.
- Automated tests of all areas have passed.
- All priority 1 and 2 bugs have been resolved and closed.
- Project Sponsors approves the test completion.

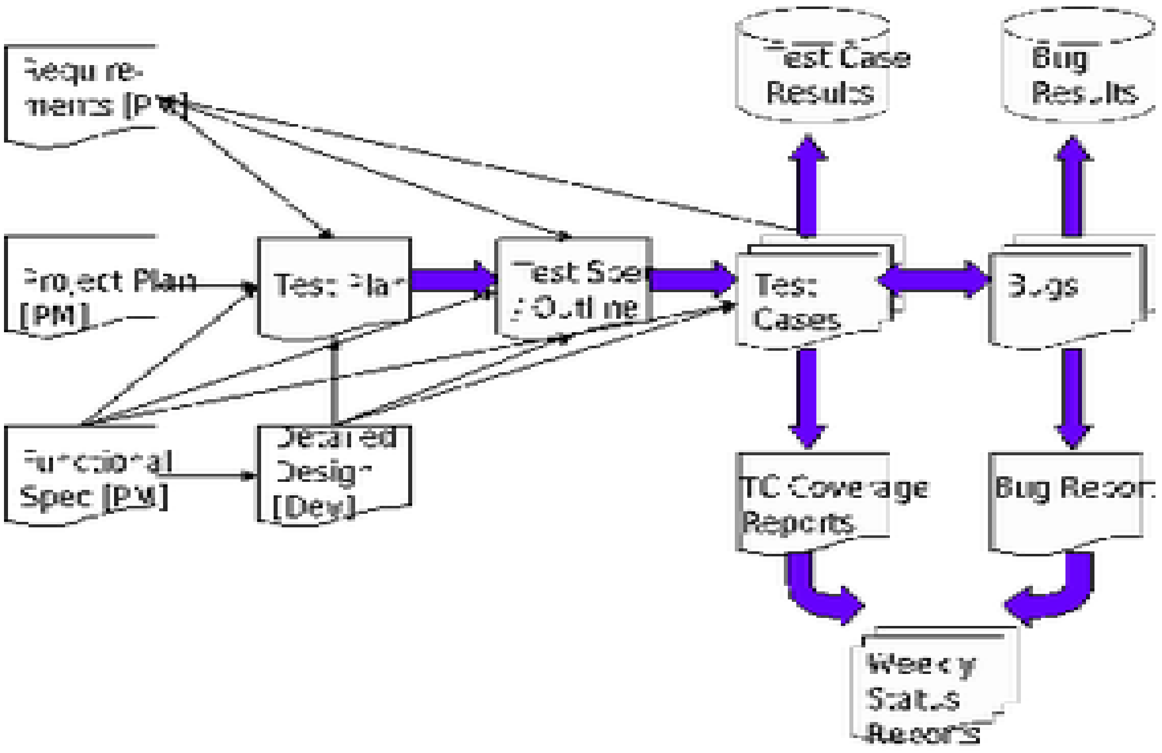
2.5.2 Bug Reporting & Triage Conditions

The following bug reporting and triage conditions will be submitted and evaluated to measure current status:

- Bug find rate
    - Expected to indicate a decreasing trend
    - If not decreasing, reevaluate designs and implementations
- But severity distribution
    - Expected steady decrease in Severity 1 and 2 bugs discovered
- No 'Must Fix' bugs remaining after sustained testing

3. Test Deliverables

Testing will provide specific deliverables during the project. These deliverables fall into three basic categories: Documents, Test Cases / Bug Write-ups, and Reports. Here is a diagram indicating the dependencies of the various deliverables:



As the diagram above shows, there is a progression from one deliverable to the next. Each deliverable has its own dependencies, without which it is not possible to fully complete the deliverable.

The following page contains a matrix depicting all of the deliverables that Testing will use.

3.1 Deliverables Matrix

The following list of artifacts are process-driven and will be produced during the testing lifecycle(s).

Deliverable
Documents
Test Approach
Test Plan
Test Schedule
Test Case / Bug Write-Ups
Test Cases / Results
Test Coverage Reports
YouTrack Bug tracker for bug reporting
Reports
Test results report
Test Final Report - Sign-Off

3.2 Documents

3.2.1 Test Plan

The purpose of the Test Plan document is to:

- Specify the construction of test cases
- Includes lists of test case areas and test objectives for each of the components to be tested
- Break the product down into distinct areas and identify features of the product that are to be tested.
- Specify the procedures to be used for testing sign-off and product release.
- Indicate the tools used to test the product.
- Indicate the contact persons responsible for various areas of the project.
- Specify criteria for acceptance of functionality

3.2.2 Requirements Traceability Matrix

A Requirements Traceability Matrix (RTM) [2] is used to link the test scenarios to the requirements and use cases. Requirements traceability is defined as the ability to describe and follow the life of a requirement, in both a forward and backward direction (i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases).

3.3 Defect Tracking & Debugging

3.3.1 Testing Workflow

In order to test a work flow, a unit test, individual code/documentation review, acceptance test, or regression test must first be performed. If a defect or bug is found, it is to be submitted to YouTrack with the following information:

- Detailed description of the bug/defect
- Subsystem relevant
- Asignee
- Priority
- Severity
- State (default to submitted)

After successful submission, a different team member is to review the defect. Should they agree that this is an issue, they are to change the state to “open”. The originating developer is expected to correct the defect, and change the state to “fixed” afterwards. The opening developer is expected to test for correctness, and change the state to “closed” after testing. Comments may be supplied to discuss the state of the defect.

3.3.2 Defect reporting using YouTrack

ALL defects should be logged using YouTrack, to address and debug defects. Developers will update the defect list on YouTrack and notify the requester after the defect has been resolved. Defect reporting is expected to be polite and non-accusational; accusations as to whose fault a bug is are not to be tolerated, and instead any bug is the responsibility of the team as a whole.

All **High priority** defects should be addressed within 2 days of the request and resolved/closed within 3 days of the initial request

All **Medium priority** defects should be addressed within 4 days of the request and resolved/closed within 5 days of the initial request

All **Low priority** defects should be resolved/closed no later than 7 days of the initial request.

3.4 Reports

The acceptance, usability, regression, and unit test leads are expected to oversee and report on the state of the testing for their role at least once during each release phase. The reports are to be verbal and informal, occurring during the weekly team meeting.

3.5 Responsibility Matrix

The following test role responsibility matrix indicates the manager of each aspect of testing. Although all developers are expected to participate in each area of testing, it is up to the manager to determine testing completeness and validity for their assigned area.

Name	Testing Role
Ian Dann	Acceptance Testing
Daquanne Dwight	Regression Testing
Tom Eiffert	Usability Testing
Elysia Haight	Unit Testing

4. Resource & Environment Needs

4.1 Testing Tools

4.1.1 Tracking Tools

YouTrack bug tracker is used to enter and track all bugs and project issues. The Test Lead is responsible for maintaining the YouTrack database.

4.2 Test Environment

4.2.1 Hardware

- Iphone 4 or newer
- Android 4.0 Phone with 256MB or more of RAM

4.2.2 Software

- Supported web browser (Safari, Chrome, Firefox, native browser)

4.3 Bug Severity and Priority Definition

Bug Severity and Priority fields are both very important for categorizing bugs and prioritizing if and when the bugs will be fixed. The bug Severity and Priority levels will be defined as outlined in the following tables below. Testing will assign a severity level to all bugs. The Test Lead will be responsible to see that a correct severity level is assigned to each bug.

4.3.1 Severity List

The tester entering a bug into YouTrack is also responsible for entering the bug Severity.

Severity ID	Severity Level	Severity Description
1	Critical	The application crashes or the bug causes non-recoverable conditions. System crashes, GP Faults, or database or file corruption, or potential data loss, program hangs requiring reboot are all examples of a Sev. 1 bug.
2	High	Major system component unusable due to failure or incorrect functionality. Sev. 2 bugs cause serious problems such as a lack of functionality, or insufficient or unclear error messages that can have a major impact to the user, prevents other areas of the app from being tested, etc. Sev. 2 bugs can have a work around, but the work around is inconvenient or difficult.
3	Medium	Incorrect functionality of component or process. There is a simple work around for the bug if it is Sev. 3.
4	Minor	Documentation errors or signed off severity 3 bugs.

4.3.2 Priority List

Priority ID	Priority Level	Priority Description
5	Must Fix	This bug must be fixed immediately; the product cannot ship with this bug.

4	Should Fix	These are important problems that should be fixed as soon as possible. It would be an embarrassment to the company if this bug shipped.
3	Fix When Have Time	The problem should be fixed within the time available. If the bug does not delay shipping date, then fix it.
2	Low Priority	It is not important (at this time) that these bugs be addressed. Fix these bugs after all other bugs have been fixed.
1	Trivial	Enhancements/ Good to have features incorporated-just are out of the current scope.

4.4 Bug Reporting

The Test Lead will be responsible for managing the bug reporting process. YouTrack will be used to report and maintain any discovered bugs. Team members will enter their data into YouTrack following the field entry definitions below.

5. Definitions

TERM/ACRONYM	DEFINITION
OOO	Out of Office
UI	User Interface

6. References

[1] Requirements Document.docx

[2] Requirements Traceability Matrix