Braille in DAISY Pipeline 2

Test strategy

|  |  |
| --- | --- |
| Document attributes | Information |
| Title and subtitle | Braille in DAISY Pipeline 2: Test strategy |
| Document owner | Björn Westling |
| Document type | Test strategy |
| Version | 1.3 |
| Latest date of change | 2015-06-23 |
| Reg.no. | ---- |

**Contents**

[1 Introduction 4](#_Toc422837914)

[2 Scope 4](#_Toc422837915)

[2.1 Scope exclusion 4](#_Toc422837916)

[3 Motivation 4](#_Toc422837917)

[4 Stakeholders 5](#_Toc422837918)

[5 Test objects 5](#_Toc422837919)

[6 Product features and requirements 5](#_Toc422837920)

[7 General workflow 6](#_Toc422837921)

[7.1 Test driven 6](#_Toc422837922)

[7.2 Agile 6](#_Toc422837923)

[7.3 Risk based 6](#_Toc422837924)

[8 V-model 6](#_Toc422837925)

[9 Test levels 7](#_Toc422837926)

[9.1 Unit testing 7](#_Toc422837927)

[9.2 Integration testing 7](#_Toc422837928)

[9.3 System testing 8](#_Toc422837929)

[9.4 Acceptance testing 8](#_Toc422837930)

[10 Test types 8](#_Toc422837931)

[10.1 Functional testing 9](#_Toc422837932)

[10.2 Regression testing 9](#_Toc422837933)

[10.3 Non-functional testing 9](#_Toc422837934)

[10.3.1 Configurability 9](#_Toc422837935)

[10.3.2 Security 9](#_Toc422837936)

[10.3.3 Speed/velocity 9](#_Toc422837937)

[10.3.4 Format support 9](#_Toc422837938)

[10.3.5 Platform support 9](#_Toc422837939)

[10.3.6 Usability 9](#_Toc422837940)

[11 Test environment 9](#_Toc422837941)

[11.1 Test server 9](#_Toc422837942)

[11.2 System design 10](#_Toc422837943)

[11.3 Code 10](#_Toc422837944)

[12 Test cases 10](#_Toc422837945)

[13 Test data 10](#_Toc422837946)

[14 Test tools 10](#_Toc422837947)

[14.1 Unit testing 10](#_Toc422837948)

[14.2 System testing 10](#_Toc422837949)

[14.3 Acceptance testing 11](#_Toc422837950)

[15 Defect management 11](#_Toc422837951)

[16 Test activities 11](#_Toc422837952)

[16.1 Test activities in a sprint 11](#_Toc422837953)

[16.2 Test activities pre-production 12](#_Toc422837954)

[17 Test roles 12](#_Toc422837955)

[18 Test documentation 12](#_Toc422837956)

1. Introduction

The purpose of this document is to provide the Braille in DAISY Pipeline 2 project with a description of the test strategy, including test objects (what shall be tested), test levels (which approach that should be used), test types (when different types of tests should be carried out), test approval criteria, and roles.

This document will also be used in order to later define the test plan and to provide visibility to project stakeholders in the testing effort that sufficient consideration has been given to various aspects of governing the testing effort, and where appropriate to have those stakeholders approve strategy and plan.

1. Scope

The scope for this document is to provide a central artifact to govern the planning and control of the test strategy. It defines the general approach that will be employed at Braille in DAISY Pipeline 2 organization to test the product and to evaluate the results of that testing, and is the top-level plan that will be used by managers to govern and direct the detailed testing work.

* 1. Scope exclusion

Validation of functionality of the final production (official release) environment post acceptance testing is not in scope for this document.

1. Motivation

The motivation behind implementing test strategy and test management for the Braille in DAISY Pipeline 2 project is to ensure that:

* the project plan is fulfilled and milestone targets are met
* the stakeholder business investment, goals and requirements are satisfied
* expected product quality, function and performance characteristics are met early and cost efficiently in the development phase
* the organization and end users gain confidence early in the product lifecycle
* product testing and quality assurance is performed cost efficiently and minimally
* project resources are utilized cost efficiently focusing on prioritized business and product requirements
* the organization secures a solid, functional product baseline for effective product lifecycle management and further product iterations
* the product secures prevention of damage or faults during live production
* late and costly reworking and consequential damage in production is prevented
* good project management practices are facilitated through the supply of visibility and risk management through access to good quality information

1. Stakeholders

|  |  |
| --- | --- |
| Name | Project partner |
| Richard Stones | MTM |
| Arne Kyrkjebø | NLB |
| Manfred Muchenberger | SBS |
| Ole Holst Andersen | Nota |
| Seppo Mallenius | Celia |
| Inge de Mönnink | Dedicon |
| Björn Westling | MTM |

1. Test objects

The current test objects in the project are also the product lifecycle management (PLM) objects:

* Liblouis module
* Dotify module
* DAISY Pipeline 2 module
* Libhyphen module
* Web UI module

DAISY Pipeline 2 is the production framework that builds on the Dotify and Liblouis components.

1. Product features and requirements

The product feature requirements are located centrally on a web repository:

<https://drive.google.com/drive/folders/0B_Q0CSMAbmEdeHFhMDBfeUJPaHM>

A working subset, including functional sub-branches, for testing and merging requirements changes are located on:

<http://dev.pef-format.org/dp2/master/index.xhtml>

The requirements repository will have a central role. It has multiple purposes:

* For interpreting, explaining the requirements on various levels including a main requirements list and separate sub-requirements lists reflecting various and isolated software branches
* A formal way of improving and requesting features (a contract between braille experts and developers: if it's not in a test it's not a requirement)
* A test suite and test protocol for unit testing (development testing) i.e. automatic validation during development of the software to detect bugs
* A showcase of product functionality

1. General workflow
   1. Test driven

Product development will be test data driven, meaning that test data is created by the braille experts first by using braille books, braille handbook reference material, or existing braille translator material. The files are then forwarded to the developers for possible file adaptation and conversion to verify the product functionality.

The main reason for this approach is that the requirements need to be reverse engineered and refined based on the fact that the individual requirements per language and format require specific output vs. input.

* 1. Agile

The work will be planned and organized in short sprints and iterations with continuous deployment and fast feedback from the braille experts (testers). In each sprint iteration the team should focus on a set of features and requirements related tasks, finish them before starting new sprints and tasks hence creating gradual functional product increments in each separate sprint along the product development lifecycle.

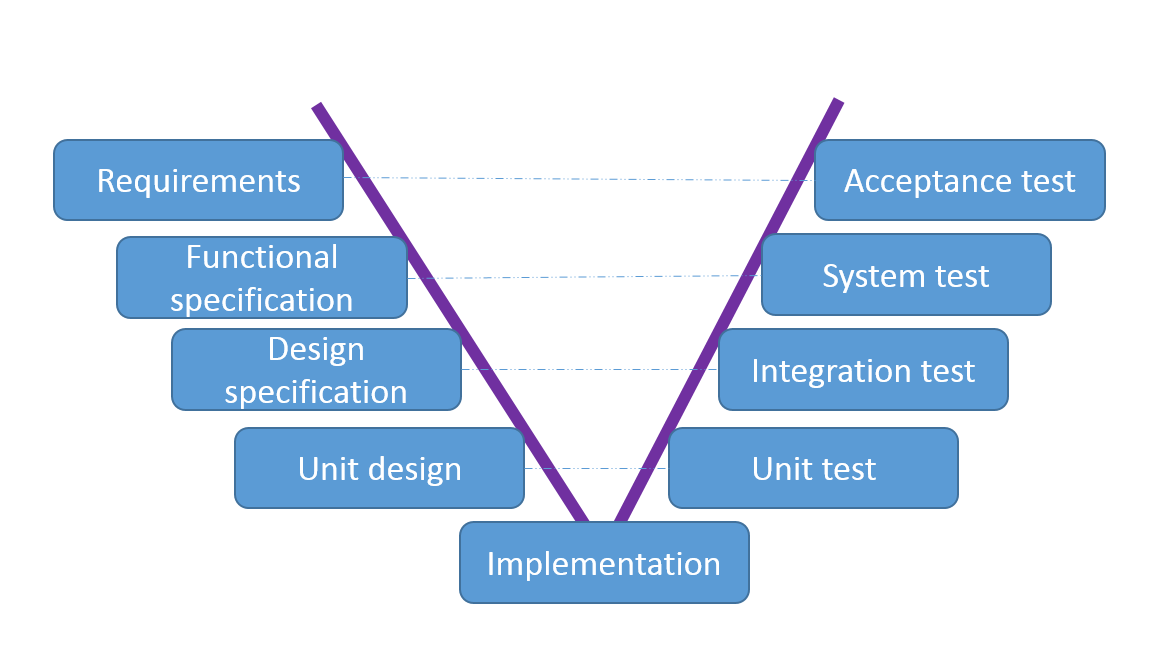
The sprints need to be planned based on feature priorities, resource allocations, pipeline roadmap and Web UI roadmap.

* 1. Risk based

The Braille in DAISY Pipeline 2 project intends to use a risk based test approach by identifying and prioritizing features from a business and functional perspective. This means that the most important product features, and risks, in the general product backlog and in the sprint backlogs (sprint-by-sprint), should be prioritized and planned for testing. In order to identify and determine the risk level in the right time a risk analysis should be done on the product requirements and features by e.g. using the Product Risk Analysis, PRA, document.

1. V-model

The V-Model is a framework and sequential model covering the complete scope and activities of a software development cycle from the requirements phase to the acceptance test phase. The V-model describes how test activities can be integrated into each step of the software development cycle. The below picture illustrates the recommended V-model for the Braille in DAISY Pipeline 2 project.



1. Test levels

The following test levels are applicable in the project. Each level has a specific focus and purpose.

* 1. Unit testing

Unit testing, also known as component testing, refers to manual or automated tests that verify the functionality of a specific section of code, e.g. a translator, usually at the function level. It is performed by the software developer during the design phase of the software development lifecycle.

**Entry and completion of test phase**

Unit testing is initiated when test data is generated by the braille experts (testers) and made available to the developers for unit and development testing.

Unit testing is completed when the expected test results are reached based on the feature requirements and an agreement between the developer and tester.

* 1. Integration testing

Integration testing is the phase in which individual software modules are combined and tested as a group, e.g. a CSS-based formatter that uses translators internally, as well as verifying that interfaces between systems are working correctly. It may also involve client/server integration testing. Integration testing for the Braille in DAISY Pipeline 2 project could be handled implicitly through code build and client/server interaction during system testing.

**Entry and completion of test phase**

Integration testing is initiated when the sub-functional unit testing is completed and the build delivery to the test server is ready.

Integration testing is completed when all integration and sub-functional tests are performed successfully including a Pass verdict.

* 1. System testing

The purpose of system testing is to verify complete end-to-end system functionality. The tests are primarily based on requirements, use cases, risks and system behavior. Testing is performed on a test server in a pre-production like (partial or full pre-release) environment. Usability is key focus for this project.

**Approval criteria**

The following approval criteria for the system testing should be fulfilled in order to start acceptance testing:

* No Blocking or High priority issues present in the system test report
* Medium or Low priority issues present in the test report but require an action plan agreed between the stakeholders
* Test report approved

**Entry and completion of test phase**

System testing is initiated when the build delivery on the test server is confirmed and ready for testing after a basic smoke test.

System testing is completed when all the system tests are performed and with results officially stated in the test report. The approval criteria need to be fulfilled.

* 1. Acceptance testing

The purpose of the acceptance test is for the organization and business to verify that the intended delivery meets the business and product requirements.

**Approval criteria**

The following approval criteria should be fulfilled for the acceptance test in order to be considered production ready:

* No Blocking or High priority issues present in acceptance test report
* Medium or Low priority issues present in the test report but require an action plan agreed between the stakeholders

**Entry and completion of test phase**

Acceptance testing is initiated upon system test completion and approval. A meeting is held between the project stakeholders in order to review the contents of the delivery and the system test results.

Acceptance testing is completed upon finalized acceptance test execution and approval. A meeting is held between the project and business stakeholders in order to review the acceptance test results.

1. Test types

Within each test level various test types are performed. There are several different test types and below are the applicable ones for this project:

* 1. Functional testing

Functional testing verifies that new functionality works according to the product requirements.

* 1. Regression testing

Regression testing is an activity performed as long as there is a need to verify that the functionality has not been affected by the developer’s corrections or changes in the previous build within the corresponding test phase.

It should not be allowed to start the system test prior to a regression being finished and evaluated by the function test team and normally the regression is not executed before the product has reached stability and maturity.

The extent of the regression test is ruled by the risk assessment of the actual changes.

* 1. Non-functional testing

Tests based on non-functional requirements.

* + 1. Configurability

Implicitly tested through functional requirements.

* + 1. Security

Testing to determine the security of the software product. Implicitly tested through functional system testing.

* + 1. Speed/velocity

Performance tests from a usability perspective reflecting on response times, job handling and support for multiple users client/server. Implicitly tested through functional system testing.

* + 1. Format support

Input and output formats implicitly tested through functional requirements.

* + 1. Platform support

Testing of industry standard server/client conformance. By design.

* + 1. Usability

Implicitly tested through functional requirements, system and acceptance testing.

1. Test environment
   1. Test server

The [test server](http://158.39.20.55) is the location where separate and integrated software builds, including both pipeline and braille modules, for system and acceptance testing are uploaded: <http://158.39.20.55>

* 1. System design

The [system design document](https://docs.google.com/document/d/1LTQRQZ0YKZ5LXyOSq3O-TXfAyIoWqCpG2T5NApEdv3Q/pub) explains the various modules and functionality split in the system.

* 1. Code

The code is placed on GitHub: <https://github.com/snaekobbi>

1. Test cases

Test cases will be an integral part of the future test protocol document (MS Word). A central repository for the test protocols and test cases need to be made available.

1. Test data

Test data is currently generated by the braille experts (testers) and forwarded to the developers for possible file type conversion during development.

Test data for unit (functional) testing is mainly stored locally by the braille testers and post conversion for development purposes on:

<http://dev.pef-format.org/dp2/master/index.xhtml>

Test data for system testing and acceptance testing will be using text files and content as intended for the final product.

Original test data reference repository is proposed to be on GitHub:

<https://github.com/snaekobbi/liblouis/tree/formal_braille_spec>

New link: <https://github.com/liblouis/braille-specs>

1. Test tools
   1. Unit testing

Test tool for automated unit/functional development testing is available on: *http://dev.pef-format.org/dp2/<branch name>/index.xhtm*

Example: <http://dev.pef-format.org/dp2/finnish/index.xhtml>

Tests are run automatically after each commit and build and a test result is published and available at the same location.

Test tool code is placed on:

<https://github.com/snaekobbi/requirements>

Unit tests are written in Java (JUnit), [XSpec](https://code.google.com/p/xspec/) and [XProcSpec](http://josteinaj.github.io/xprocspec/).

* 1. System testing

System testing will be manual and the test protocol document will be the test tool for step-by-step test case execution and tracking.

Test tool for tracking overall test tasks such as test case development and status in each sprint is placed on the product backlog repository, HuBoard:

<https://huboard.com/snaekobbi/issues>

In order to track sprint status and test status more efficiently, some ore granularity and visibility (steps) need to be added to HuBoard.

One additional tool, [Trello](https://trello.com/b/bwspupRy/braille-in-dp2), has been discussed for more top level sprint execution visibility and tracking on top of HuBoard.

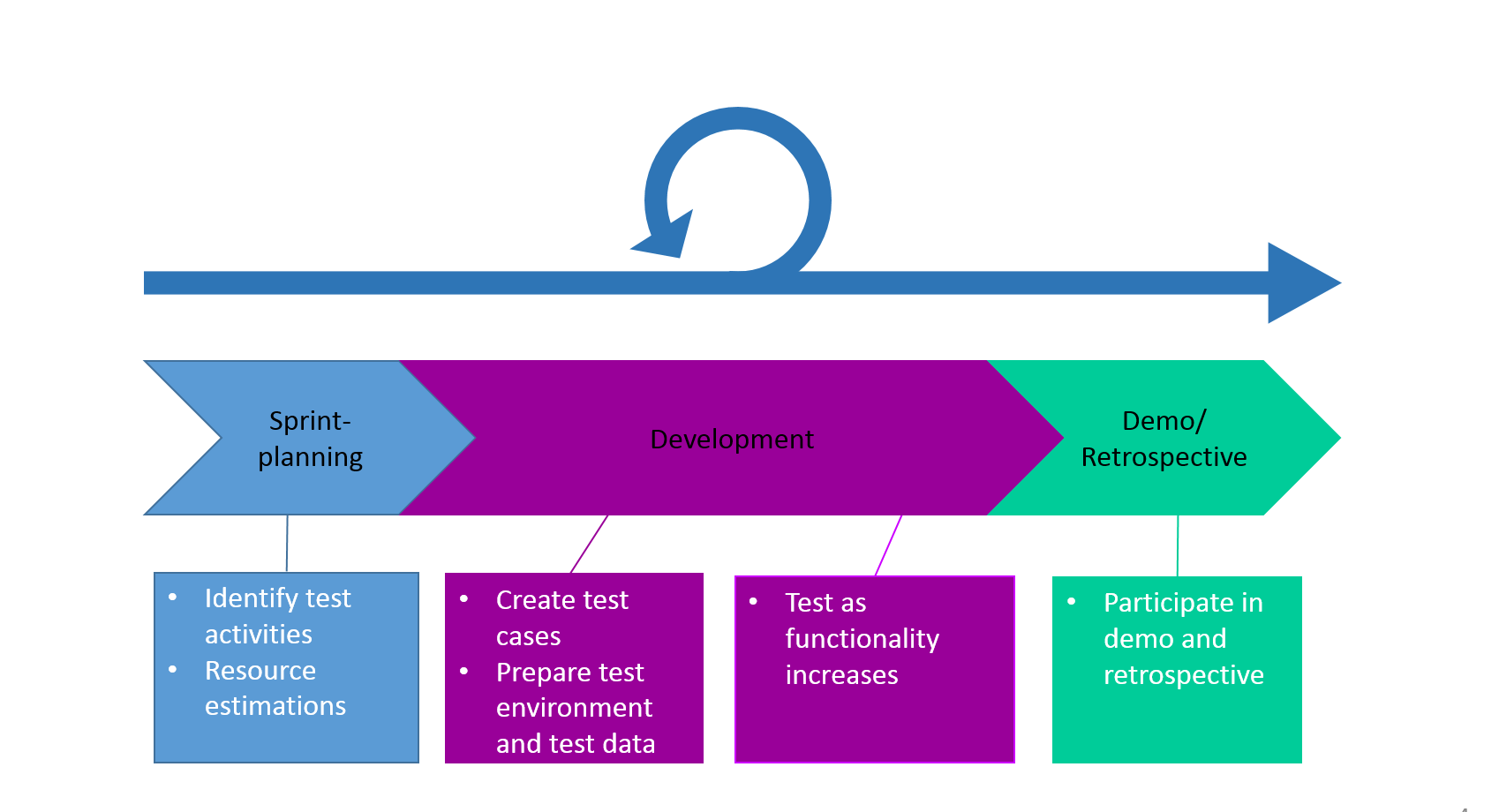
* 1. Acceptance testing

Acceptance testing will be manual with select key functionality features tested and logged in an acceptance test protocol (MS Excel). The system test protocol document, including the test cases for the select features, will be the tool for step-by-step acceptance testing.

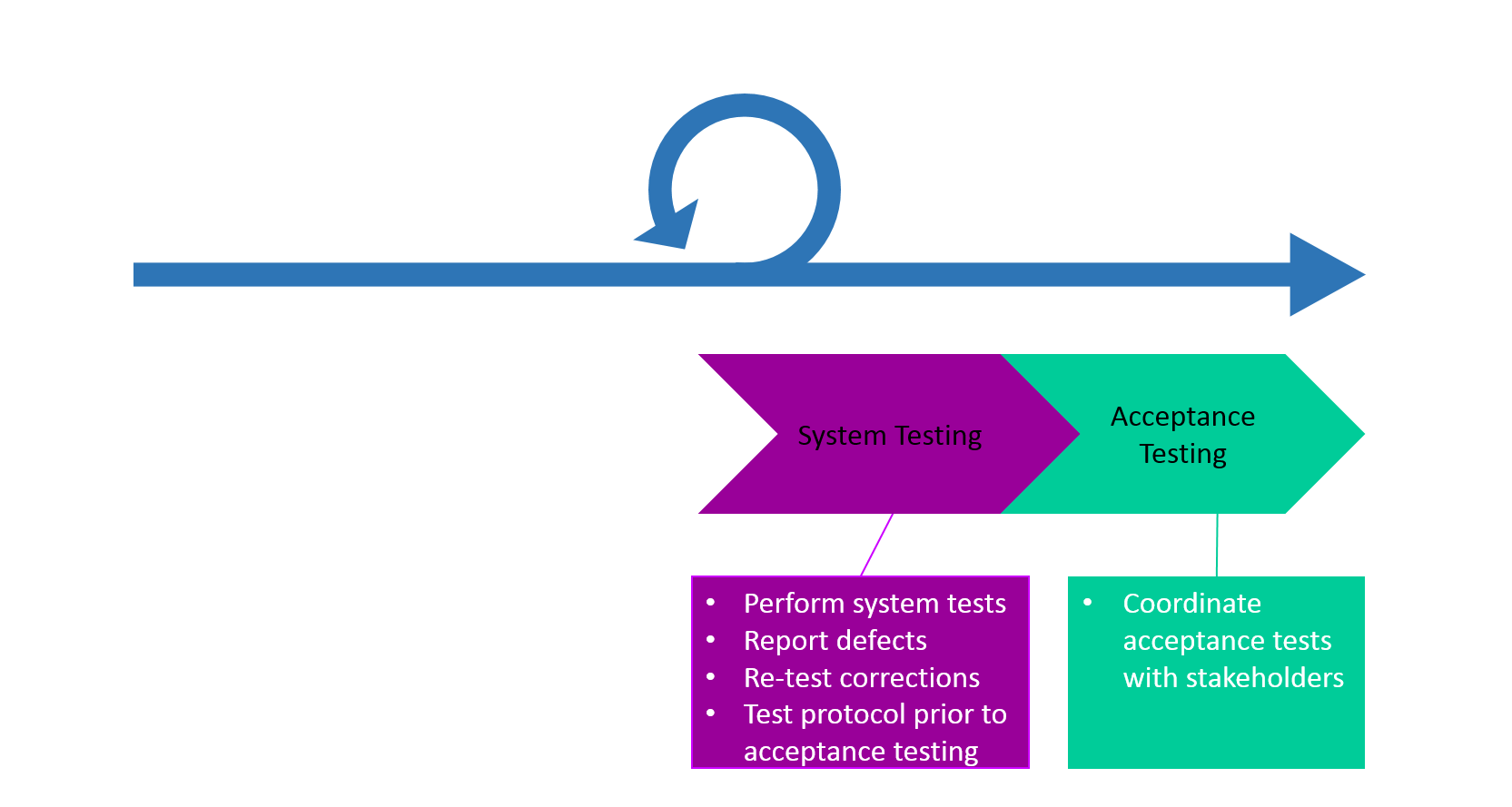
1. Defect management

Tools for defect management are not in place at this point in time.

1. Test activities
   1. Test activities in a sprint



* 1. Test activities pre-production



1. Test roles

A test role can be held by one or more individuals and one individual can have multiple test roles.

|  |  |  |
| --- | --- | --- |
| Test role | Responsibility | Reports to |
| Test manager | Creates test plan. Writes test cases. Coordinates tests and reports progress and results. | Project manager |
| Tester | Creates test data. Writes test cases. Refines requirements. Does system testing. | Test manager |
| Developer | Does unit testing. Refines requirements. Fixes and re-tests issues. | Test manager |

1. Test documentation

The responsibility to document testing in a structured fashion is on the test manager. All test documents need to be reviewed and made available on a central repository. The test manager coordinates the review meetings.

The test protocol documentation should reflect good revision and traceability practice between requirements references, test cases, test data and test results.

|  |  |  |
| --- | --- | --- |
| Documentation | Responsible | Delivered to |
| Test plan | Test manager | Project manager |
| Test case | Tester | Project manager |
| Test report | Test manager | Project manager |