SITE Product Integration Strategy

# Integration Environment

The integration environment for the SITE project is located in the team’s document repository (<https://github.com/siteadmin/documentation.git>).

# Analysis/Requirements

The client and product owner identify functional/business requirements. These requirements are entered into the product backlog in ONC’s JIRA instance (<http://jira.oncprojectracking.org>) as user stories. As the requirements are prioritized in the product backlog, high-level design components are logged in JIRA as Epics to allow smaller, more manageable user stories, to be created. All user stories should tie back to a particular Epic.

For each development related user story that is added to the product backlog, a corresponding testing (QA), and delivery (deployment) user story should be added to the backlog.

Prior to adding a user story to a sprint, the user story should be documented with enough detail to successfully complete the associated task. The acceptance criteria for the user story should be defined by the product owner, and documented on the user story in JIRA.

# Design

The overall technical architecture design is managed by the technical lead/architect, and is located in the SITE document repository (<https://github.com/siteadmin/documentation.git>).

## Interfaces

During sprint planning, as user stories are prioritized, new features are analyzed to evaluate if the feature impacts the current high level design/interfaces. If a feature impacts the high-level design/interfaces, a user story is created in JIRA and assigned to the technical lead/architect to update the technical architecture design.

## Refactoring

If the feature requires a new component to be added to the high-level design or modifications to existing component interfaces, refactoring of the code-base is acceptable. Refactoring of the code for an individual component is acceptable as long as the refactoring of the component is justified by added value to the client. All refactoring decisions should be vetted through the technical lead/architect. The product owner must approve refactoring of the high-level design.

## Vendor Supplied Components/Modules

If the required feature dictates a specific technology (as prescribed the client), the specific vendor supplied component will be utilized. If a user story dictates common functionality, third party components will be evaluated. The SITE project is released as open source; therefore, all third-party components must be similarly licensed.

If the estimated cost of procuring, configuring, and supporting a third-party supplied component is less than the comparative cost to build the same component, then the team will recommend the adoption of the of the vendor supplied component. Once a technology is recommended, a user story is created to establish a proof-of-concept, prior to purchase. This gives the team an opportunity to verify that the technology/tool works as advertised.

# Development

Developers commit all of their product source code into the SITE github repository located at <https://github.com/siteadmin/>. All development is initially committed to a personal development branch. Once a feature has been fully unit-tested, the code can be merged with the current TEST branch and deployed to the corresponding TEST server.

## Code Review Verification

Upon completion of the development a new component or a refactor of an existing component, a peer code review is performed. If for some reason there is less than 1 developer FTE allocated to the project at any time, a peer code review can be excused, as there are not enough billable developers to perform the review. Recommended changes that are output of the peer review are to be made and reviewed prior to promotion to the TEST branch.

Code Peer Reviews are tracked via user stories in the JIRA project manager tool. Any findings are logged in the user story and work performed to fix any issues is logged against the user story.

# Quality Assurance

The QA team performs product verification tasks. The QA team prepares a test plan that is stored in the project’s document repository (<https://github.com/siteadmin/documentation>).

The test plan should contain the following:

* Reference docs and location (Requirements, Design, User Guide, etc.)
* Supporting files and location (e.g. sample files)
* Defines the different areas of the Module/Project that requires validation.
* Defines High Level Testing steps to undertake and different checkpoints per phase/area.

From the test plan, the QA team creates detailed test cases from the reference documentation. The detailed test cases are documented in the SITE project’s document repository (<https://github.com/siteadmin/documentation.git>). These documents contain the following:

* Test Cases
* Requirement details

Once the test cases are documented, a QA Engineer executes the test cases and logs the results in the detailed test cases document. Any identified defects are logged into the JIRA system.

## Defects

Defects are logged into the JIRA system. Upon creation, the defect is assigned to an Epic. Non-blocking defects are prioritized and assigned in the same manner as user stories (through the sprint planning process). During the product backlog prioritization process, the product owner can elect to reject any non-blocking defects.

Defects that block the QA process are assigned to the immediate sprint and should be addressed as soon as possible to permit testing to continue.

Once defects are addressed, the QA engineer will retest the defects and log the results. All test cases should pass, unless the product owner rejects a specific defect.

## Verification of Testing

Once testing is completed, the Technical Lead/Product Owner reviews the test results and schedules a deployment of the component.

## Bi-directional Traceability of Requirements

To ensure bidirectional traceability of requirements, the SITE team utilizes “Epics” in JIRA. Requirements are logged in User Stories that are tied to a specific Epic. Testing tasks are assigned to the same Epic. Any defects logged would be assigned to the same Epic, for traceability.

# Product Delivery/Deployment

New code/functionality is deployed to the Production environment by either the responsible developer or the Technical Lead.

## Deployment Checklist

The developer/technical lead follows the project’s deployment checklist. The project’s deployment checklist defines the steps required to perform an application deployment. Completed checklists are stored in the team’s document repository (<https://github.com/siteadmin/documentation>).

## Supporting Documentation

The developer is responsible for generating user/install documentation. This documentation should be stored in the team’s document repository (<https://github.com/siteadmin/documentation>).