ADS 18f Prototype

Drug Interaction Query

(Drug IQ)

Geocent Program Management Plan

Job 4QTFHS150004

4QTFHS150004

Version 1.0

July 7, 2015



**Distribution Statement: This document is for internal planning purposes only.**

**Document Authorization**

|  |  |
| --- | --- |
| **Review/Approval Title** | **Name** |
| **Originator:**  Technical Architect | Jared Ladner |
| **Approved By**:  Program Manager | Keith Alphanso |
| **Approved By**:  Project Lead | Jared Ladner |
| **Approved By**:  Quality Assurance | Brian Priest |

Electronic signatures were provided on 07/07/2015

**Document Revision History**

Revision history for the Project Management Plan (PMP) will be tracked in GitHub at the following link:

<https://github.com/Geocent/18f-prototype/tree/master/docs>

Each iteration of the document will be stored within the Sprint folders.

**Table of Contents**

[1 Project Overview 1](#_Toc423893734)

[1.1 Project Scope 1](#_Toc423893735)

[1.1.1 Customer 1](#_Toc423893736)

[1.1.2 Project Vision 1](#_Toc423893737)

[1.1.3 Schedule Summary 1](#_Toc423893738)

[1.1.4 Period Of Performance 2](#_Toc423893739)

[1.1.5 Milestones & Releases 2](#_Toc423893740)

[1.1.6 Deliverables 3](#_Toc423893741)

[2 References 3](#_Toc423893742)

[2.1 Source Documentation 3](#_Toc423893743)

[2.2 Project Working Documents 4](#_Toc423893744)

[3 Organization 5](#_Toc423893745)

[3.1 Internal Structure: Geocent Program Management Structure 5](#_Toc423893746)

[3.2 Roles and Responsibilities 6](#_Toc423893747)

[4 Managerial Plans 7](#_Toc423893748)

[4.1 Start-Up Plan 7](#_Toc423893749)

[4.1.1 Establish Resources 8](#_Toc423893750)

[4.2 Execution Plan 9](#_Toc423893751)

[4.2.1 Manage Code 9](#_Toc423893752)

[4.2.2 Manage Requirements 16](#_Toc423893753)

[4.2.3 Manage Communications 16](#_Toc423893754)

[4.2.4 Manage Quality 17](#_Toc423893755)

[4.2.5 Manage Risks/Issues/Corrective Actions 19](#_Toc423893756)

[4.2.6 Manage Measurements (Metrics) 20](#_Toc423893757)

[4.3 Work Plan 20](#_Toc423893758)

**List of Figures**

[Figure 1: Program Management Structure 5](#_Toc423893759)

[Figure 2: Master/Integration Branch 9](file:///C:\Users\rhazelbaker\Desktop\GSA%20BPA%20Agile%20RFQ\Sprint%20Delivery\Geocent%20Project%20Management%20Plan%20-%20Delivery.docx#_Toc423893760)

[Figure 3: Feature Branch 10](file:///C:\Users\rhazelbaker\Desktop\GSA%20BPA%20Agile%20RFQ\Sprint%20Delivery\Geocent%20Project%20Management%20Plan%20-%20Delivery.docx#_Toc423893761)

[Figure 4: Feature Branch Incorporated Back to Integration 11](#_Toc423893762)

[Figure 5: Hotfix Branching 14](file:///C:\Users\rhazelbaker\Desktop\GSA%20BPA%20Agile%20RFQ\Sprint%20Delivery\Geocent%20Project%20Management%20Plan%20-%20Delivery.docx#_Toc423893763)

**List of Tables**

[Table 1: Sprint Schedule 1](#_Toc423893764)

[Table 2: Project Source Documents 3](#_Toc423893765)

[Table 3: Project Internal Working Documents 4](#_Toc423893766)

[Table 4: Drug IQ Roles, Labor Categories, and Personnel 6](#_Toc423893767)

[Table 5: Training Course Description 8](#_Toc423893768)

[Table 6: Hardware Resources 8](#_Toc423893769)

[Table 7: Tools and Methods to Manage Requirements 16](#_Toc423893770)

[Table 8: Scrum Meeting Plan 17](#_Toc423893771)

[Table 9: Project Measures 20](#_Toc423893772)

# Project Overview

This document will establish the Program Management Plan (PMP) for the Drug Interaction Query (Drug IQ) project and will serve as the execution plan that Geocent will use to manage the project.

## Project Scope

### Customer

The customer for this project is GSA 18f. Drug IQ will be designed to support customer (non-medical professionals) and medical professionals.

### Project Vision

The Project Vision that will define Drug IQ is stored at the following link:

<https://github.com/Geocent/18f-prototype/tree/master/docs/scrum/sprint0>

### Schedule Summary

Drug IQ will be developed using an agile schedule based on five Sprints. Table 1 details the Sprint schedule that will be used to support Drug IQ development and delivery.

Table : Sprint Schedule

| **Sprint Iteration** | **Description** | **Dates** |
| --- | --- | --- |
| Sprint 0 | Capture Sprint  (kick-off meeting, brainstorm project vision, goals, Definition of Done, development plan, testing plan, usability testing team, establish environments, establish Sprint duration and schedules) | Project Kick-off: June 16  Sprint 0:   * Start: 6/17/2015 * Sprint Planning: 6/17 * Sprint: 6/17-6/18 * Sprint Review: 6/18   Duration: 2 days |
| Sprint 1 | Prototype: Architecture, Development, and Test Sprint | Sprint 1:   * Sprint: 6/18-6/22 * Sprint Planning: 6/18 * Sprint Review: 6/23   Duration: 2 days |
| Sprint 2 | Prototype: Finalization Sprint | Sprint 2:   * Sprint: 6/23-6/25 * Sprint Planning: 6/23 * Sprint Review: 6/25   Duration: 2 days |
| Sprint 3 | Prototype: Enhancement Sprint | Sprint 3:   * Sprint: 6/25-6/29 * Sprint Planning: 6/25 * Sprint Review: 6/29   Duration: 2 days |
| Sprint 4 | Stabilization Sprint | Sprint 4:   * Sprint: 6/29-6/30 * Sprint Planning: 6/29 * Sprint Review: 6/30   Duration: 1 day |
|  | Delivery | * Sprint: 7/1-7/7 * Sprint Planning: 7/1 * Sprint Review: 7/7   Deliver: 7/7 |

The agile Sprint schedule will be managed in ScrumDo. Agile Sprint artifacts in ScrumDo will be exported to GitHub and will be stored at the following link:

Placeholder for link

### Period Of Performance

The Period of Performance will be from Wednesday June 17th, 2015 through Tuesday,   
July 7th, 2015.

### Milestones & Releases

The milestones for Drug IQ will include:

* Initial mock up design
* Initial working prototype on AWS
* Final delivery of Drug IQ

Drug IQ will utilize continuous delivery which will release new versions at the end of each Sprint. The release will be contingent upon code build pass, Unit Tests and Integration Tests pass, and the Product Owner verifying User Acceptance Testing.

### Deliverables

The following list will comprise the services, products, documentation, and other deliverables associated with the project:

* Working Prototype on Public Site (AWS)
* Technical Approach hosted in GitHub (add link)
* Agile practices in ScrumDo and exported into GitHub (add link)
* Complete Agile Delivery Services RFQ Compliance Review Checklist Google Form
* Attachment B: Labor Category Mapping and Discount
* Attachment C: Price Quote Sheet

# References

## Source Documentation

Table 2 lists the documents that will form the foundation for establishing the Drug IQ PMP. The project source documents are listed in order of precedence in the event there are competing statements within multiple documents.

Table : Project Source Documents

|  |  |
| --- | --- |
| **Work Product Identifier/Title** | **Path / Storage Location** |
| SOW/RFP/RFQ/Initial Requirements | <https://intranet.geocent.com/secure/businessdevelopment/SiteDirectory/gsaagile/_layouts/15/start.aspx#/> |
| Contract | <https://intranet.geocent.com/secure/businessdevelopment/SiteDirectory/gsaagile/_layouts/15/start.aspx#/> |
| Proposal | <https://intranet.geocent.com/secure/businessdevelopment/SiteDirectory/gsaagile/_layouts/15/start.aspx#/> |

## Project Working Documents

Table 3 lists the documents that will support the regular tracking and management of all aspects of the Drug IQ project. Table 3 includes the name of the work product, the link/path to the document’s location, and the level of Configuration Management (CM) control.

Table 3: Project Internal Working Documents

|  |  |  |  |
| --- | --- | --- | --- |
| Project Internal Working Documents | | | |
| **Work Product Identifier/Title** | **Path / Storage Location** | **Level of CM Control** |
| Project Management Plan | <https://github.com/Geocent/18f-prototype/tree/master/docs> | Project Lead |
| Product Backlog | [Scrumdo.com Backlog](https://www.scrumdo.com/projects/project/18f-ads-prototype/iteration/128546) | Team |
| Backlog Grooming /Sprint Planning Report | <https://github.com/Geocent/18f-prototype/tree/master/docs/scrum> | Scrum Master |
| Sprint Review | <https://github.com/Geocent/18f-prototype/tree/master/docs/scrum> | Scrum Master |
| Sprint Launch/Kick Off/Sprint 0 | <https://github.com/Geocent/18f-prototype/tree/master/docs/scrum> | Scrum Master |
| Sprint Retrospective | Included in Sprint Review | Scrum Master |
| Code Review Form Template | <https://github.com/Geocent/18f-prototype/tree/master/docs/scrum/sprint0> | QA Manager |
| Code Review Forms - Completed | <https://github.com/Geocent/18f-prototype/tree/master/docs/scrum>  Code Review forms tracked by Story number within each Sprint | QA Manager/ Reviewers  Developers |
| Sprint Status Reports | [ScrumDo](https://www.scrumdo.com/projects/project/18f-ads-prototype/summary) and export to [GitHub Repo](https://github.com/Geocent/18f-prototype/tree/master/docs/scrum) | Scrum Master |
| Geocent Status Reports | [ScrumDo](https://www.scrumdo.com/projects/project/18f-ads-prototype/summary) and export to [GitHub Repo](https://github.com/Geocent/18f-prototype/tree/master/docs/scrum) | Project Lead |
| Project Review Documents (IPR Presentations) | [ScrumDo](https://www.scrumdo.com/projects/project/18f-ads-prototype/summary) and export to [GitHub Repo](https://github.com/Geocent/18f-prototype/tree/master/docs/scrum) | Project Lead |
| Project Burn-Down Chart | [ScrumDo](https://www.scrumdo.com/projects/project/18f-ads-prototype/summary) and export to [GitHub Repo](https://github.com/Geocent/18f-prototype/tree/master/docs/scrum) | Scrum Master |
| Geocent Project Estimator | [ScrumDo](https://www.scrumdo.com/projects/project/18f-ads-prototype/summary) and export to [GitHub Repo](https://github.com/Geocent/18f-prototype/tree/master/docs/scrum) | Project Manager |

# Organization

## Internal Structure: Geocent Program Management Structure

Figure 1 illustrates the 18f ADS Prototype program management structure that will support this effort. This organizational chart also represents the program’s relationship to the Geocent organization.

Figure : Program Management Structure

## Roles and Responsibilities

Table 4 details the roles, labor categories, and personnel targeted to support the Drug IQ project.

Table : Drug IQ Roles, Labor Categories, and Personnel

| **Drug IQ Project Roles** | | | |
| --- | --- | --- | --- |
| **Role** | **Responsibilities, Required Knowledge/Skills** | **Labor Category** | **Person** |
| Product Owner | See Attachment A Labor Category Descriptions.pdf | Product Manager | Keith Alphonso |
| Project Lead | See Attachment A Labor Category Descriptions.pdf | Technical Architect | Jared Ladner |
| Scrum Master | See Attachment A Labor Category Descriptions.pdf | Agile Coach | Roberta Hazelbaker |
| UX Lead | See Attachment A Labor Category Descriptions.pdf | Visual Designer | Vance Lowe |
| Developers | See Attachment A Labor Category Descriptions.pdf | Front end Web Developers | Randy Nolan Aaron Whitney  Josh Penton |
| DevOPS Lead | See Attachment A Labor Category Descriptions.pdf | DevOPS Engineer (Eng) | Tyler Sanders |
| QA Lead | See Attachment A Labor Category Descriptions.pdf | Delivery Manager | Brian Priest |

# Managerial Plans

## Start-Up Plan

The start-up plan will constitute Sprint 0. It will begin with a Kick-off meeting to outline the details of the project requirements set by GSA 18f, the project vision, project goals, Sprint iterations, Epics and Features, and Definition of Done will be set. Sprint 0 is considered the Capture Sprint, which includes reviewing materials and establishing the environment, infrastructure, Continuous Integration (CI), Configuration Management (CM), Testing plans, and Scrum schedules. The following actions will be addressed and implemented:

* Establish Communications
* Set up GitHub repo
* Set up Skype channel
* Set up ScrumDo
* Set up intranet document site
* Initiate project: Sprint 0
* Concept
  + - Review OpenFDA data and define concept for application
    - Conduct brainstorming meeting
    - Develop Vision Document
* Development
  + - Study OpenFDA APIs and architecture requirements
    - Develop initial architecture design
    - Develop Architecture Document
* Infrastructure / DevOps
  + - Build out standard Geocent CI environment
    - Develop CI description document
* Develop PMP / Rules of the Road
  + - Identify Branching Strategy
    - Establish Definition of Done
      * Test coverage requirements
      * Check-in processes
      * Code review requirements
    - Establish Deployment process / Configuration Management (CM)
    - Organize Scrum Plan
      * Daily Standups
      * Sprint duration / schedule
      * Backlog grooming schedule

### Establish Resources

Staffing

The staffing plan for Drug IQ will adhere to the details in Table 4 and Attachment A Labor Category Descriptions.pdf.

Training

Table 5 details the training that was identified as necessary for execution of the Drug IQ project.

Table : Training Course Description

|  |  |  |
| --- | --- | --- |
| **Training** | **Description** | **Formal/Informal** |
| AngularJS | Training will be provided to all developer resources by the Front end Designer | Informal |
| Docker | Training will be provided to all developer resources by the DevOps Engineer | Informal |

Facilities

The primary work facility for this effort will be Geocent facilities in New Orleans and Baton Rouge, LA; Stennis Space Center, MS; and Charleston, SC.

Subcontractors

Subcontractors will not be used on this project.

Tools

The tools and technologies that will support the development of Drug IQ are listed in the README.md file located at:

<https://github.com/Geocent/18f-prototype> .

In addition to the tools listed in README.md, we will use ScrumDo and GitHub.

#### Hardware Resources

Table 6 details the hardware resources that will be needed to support the Drug IQ project.

Table : Hardware Resources

| **Product** | **Specifications** | **Quantity** |
| --- | --- | --- |
| Dev Laptops | Mac Book Pro | 8 |
| Other Laptop | Dell | 1 |
| AWS | 3 Mid Servers: CI, Dev, and Prod  ‘ads.geocent.com’  ads-dev.geocent.com  ads-ci.geocent.com | 3 |

## Execution Plan

The execution plan to support Drug IQ will follow Scrum methodology.

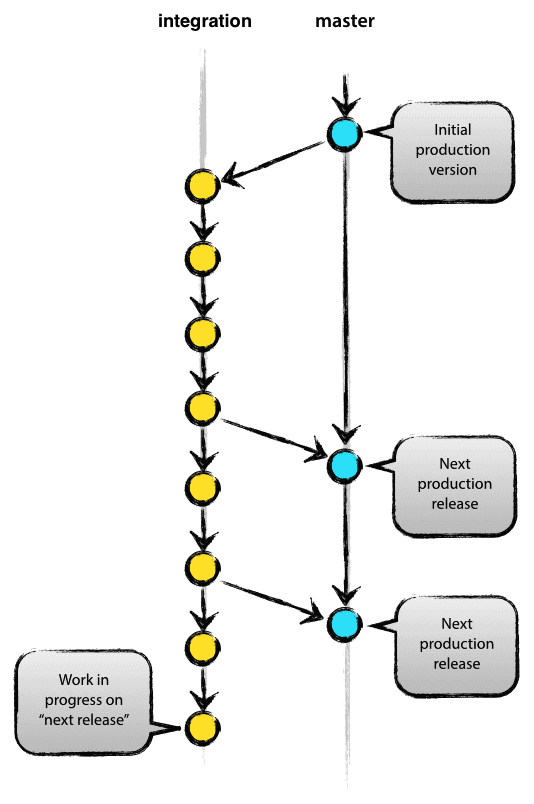
### Manage Code

All code will be managed in GitHub Repository (<https://github.com/Geocent/18f-prototype>). To manage code, Geocent will utilize the branching strategy model documented by Vincent Driessen (<http://nvie.com/posts/a-successful-git-branching-model/>). The branching strategy will consist of

* Main Branch
* Support Branch
* Feature Branch
* Release Branch
* Hotfix Branch

Geocent has employed Mr. Driessen’s branching strategies with much success. These branching strategies will be implemented to support Drug IQ.

#### ****Main Branch****

The central repository will contain two main branches:

* master
* integration

Both branches will have infinite lifetime conditions (Figure 2).

The master branch at origin should be familiar to every Git user. Parallel to the master branch, another branch exists called integration.

We consider origin/master to be the main branch where the source code of HEAD always reflects a production-ready state.

We consider origin/integration to be the main branch where the source code of HEAD always reflects a state with the latest delivered development changes for the next release. Some would call this the “integration branch.” This is where the code for automatic nightly builds are stored.

When the source code in the integration branch reaches a point where it is ready to be released, all of the changes should be merged back into master and tagged with a release number.

Figure : Master/Integration Branch

Therefore, each time when changes are merged back into master, it is considered a new production release. At this point we would use a Git hook script to automatically build and roll-out Drug IQ to production servers every time there is a commit on master.

#### ****Support Branch****

Next to the main branches master and integration, our development model will use a variety of supporting branches. Support branches will be employed to aid with parallel development between team members, for ease in tracking of features, to prepare for production releases, and to assist quickly with fixing live production problems. Unlike the main branches, these branches have a limited life time, because once they are no longer needed they will be removed.

The different types of branches we plan to use are:

* Feature Branches
* Release Branches
* Hotfix Branches

Each of these branches have a specific purpose in terms of being the originating branch and the merge targets. We plan to adhere to the strict rules governing these branches.

#### ****Feature Branches****

Feature branches (Figure 3) can branch off from:

integration

but, must eventually be merged back into:

integration

**In support of Drug IQ, we will use** feature **as the naming convention for Feature branches.**

Feature branches are also referred to as topic branches. These will be used to develop new features for the current or future release. When starting development of a feature, the target release for the feature may not be known. The essence of a feature branch is that it exists as long as the feature is in development. Eventually it will be merge back into the development or discarded, if deemed unnecessary to the final product. Feature branches typically exist in developer repositories only. Collaboration can occur on a feature; however it is never found in the origin.

**Creating a Feature Branch**

When starting work on a new feature, branch off from the integration branch using:

$ git checkout -b myfeature integration

Figure : Feature Branch

Switched to a new branch "myfeature"

This will begin the new feature branch off the main integration branch.

**Incorporating a Finished Feature on Integration**

Finished features will be merged back into the integration branch so it can be added to the upcoming release using:

$ git checkout integration

Switched to branch 'integration'

$ git merge --no-ff myfeature

Updating ea1b82a..05e9557

(Summary of changes)

$ git branch -d myfeature

Deleted branch myfeature (was 05e9557).

$ git push origin integration

The --no-ff flag ensures that the merge will always create a new commit object, even if the merge could be performed with a fast-forward. This will avoid losing information about the historical existence of a feature branch and groups together all commits that together added the feature. This is illustrated in Figure 4.

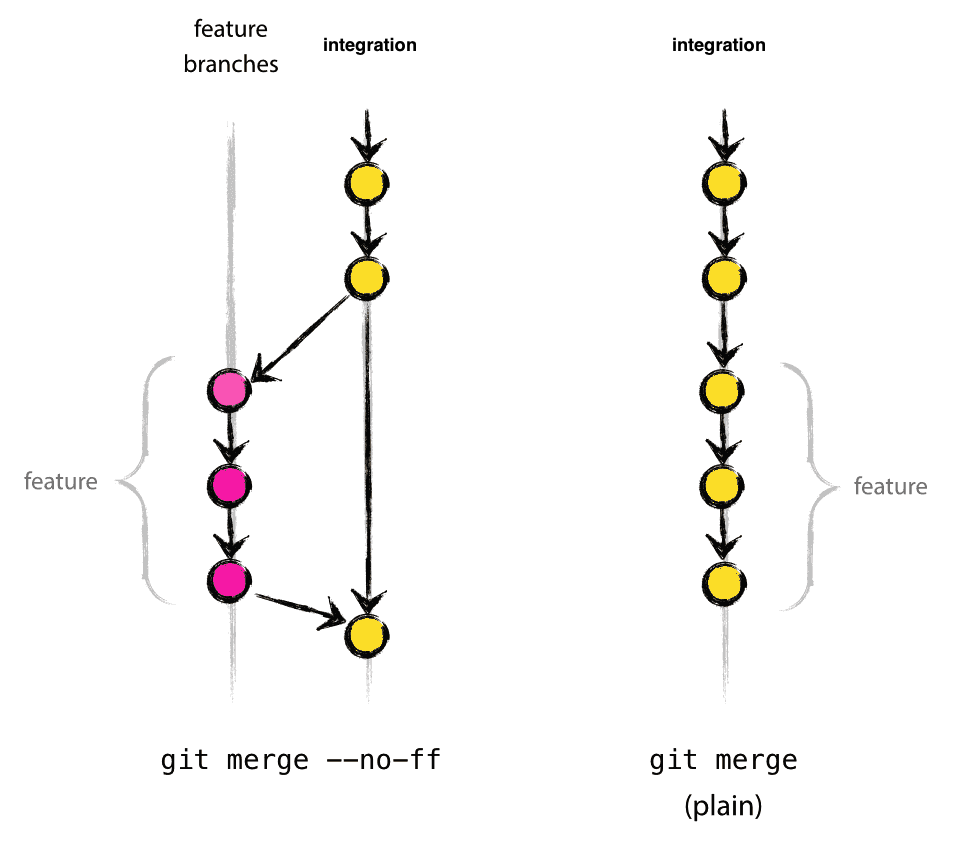


Figure : Feature Branch Incorporated Back to Integration

In the latter case, it is difficult to see from the Git history which of the commit objects together have implemented a feature. The log message would need to be reviewed to understand what occurred. Hence, Geocent, will use the --no-ff flag to easily document feature branch merging back into the integration branch.

#### ****Release Branches****

The release branch will branch off from:

integration

but, must be merge back into:

integration and master

The branch naming convention for release branches is release-\*.

Release branches will support preparation of a new production release. They will allow for last-minute updates, minor bug fixes, and prepares meta-data for the release in terms of version number, build dates, and so forth. By doing all of this work on a release branch, the integration branch will be cleared to receive features for the next release.

The key moment to branch off a new release branch from integration will be when integration reflects the desired state of the new release. This means that all features that are targeted for the release-to-be-built are merged into integration. Future targeted features will not be merged until after the current release branch is branched off.

Versioning releases will occur at the start of a release branch and will be assigned a version number. Once the integration branch reflects changes for the “next release”, the version number value will be set. That decision will be made on the start of the release branch and will be carried out by the project’s rules on version number bumping.

**Creating a Release Branch**

Release branches will be created from the integration branch. For example, if version 1.1.5 is the current production release. The state of integration is ready for the “next release” and we have decided that this will become version 1.2 (rather than 1.1.6 or 2.0). Branch off will occur and the release branch will have a name reflecting the new version number. This will be accomplished using:

$ git checkout -b release-1.2 integration

Switched to a new branch "release-1.2"

$ ./bump-version.sh 1.2

Files modified successfully, version bumped to 1.2.

$ git commit -a -m "Bumped version number to 1.2"

[release-1.2 74d9424] Bumped version number to 1.2

1 files changed, 1 insertions(+), 1 deletions(-)

After creating a new branch and switching to it, the version number will be bumped. We will use the bump-version.sh, a fictional shell script, which will change files in the working copy to reflect the new version. After this, the bumped version number will be committed.

This new branch will exist there for a while, until the release is rolled out definitely. During that time, bug fixes will be applied in this branch (rather than on the integration branch). Adding large new features here will be strictly prohibited. Any new features will need to be merged into integration, and wait for the next big release.

**Finishing a Release Branch**

When the state of the release branch is ready to become a real release, some actions will need to be carried out. First, the release branch will be merged into master. Next, that commit on master will be tagged for easy future reference to this historical version. Finally, the changes made on the release branch will need to be merged back into integration, so that future releases also contain these bug fixes.

The first two steps in Git include:

$ git checkout master

Switched to branch 'master'

$ git merge --no-ff release-1.2

Merge made by recursive.

(Summary of changes)

$ git tag -a 1.2

The release is now done, and tagged for future reference.

To keep the changes made in the release branch, the changes will need to be merged back into integration using:

$ git checkout integration

Switched to branch 'integration'

$ git merge --no-ff release-1.2

Merge made by recursive.

(Summary of changes)

At this point, the release branch will be done and will be removed using:

$ git branch -d release-1.2

Deleted branch release-1.2 (was ff452fe)

#### Hotfix Branches

Hotfix branches will branch off from:

master

but, will need to be merged back into:

integration and master

The naming convention for hotfix branches is hotfix-\*.

Hotfix branches, as shown in Figure 5, are very much like release branches in that they are also meant to prepare for a new production release, albeit unplanned. They arise from the necessity to act immediately upon an undesired state of a live production version. When a critical bug in a production version must be resolved immediately, a hotfix branch will be branched off from the corresponding tag on the master branch that marks the production version. The importance of a hotfix branch is that a team member can still continue working new features on the integration branch; while another team member is preparing a quick production fix.

Figure : Hotfix Branching

**Creating the Hotfix Branch**

Hotfix branches will be created from the master branch. For example, if version 1.2 is the current production release running live and causing problems due to a severe bug and changes on integration are needed, then branching off a hotfix branch to fixing the production problem can occur. Creating the hotfix branch will use:

$ git checkout -b hotfix-1.2.1 master

Switched to a new branch "hotfix-1.2.1"

$ ./bump-version.sh 1.2.1

Files modified successfully, version bumped to 1.2.1.

$ git commit -a -m "Bumped version number to 1.2.1"

[hotfix-1.2.1 41e61bb] Bumped version number to 1.2.1

1 files changed, 1 insertions(+), 1 deletions(-)

After branching off, the version number will need to be bumped in accordance with versioning rules. The bug will be fixed and the commit will occur using:

$ git commit -m "Fixed severe production problem"

[hotfix-1.2.1 abbe5d6] Fixed severe production problem

5 files changed, 32 insertions(+), 17 deletions(-)

**Finishing a Hotfix Branch**

When finished, the bug fix will need to be merged back into master and integration, in order to safeguard that the bug fix will be included in the next release. This is similar to how release branches will be finished.

First, update master and tag the release using:

$ git checkout master

Switched to branch 'master'

$ git merge --no-ff hotfix-1.2.1

Merge made by recursive.

(Summary of changes)

$ git tag -a 1.2.1

Next, include the bug fix in integration using:

$ git checkout integration

Switched to branch 'integration '

$ git merge --no-ff hotfix-1.2.1

Merge made by recursive.

(Summary of changes)

There is one exception to the rule. When the release branch hotfix changes are done, they will need to be merged into that release branch, instead of integration. Back-merging the bug fix into the release branch will eventually result in the bug fix being merged into integration, when the release branch is finished. If work in integration immediately requires this bug fix and cannot wait for the release branch to be finished, we will safely merge the bug fix into integration which will be ready.

This temporary branch will need to be removed using:

$ git branch -d hotfix-1.2.1

Deleted branch hotfix-1.2.1 (was abbe5d6).

This branching strategy will be used to support the Drug IQ project and will be established in GitHub (<https://github.com/Geocent/18f-prototype>).

### Manage Requirements

Requirements will be managed using the practices and tools detailed in Table 7.

Table : Tools and Methods to Manage Requirements

| **Requirements Management Method/Tool** | **Procedure** | **Evidence** | **Description** |
| --- | --- | --- | --- |
| Initial Requirements | Proposal Process | Proposal | Initial requirements and understanding are provided in the proposal and accepted with award. |
| Requirements Understanding | Backlog Grooming | ScrumDo Stories | Requirements will be analyzed, decomposed, clarified, prioritized, and documented into Stories and stored/tracked in Backlog. |
| Requirement Commitment | Sprint Planning | ScrumDo Assignment | During Sprint planning the Product Owner commits to a specific set of user stories. |
| Requirements Change Management | Backlog Grooming | ScrumDo | Requirements will be analyzed, decomposed, clarified, prioritized, and documented by the Product Owner, Scrum Master, and Developers. |
| Requirements Traceability | Backlog Grooming | Requirement for GitHub check-ins to reference ScrumDo Stories | Developers will commit source code by prefixing comment with the ScrumDo Story number, for example, “Story #12: This is my comment.” |
| Requirements Implementation Consistency | Sprint Review | Review tasks in ScrumDo | User Acceptance Tests by Product Owner and Usability Testers accepts Stories implemented in Sprint. |

### Manage Communications

The following communications mechanisms will be used to ensure effective communication between stakeholder management, Geocent senior management, and the Project Team.

#### Reporting Mechanism

During each Sprint Review and Retrospective, the status of the project will be discussed and demonstrated to the 18f ADS Prototype GovernmentTeam and Product Owner. Sprint Planning with the Product Owner, Scrum Master, and Project Team will discusss and identity the Stories to include in each Sprint. Artifacts from Sprint Review and Retrospectives and Sprint Planning will be created and stored in a CM repository that will be accessible to Government and Contractor team members. For Drug IQ, reporting artifacts will be stored in:

<https://github.com/Geocent/18f-prototype/tree/master/docs/scrum>

#### Meetings

In support of the Drug IQ project, Geocent will conduct several meetings to support Scrum as outlined in Table 8.

Table : Scrum Meeting Plan

| **Meeting & Interface Plan** | | |
| --- | --- | --- |
| **Meeting Name & Purpose** | **Attendees** | **Frequency** |
| Project Kick-Off | Project Leads  Team  Geocent Senior Management  Product Owner/Customer | Once |
| Sprint Planning | Product Owner  Scrum Master  Team | 2 days |
| Sprint Review and Retrospective | Product Owner  Scrum Master  Team | 2 days |
| Backlog Grooming | Product Owner  Scrum Master | As needed |
| Daily Stand-up | Product Owner  Scrum Master  Team | Daily |
| Project Closeout/Retrospective | Project Leads  Team  Geocent Senior Management  Product Owner/Customer | End of Project |

### Manage Quality

Geocent will implement Quality Assurance (QA) in order to assure it meets both internal standards and contractual obligations. The Delivery Manager (QA Manager) will oversee all activities include:

* Developer Testing
* Continuous Integration (CI) Testing
* Code Review
* Usability Testing
* User Acceptance Testing

#### Developer Testing

As part of the development effort, developers will create and execute automated tests against their individual code and as part of integrating the code into the main branch. Developer testing will include:

* Unit Testing: Using Karma by executing ‘gulp test’
* Integration Testing: Using Protractor by executing ‘gulp protractor’

#### Continuous Integration (CI) Testing

Developers will execute CI testing on the ‘integration’ and ‘master’ branches. These tests will include:

* Unit Testing: Using Karma by executing ‘gulp test’:   
  <https://ads-ci.geocent.com/job/build-dev/>
* Integration Testing: Using Protractor by executing ‘gulp protractor’:   
  <https://ads-ci.geocent.com/job/functional-dev/>

#### Code Review

Code reviews will be conducted as a task in the development Stories. Peers will conduct code reviews and document the results on the Code Review Form. This level of review will focus on:

* Feature implementation completion
* Unit test coverage
* E2E test review
* CI validation as viewed on Drug IQ product at:

<http://ads-dev.geocent.com>

Code Review Forms supporting each User Story will be located in the Sprint folder associated with the User Story. These will be located at:

<https://github.com/Geocent/18f-prototype/tree/master/docs/scrum>

#### Usability Testing

Usability testing will be performed on UI Wireframe Mockups and on the working prototype after the first working iteration is deployed. The location to view the prototype will be:

<http://ads-dev.geocent.com>

The testing group to support usability testing will consist of medical professions and non-medical professionals. This group will evaluate the UI/UX diagrams prior to coding and test the software through each iteration.

Results of usability testing, including bugs, issues, and change requests (enhancements) will be tracked and mitigated through ScrumDo and GitHub as defined in Section 4.2.5.

#### User Acceptance Testing

User Acceptance Testing (UAT) will be conducted by the Product Owner. During Sprint Review, individual features and results of UAT will be discussed.

### Manage Risks/Issues/Corrective Actions

#### Manage Project Risks

Project risks will be identified during Daily Stand-ups, Sprint Review and Retrospective, and Sprint Planning. Analysis of the risk in terms of schedule and integrity of product will be completed by the developer, Scrum Master, and Project Lead. Risks will be discussed with the Product Owner as soon as they are identified. Risks will be documented in the Sprint Review and Retrospective discussion points.

#### Manage Issues

The Drug IQ project will use GitHub to track all issues including software bugs. Issues relating to user experience (UI/UX) that result in new features (such as enhancement) are recorded in ScrumDo as a new user Story. All issues will be assessed, prioritized, and assigned to a Sprint by the Product Owner.

Resolving issues will involve discussions between the development team, Project Lead, Scrum Master, and Product Owner. The issues will be assessed in terms of impact to project requirement, product operation, project schedule, and priority.

During Stabilization Sprint, all bugs and enhancements will be documented, tracked, and prioritized in GitHub as Issues. This will allow for rapid resolution of bugs and enhancements during stabilization and packaging for delivery.

#### Manage Corrective Action to Closure

The process to resolve bugs/CRs/issues will involve the following:

1. Issues found during UI/UX meetings will be recorded in GitHub Issue Tracker and will include the name of tester, related UX session, and change request.
2. If approved by the Product Owner and Scrum Master, Story/Stories will be created in ScrumDo to support mitigating the issue, prioritized, and identified for a Sprint.
3. As work commences, status will be updated in GitHub Issue Tracker (for issues) and ScrumDo stories (for bugs and change requests). In ScrumDo, stories will reflect To Do, Doing, Testing, and Done.

**Note:** Geocent’s ScrumDo location will be at

<https://www.scrumdo.com/projects/project/18f-ads-prototype/summary>

**Note:** Geocent’s GitHub location will be at:

<https://github.com/Geocent/18f-prototype>

### Manage Measurements (Metrics)

Project measures and metrics will be collected, analyzed, and reported using the measures described in Table 9.

Table : Project Measures

| **Project Measurements and Metrics** | | | | |
| --- | --- | --- | --- | --- |
| **Measure** | **Frequency Collected** | **Collected by Whom** | **Analyzed by Whom** | **Used by Whom** |
| Schedule Adherence | Daily | Project Lead (PL) | Geocent Senior Management | Geocent PL  Product Owner |
| Budget Adherence | Daily | Project Lead | Geocent Senior Management | Geocent PL  Product Owner |
| Funding | Daily | Project Lead | Geocent Senior Management | Geocent PL  Product Owner |

## Work Plan

The work plan for the Drug IQ project will follow Agile/Scrum methodologies and will be documented as User Stories and tasks in ScrumDo. This will be exported after each Sprint to GitHub, located at:

Placeholder for location of export