PROJECT PLANNING

Team Members:
Mark Viola
Mike Petrella
Hyeong Ook Yu
ZiXian Chen
Michael Liu

Table of Contents

1. Product Backlog

1.1 Plugin Support

Priority: High
Cost: 12 Story Points

As Hanno (an administrator), I want plugin API support for the application, where I can create, add, modify plugins written in a programming language that the application can use to extract data from other catalogs.

Tasks:

1. Build/find a class library to make it easier to deal with web requests.
2. Create plugin interface.
3. Create a XML parser that will take a dictionary and output a XML format text, with fields in the same order as dictionary.
4. Create serializer class and add plugin loading functionality
5. Create two default plugins for exoplanet.eu and NASA Exoplanet Archive

---

1.2 Standardizing Units

Priority: High
Cost: 8 Story Points

As Hanno (an administrator), I want each plugin to be able to standardize units of measurements found in its catalogue to one that I specify in the settings of the application.

Tasks:

1. Find/build a unit conversion library
   - Create a way of making general conversion (i.e., specify original unit and convert to whatever is defined in the settings)

1.3 Scheduling a Check for Updates

Priority: High
Cost: 5 Story Points

As Hanno (an administrator), I want the application to check the other exoplanet databases it is tracking, for new updates, on a specified interval (default is daily at 11:59pm).

Tasks:

1. Create a server framework that will run the plugins with threads
2. Have server constantly check system time

1.4 User Permissions

Priority: High
Cost: 5 Story Points

As Hanno (an administrator), I want to use GitHub repo collaborators as a way of distinguishing which user has access to use the application (manage pull-requests, run the server).

Tasks:

1. Create GitHub API management class, and add oauth2 login functionality
2. Add ability to check if user is a collaborator of the repo

---

1.5 Application Making Pull Requests

Priority: High
Cost: 20 Story Points

As Hanno (an administrator), I want the program to make pull requests to the OEC database when it finds changes in the databases that the application is monitoring, using an automated GitHub user account.

Tasks:

1. Add repo management functionality to GitHub API class
2. Add the ability to make pull requests to GitHub API class
3. Integrate GitHub API into a server so that pull-request are made with each new change
4. Add XML file reading to serializer
5. Add the ability to update a XML file with changes to serializer

---

1.6 Accepting/Rejecting Pending Pull Requests

Priority: High
Cost: 5 Story Points

As Irving (a user), I want to be able to accept, reject, or amend pull requests made by the application through a user interface by pressing a button on a pop-up window.

Tasks:

1. Add the ability to accept, reject, and change pull requests
2. Add pull-request to GUI
    - Add viewing
    - Add accepting/denying
    - Add sorting to display
    - Add editing

1.7 Merging Data from Plugins

Priority: High
Cost: 12 Story Points

As Hanno (an administrator), I want the data retrieved by a plugin to be merged with existing data in the Open Exoplanet Catalogue.

Tasks:

1. Write a class that merges two data types together
2. Turn the merged data back into an XML file

1.8 Source of the Update

Priority: Medium
Cost: 2 Story Points

As Hanno (an administrator), I want the website source of the new changes to be a comment in the pull request in the database.

Tasks:

1. Create a field that shows all of the publications related to a specific system. Shown when clicking on the system from the list of altered systems.
2. Make it mandatory for plugins to extract publication data

1.9 See History

Priority: Medium
Cost: 3 Story Points

As Hanno (an administrator), I want to see the history of pull requests, on a user interface.

Tasks:

1. Create a button on the Home screen that allows the user to go into the History screen.
2. Create a History screen that displays a list of the 20 most recent pull requests made by the GitHub bot.

1.10 Edit Pending Pull Requests

Priority: Medium
Cost: 4 Story Points

As Irving (a user), I want to be able to edit a pending pull request through a user interface by pressing an edit button when shown a list of new changes.

Tasks:

1. Create an edit button that opens an interface that allows the user to edit a specified XML file.
2. Update the user's current commit to the one that was edited using the interface.

1.11 History of Updates

Priority: Medium
Cost: 4 Story Points

As Irving (a user), I want to see the planets that have been most recently added/edited by the application in the OEC, using the user interface.

Tasks:

1. Use GitHub API and get the most recent changes
2. Format and parse data from the API and display it in the GUI

1.12 Force a Check for Updates

Priority: Medium
Cost: 3 Story Points

As Hanno (an administrator), I want to be able to force the application to run even if it is not scheduled to check the databases.

Tasks:

1. Create a button in the Home screen that allows the application to check the tracked databases for updates.

1.13 Creating a User Interface

Priority: Medium
Cost: 49 Story Points

As Irving (a user), I want to be able to interact with the application using a user interface.

Tasks:

1. Research UI implementation
2. Create basic UI application
3. Add GitHub login UI
4. Add force check button
5. Add a status checker
6. Add pull request viewer to UI
7. Add plugin management to UI
8. Add settings editor to UI

1.14 Difference Between Updates

Priority: Low
Cost: 25 Story Points

As Irving (a user), I want to be able to click on a newly updated system from a list in the UI and have lines highlighted, showcasing which parts in the XML file have changed.

Tasks:

1. Add repo checking to GUI
    - Add the ability to see recent changes compared to the master branch
        - Create an interactable list of all recent changes
        - Create a GUI that will display changes of the XML files

---

1.15 Settings Form

Priority: Low
Cost: 15 Story Points

As Hanno (an administrator), I want to be able to change time controls, bot account settings, and conversion units, in a settings form in the application.

Tasks:

1. Use C# settings designer to create a settings form and maintain user settings
2. Add timed update controls to user settings
3. Add GitHub bot user account specification to user settings
4. Add a list of possible conversion units to user settings
5. Integrate user settings into the server with on demand changing
6. Add user settings to GUI

1.15 Changes from Deliverable 3

We added a new user story to address the fact that Hanno ultimately wants the data we've pulled to be merged into his catalogue; user story 2.7 is the one that covers this issue. We've also adjusted some of the costs associated with the user stories as we worked more with the actual task. We found some stories were over estimated and some were under estimated. We felt the priorities were accurately estimated and did not need changing.

The stories with changed costs are:
Story 2.1 Plugin Support – 12 -> 14
- We forgot to account for the learning of the API for each site and this caused us to underestimate the task

Story 2.3 Unit Conversion – 8 -> 3
- We found the task was a lot easier than we initially thought, so the time to implement it was reduced by a lot.

Story 2.5 Application Making Pull Requests – 20 -> 18
- After finding the Octokit library provided by GitHub it significantly reduced the amount of work needed for this task, however when accounting for the time it took to learn the library it only reduced the total cost by 2 points.

2. Release Plan

Sprint 1:

- As Hanno (an administrator), I want plugin API support for the application, where I can create, add, modify plugins written in a programing language that the application can use to extract data from other catalogs.
- As Hanno (an administrator), I want the application to check other exoplanet databases for new changes on a specified interval (default is daily at 11:59pm).

Estimate: 19 Story Points

Sprint 2:

- As Hanno (an administrator), I want each to be able to standardize units of measurements found in its catalogue to one that I specify in the settings of the application. (**Finish remaining**)
- As Hanno (an administrator), I want to use GitHub repo collaborators as a way of distinguishing which user has access to use the application (manage pull-requests, run the server).
- As Hanno (an administrator), I want the program to make pull requests to the OEC database when it finds changes in the databases that the application is monitoring, using an automated GitHub user account.
- As Hanno (an administrator), I want the website source of the new changes to be a comment in the pull-request in the database.

Estimate: 30 Story Points

Sprint 3:

- As Irving (a user), I want to be able to interact with the application using a user interface. (**Half**)
- As Irving (a user), I want to be able to click on a newly updated system from a list in the UI and have lines highlighted, showcasing which parts in the XML file have changed.

Estimate: 25 Story Points

Sprint 4:

- As Irving (a user), I want to see the history of pull requests, on a user interface
- As Irving (a user), I want to be able to click on a newly updated system from a list in the UI and have lines highlighted, showcasing which parts in the XML file have changed.

Estimate: 25 Story Points

Sprint 5:

- As Hanno (an administrator), I want to the data retrieved by a plugin to be merged with existing data in the Open Exoplanet Catalogue.
- As Hanno (an administrator), I want to be able to force the application to run even if it is not scheduled to check the databases.
- As Irving (a user), I want to see the history of pull requests, on a user interface
- As Irving (a user), I want to see the planets that have been most recently added/edited by the application in the OEC, using the user interface.
- As Irving (a user), I want to be able to accept, reject, or amend pull requests made by the application through a user interface by pressing a button on a pop-up window.

Estimate: 31 Story Points

Sprint 6:

- As Irving (a user), I want to be able to edit a pending pull request through a user interface by pressing an edit button when shown a list of new changes.
- As Hanno (an administrator), I want to be able to change time controls, bot account settings, conversion units in a settings form in the application.
- Finish leftovers and testing

Estimate: 20 Story Points

3. Sprint Plan

3.1 Sprint Backlog (Sprint #2)

For the first sprint (10/24/2016 – 10/30/2016), the user stories that will be completed are the users stories found in 2.2, 2.4, 2.5, 2.7. This sprint will have a sprint velocity of 25 story points and will be 7 days long, starting on the Monday of this week. The developers at Spazio have estimated that they can each complete 5 story points per week. The following, details the user stories being completed and their corresponding tasks.

3.1.1 Standardizing Units

Priority: High
Cost: 8 Story Points

As Hanno (an administrator), I want each plugin to be able to standardize units of measurements found in its catalogue to one that I specify in the settings of the application.

Completed Task. Find/build a unit conversion library (5 story points)
Task 1. Create a way of making general conversion (i.e., specify original unit and convert to whatever is defined in the settings) **(3 story points)**

3.1.2 User Permissions

Priority: High
Cost: 2 Story Points

As Hanno (an administrator), I want to use GitHub repo collaborators as a way of distinguishing which user has access to use the application (manage pull-requests, run the server).

Task 2. Create GitHub API management class, and add oauth2 login functionality (add ability to check if user is a collaborator of the repo). **(2 story points)**

3.1.3 Application Making Pull Requests

Priority: High
Cost: 18 Story Points

As Hanno (an administrator), I want the program to make pull requests to the OEC database when it finds changes in the databases that the application is monitoring, using an automated GitHub user account.

Task 3. Add repo management functionality to GitHub API class **(5 story points)**
Task 4. Add the ability to make pull requests to GitHub API class **(4 story points)**
Task 5. Integrate GitHub API into a server so that pull-request are made with each new change **(4 story points)**
Task 6. Add XML file reading to serializer **(3 story points)**
Task 7. Add the ability to update a XML file with changes to serializer **(2 story points)**

3.1.4 Source of the Update

Priority: High
Cost: 2 Story Points

As Hanno (an administrator), I want the website source of the new changes to be a comment in the pull request in the database.

Task 8. Make it mandatory for plugins to extract publication data **(2 story points)**

Detailed Plan

	Oct 24	Oct 25	Oct 26	Oct 27	Oct 28	Oct 29	Oct 30
Task 1		2					
Task 2			5				
Task 3			3				
Task 4				2			
Task 5				2			
Task 6					3		
Task 7						1 + 2	
Task 8						2 + 3	

Actual

	Oct 24	Oct 25	Oct 26	Oct 27	Oct 28	Oct 29	Oct 30
Task 1		2	2				
Task 2							5
Task 3							
Task 4							2
Task 5		2					
Task 6		4					
Task 7			2				
Task 8							

Legend:

Mark Viola
Mike Petrella
Hyeong Ook Yu
ZiXian Chen
Michael Liu

3.2 Sprint Backlog (Sprint #3)

For the first sprint (10/31/2016 – 11/6/2016), the user stories that will be completed are the users stories found in 2.12, and 2.13. This sprint will have a sprint velocity of 25 story points and will be 7 days long, starting on the Monday of this week. The developers at Spazio have estimated that they can each complete 5 story points per week. The following, details the user stories being completed and their corresponding tasks.

3.2.1 Difference Between Updates – 2.13

Priority: Low
Cost: 25 Story Points

As Irving (a user), I want to be able to click on a newly updated system from a list in the UI and have lines highlighted, showcasing which parts in the XML file have changed.

Task 1. Add the ability to see recent changes compared to the master branch **(10 story points)**
Future Task. Create an interactable list of all recent changes (6 story points)
Future Task. Create a GUI that will display changes of the XML files (9 story points)

3.2.2 Creating a User Interface – 2.12

Priority: Medium
Cost: 35 Story Points

As Irving (a user), I want to be able to interact with the application using a user interface.

Task 2. Create basic UI application **(4 story points)**
Task 3. Add GitHub login UI **(3 story points)**
Task 4. Bot management GUI (add force check button and status checker) **(3 story points)**
Task 5. Add plugin management to UI **(5 story points)**
Task 6. Add pull request viewer to UI (14 story points)
Future Task. Add settings editor to UI (6 story points)

Detailed Plan

	Oct 31	Nov 1	Nov 2	Nov 3	Nov 4	Nov 5	Nov 6
Task 1		2 + 2	3 + 3				
Task 2			2 + 2				
Task 3				3			
Task 4				3			
Task 5					5		
Task 6							

Actual

	Oct 31	Nov 1	Nov 2	Nov 3	Nov 4	Nov 5	Nov 6
Task 1				3			
Task 2							4
Task 3		1					
Task 4							
Task 5						5	
Task 6				3			

Legend:

Mark Viola
Mike Petrella
Hyeong Ook Yu
ZiXian Chen
Michael Liu

3.3 Sprint Backlog (Sprint #4)

For the first sprint (11/7/2016 – 11/13/2016), the user stories that will be completed are the users stories found in 2.12, and 2.13. This sprint will have a sprint velocity of 25 story points and will be 7 days long, starting on the Monday of this week. The developers at Spazio have estimated that they can each complete 5 story points per week. The following, details the user stories being completed and their corresponding tasks.

3.3.1 Difference Between Updates – 2.13

Priority: Low
Cost: 25 Story Points

As Irving (a user), I want to be able to click on a newly updated system from a list in the UI and have lines highlighted, showcasing which parts in the XML file have changed.

Task 1. Add the ability to see recent changes compared to the master branch **(10 story points)**
Future Task. Create an interactable list of all recent changes (6 story points)
Future Task. Create a GUI that will display changes of the XML files (9 story points)

3.3.2 Creating a User Interface – 2.12

Priority: Medium
Cost: 28 Story Points

As Irving (a user), I want to be able to interact with the application using a user interface.

Completed Task. Create basic UI application (4 story points)
Completed Task. Add GitHub login UI (3 story points)
Task 2. Bot management GUI (add force check button and status checker) **(3 story points)**
Task 3. Add plugin management to UI **(5 story points)**
Task 4. Add pull request viewer to UI **(7 story points)**
Future Task. Add settings editor to UI (6 story points)

Detailed Plan

	Nov 7	Nov 8	Nov 9	Nov 10	Nov 11	Nov 12	Nov 13
Task 1			5 + 5				
Task 2			3				
Task 3				5			
Task 4					5 + 2		

Actual

	Nov 7	Nov 8	Nov 9	Nov 10	Nov 11	Nov 12	Nov 13
Task 1							
Task 2							
Task 3							
Task 4							

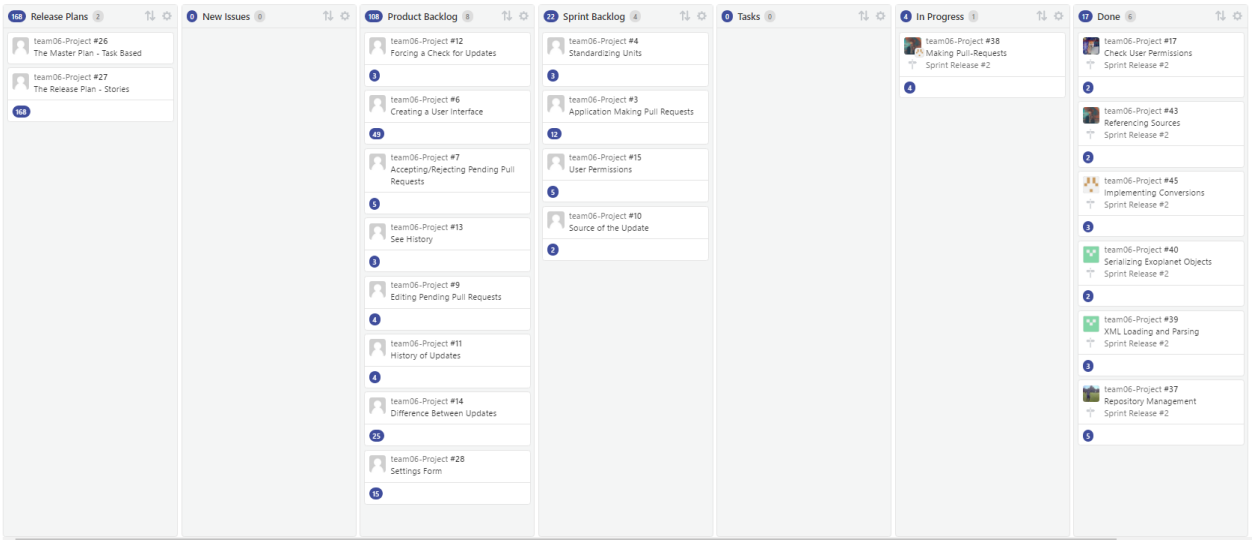
Legend:

Mark Viola
Mike Petrella
Hyeong Ook Yu
ZiXian Chen
Michael Liu

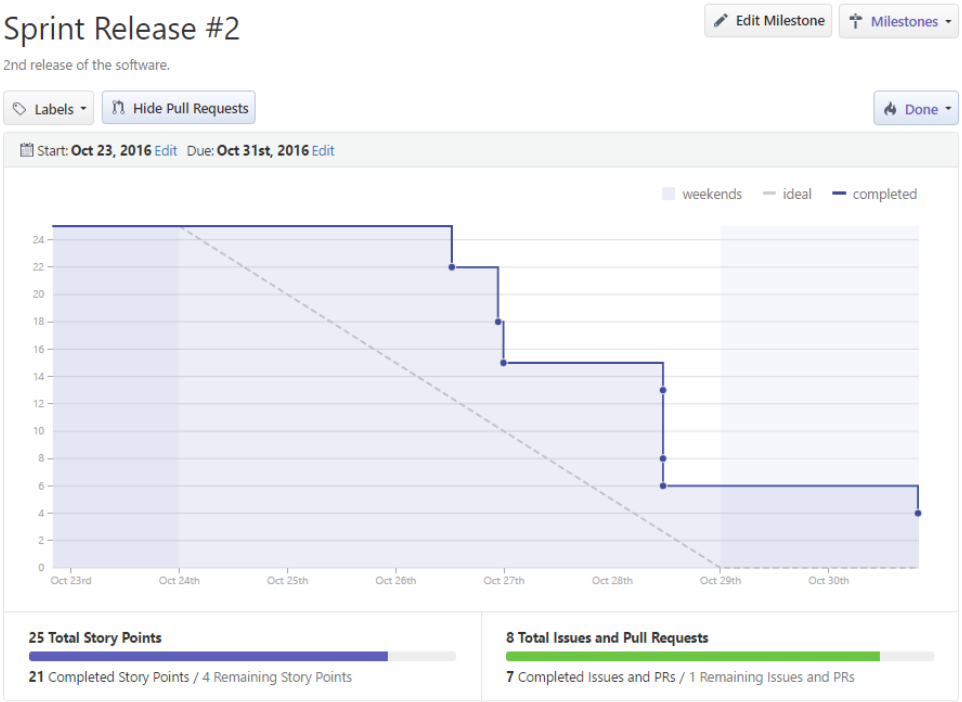
4. Sprint Results

4.1 Sprint #2 Results

4.1.1 Task Board



4.1.2 Burndown Chart

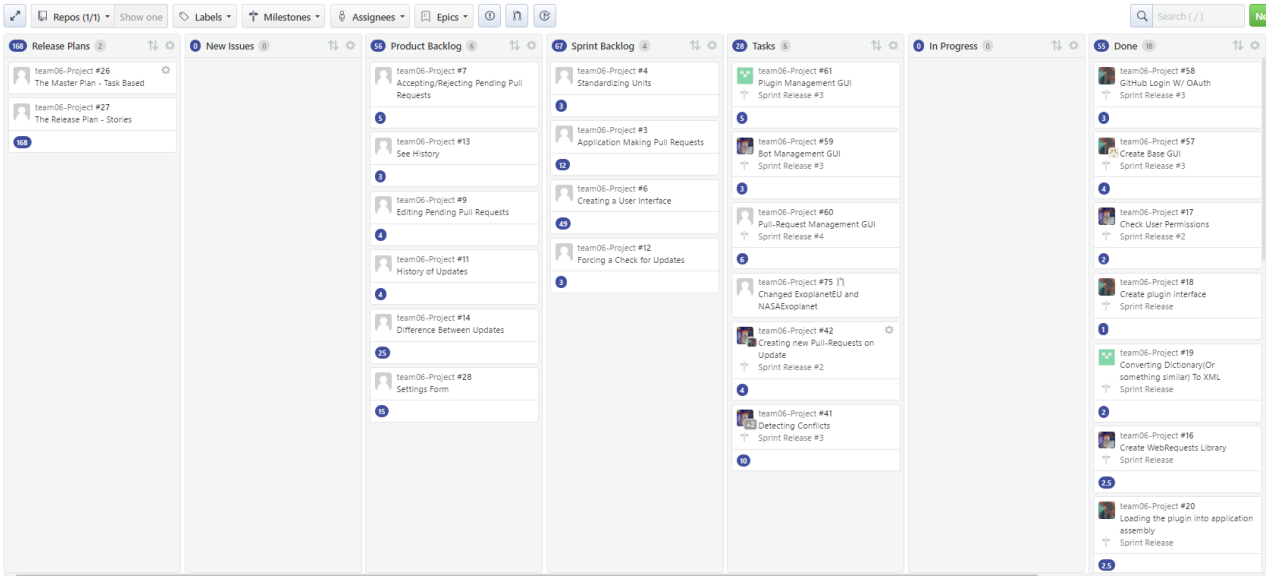


4.1.3 Project Velocity

Projected project velocity: 25 story points
Actual project velocity: 21 story points

4.2 Sprint #3 Results

4.2.1 Task Board



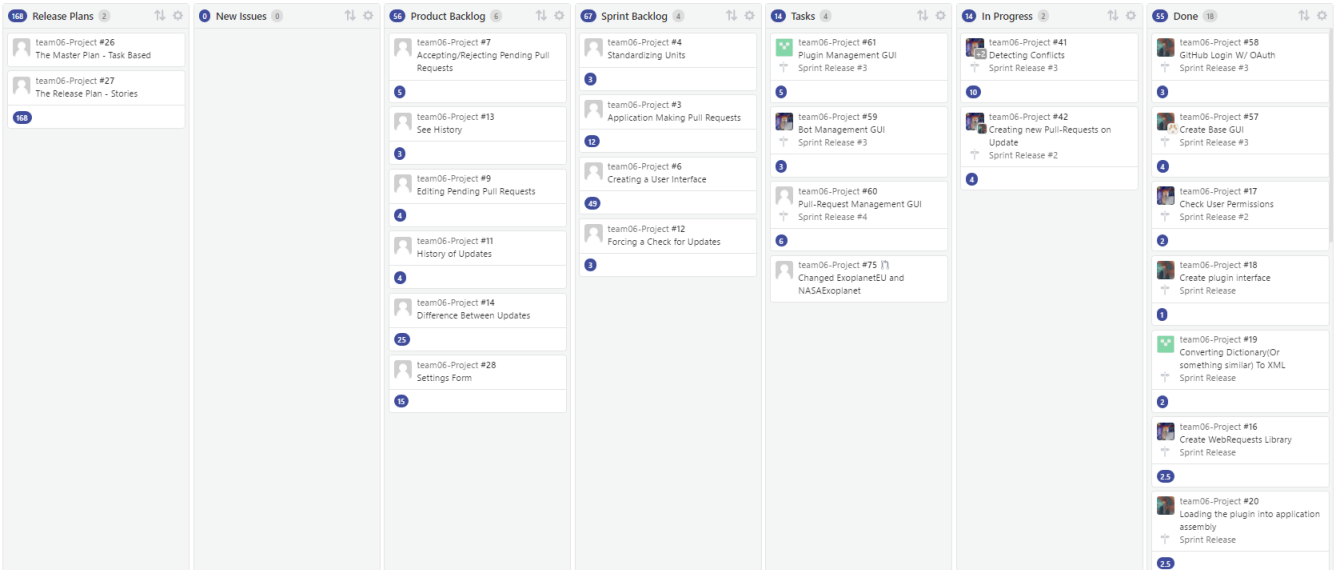
4.2.2 Burndown Chart

4.2.3 Project Velocity

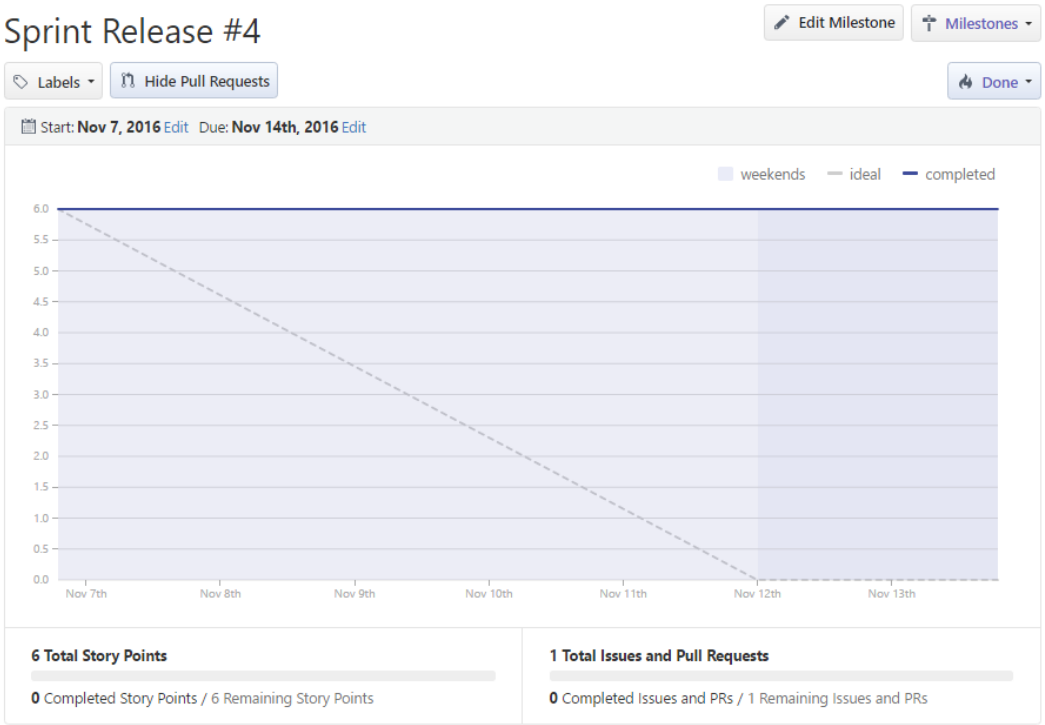
Projected project velocity: 25 story points
Actual project velocity: 7 story points

4.3 Sprint #4 Results

4.3.1 Task Board



4.3.2 Burndown Chart



4.3.3 Project Velocity

Projected project velocity: 25 story points
Actual project velocity: 0 story points

5. System Design

5.1 Session

Manages user GitHub session. A wrapper for Octokit GitHub client, handles authorization.

5.2 Repository Manager

Manages all repository activities for a specified repo in addition of a Session. Wrapper for Octokit repository clients, deals with functions such as adding/updating files, managing branches, commits, and pull requests. Uses session as authorization for GitHub API.

5.3 Serializer

Handles plugin loading.

5.4 OECBot

Functional automation of OEC database updates. Schedule checks for plugins, and creates new updates for the OEC database, through branching, commits, and pull requests. Also uses Session to deal with GitHub API authorization. Uses PlanetMerger to handle merging of systems before committing it to OEC. Uses XMLDeserializer to parse xml files from OEC so that it may be used for comparisons. Uses serializer to get plugins.

5.5 Plugin Interface

Interface for creation and usage of plugins. Plugins are used to extract data from exoplanet databases.

5.6 CallBackServer

Utility class. Used for Oauth authorization callback, in order to obtain authorization codes. Creates Session object from GitHub callback.

5.7 HttpRequests

Utility class. Used to facilitate http requests and get back information. Plugins use this to create http requests to extract data from databases.

5.8 PlanetMerger

Utility class. Used to merge exoplanet systems from different sources.

5.9 XMLDeserializer

Utility class. Used to create systems objects from xml input.

5.10 System Design Changes

One change we made was separating GitHub API client into two different components, one just for authorization handling and one for repository managing. This way everything is compartmentalized and each class has a dedicated usage and functionality. Also this way authorization can be independent of application's main functionality.

Added PlanetMerger, because it is important to be able to get the most accurate information from all possible sources without loss of information, as well as making it so that there are less potential conflicts from multiple commits.

CallBackServer was added because there needed to be a way of facilitating Oauth authorization with GitHub, this way users do not need to store their password in the application and is generally more secure overall.

6. Verification

To verify that the application was meeting the specifications of Spazio's requirements, unit tests were created. The following unit tests were created:

- ExoplanetEUUnitTest
  - Unit tests for the ExoplanetEU class
- GitHubUnitTests
  - Unit tests for RepositoryManager and Session
- HTTPRequestsUnitTests
  - Unit tests for HTTPRquests
- MeasurementDSUnitTests
  - Unit tests for Measurement, MeasurementUnit, NumberErrorMeasurement, NumberMeasurement, StringMeasurement
- NASAExoplanetUnitTests
  - Unit tests for NASAExoplanetPlugin
- SerializerUnitTests
  - Unit tests for Serializer and XMLDeserializer
- StellarObjectDSUnitTests
  - Unit tests for Binary, Planet, SolarSystem, Star, and StellarObject

After running the unit tests, the following results were obtained:

```
Runtime Environment
   OS Version: Microsoft Windows NT 10.0.14393.0
  CLR Version: 4.0.30319.42000

Test Files
    OECUpdater/UnitTests/bin/Debug/UnitTests.dll

=> UnitTests.GitHubUnitTests.UpdateFileTest
Octokit.RepositoryContentInfo

Run Settings
    DisposeRunners: True
    WorkDirectory: C:\Users\Mark\Desktop\PROJECT1\team06-Project
    ImageRuntimeVersion: 4.0.30319
    ImageTargetFrameworkName: .NETFramework,Version=v4.5
    ImageRequiresX86: False
    ImageRequiresDefaultAppDomainAssemblyResolver: False
    NumberOfTestWorkers: 4

Test Run Summary
  Overall result: Passed
  Test Count: 65, Passed: 65, Failed: 0, Inconclusive: 0, Skipped: 0
  Start time: 2016-11-14 05:44:27Z
    End time: 2016-11-14 05:44:33Z
    Duration: 5.911 seconds

Results (nunit3) saved as TestResult.xml
```

65 out of 65 tests were successful which means that the software Spazio has created meets their requirements.