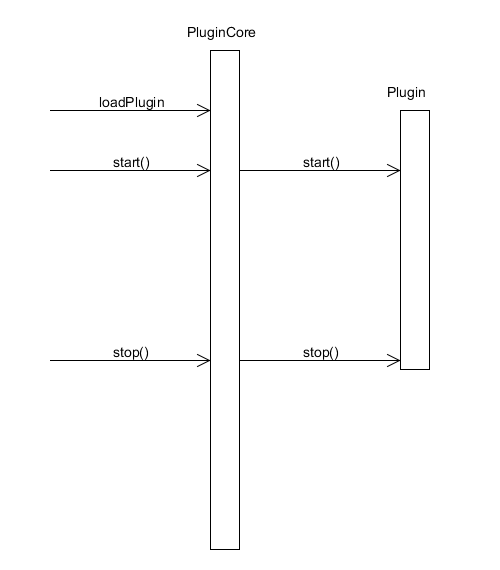
Brahma Stew Design Document

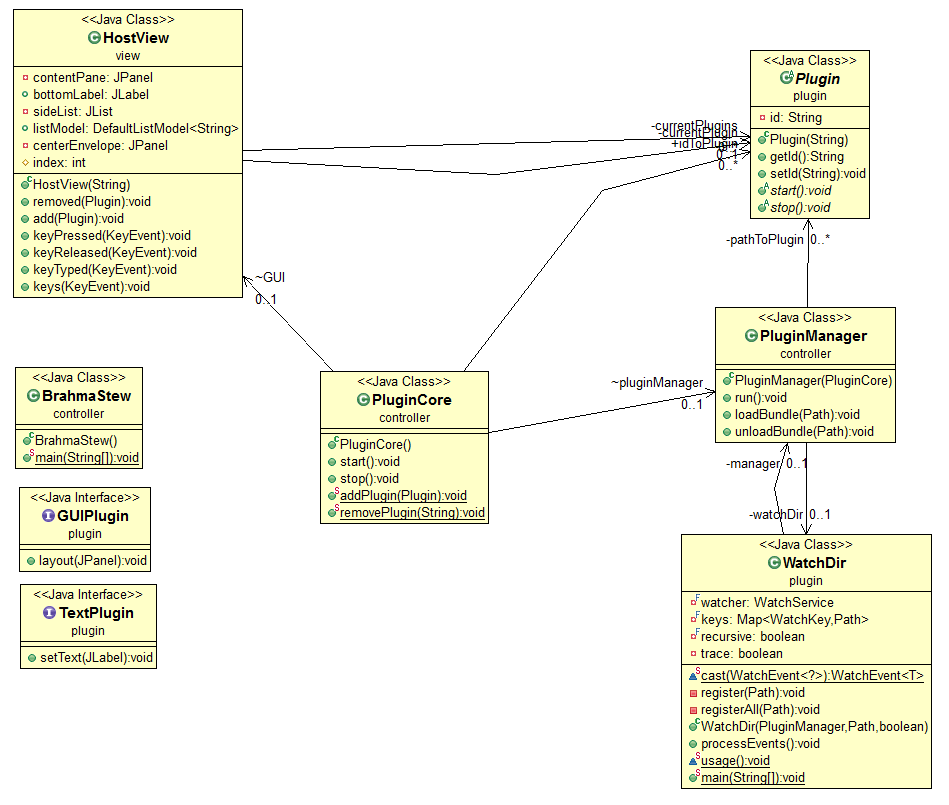
**Richard Thai**

**Brandon Knight**

Interaction Diagram



Dependency + Class Diagram



Test Specifications

**JUnit Testing**

Currently, a simple case exists that tests the model and ensures the ID is set.

**Visual Testing**

When a plug in is loaded, we expect the GUI to pronounce that “-plug in- has loaded”.

We expect if a plugin implements layout() that it will display in the center panel

We expect if a plugin implements setText() that it will put text in the bottomLabel

We expect if a plugin does not implement either method, that they will run in the background with a default bottomLabel of “This plugin is running.”

Work Allocation Table / Daily Scrums

It was decided early on that creating a work allocation table would be redundant if our ‘daily’ scrum meetings were being tracked. The work allocated would be the work completed.

**21 September 2012**

Reworked the module decomposition diagram, but the default manifest for the two extensions don’t seem to work. We’re going to try to fix this tomorrow because the error message is not particularly helpful. After fixing this issue, we’ll be able to start to try reading the dependencies between plugins.

**21 September 2012**

Some progress on figuring out how to hook the plugin to the host application

**20 September 2012**

Work on class skeleton started

**19 September 2012**

Repository initialized on GitHub

**24 October 2012**

Worked on getting our project to load plug-ins that could utilize the bottom text label as an extension point.

**26 September 2012**

Worked on getting our project to be able to utilize plug-ins that have no visual or text layout methods. Just a plug-in that runs in the background.

**2 October 2012**

Our project can successfully run plug ins that either use text, layout, or neither!

**3 October 2012**

Created a sound plug in to test a plug in running in the background. Doesn’t quite work but proof of concept.

**5 October 2012**

JUnit test case written to ensure model ID’s are registering. Visual testing is used to verify the GUI. Final design document compiled.

Quality Attributes

This portion of the document lists the quality attributes supported by our project.

**Extensibility**: The application contains public extension points for other plugins to utilize. In addition, sample applications are also included as an example for other users.

**Modifiability**: The host application is split into a MVC architecture to make it easy to learn and change without affecting too much other code.

**Testability**: All of the interfaces that a plugin can affect is shown on the screen of the application. The Plugin model is a public interface that can be tested.

**Usability**: The application contains a limited amount of interfaces and no menus to click through.

**Fault Tolerance**: The host application is able to handle changes to the plugin directory as the application is running.