Software Design Document

for

Weather RSS: A Map Based RSS Reader for Weather Alerts

Team: Nerds INC.

Project: Weather RSS

Team Members:
Sandra Mondragon
Drew Warner

Last Updated: [4/26/2016 4:04:20 PM]

Table of Contents

Document Revision History

Revision Number	Revision Date	Description	Rationale
0.1	4/12/2016	All Sections	1st draft
0.2	4/13/2016	Section 2	Include diagrams
0.3	4/14/2016	Written Sections	Write-ups
0.4	4/15/2016	Finishing Written Sections	Finishing up written sections

0.5	4/25/2016	Added ball and socket diagram	Previous had not been added

List of Figures

List of Tables

1. Introduction

This document contains the design pattern for the Weather RSS Alert System, including the architectural design goals, software architecture and subsystem services. The Weather RSS Alert System parses RSS feeds from the Internet and returns weather alerts in selected regions, severity, or time.

1.1 Architectural Design Goals

The performance goals for the Weather RSS Alert System are that the system can reliably return feeds in a timely manner. Although it would vary based on user Internet connection speeds, a reasonable time that the system return and display feeds would be within ten seconds. As for reaction of the software itself, there should be less than a second of latency between mouse clicks and reactions on the screen, and it should not leak memory and eventually eat up all resources.

The end user criteria is that it should be relatively easy to use and navigate. The user should be prompted to and easily update or patch the software if necessary, and should be able to easily remove the software if they so wish. The GUI should be easy to understand and intuitive, and the user should have little to no trouble navigating it.

1.2 Scope

Section 2

Provides the software architectural pattern chosen to demonstrate the overall structure of the Weather RSS Alert System. Along with a brief description of each subsystem package and its components, class diagrams for each significant class contained within the subsystem package are included.

The system requirements and software components required for the Weather RSS Alert application is discussed in this section, as well as, the persistent data stored by the system and the data storage and management strategy used to store and manage that data.

Section 3

Provides a description of the services provided by each subsystem including dependencies and responsibilities. This includes services that each subsystem provides for and requires from other subsystems. A UML component diagram is included for each component to demonstrate subsystem dependencies.

2. Software Architecture

2.1 Overview

The architectural pattern used when designing the WeatherRSS Alert System is the Model-Vew-Controller (MVC) pattern. This pattern was used because it seemed best suited to fit the needs the software must fulfill. The user will click on and type in different input in order to manipulate the model, in this case the map and the feeds displayed, and then the model will update the view, that is, the user interface, to give the user an updated account of what the user wishes to see. The cycle then continues as much and as often as the user wishes. This is almost the exact definition of the MVC pattern, and thus it was used as the design for this system.

Name	**Summary**
FeedManager	Core of the Weather RSS Alert application that accesses and modifies feeds. Responds to queries made by MapViewer and notifies it of any view changes received by SettingsManager.
MapViewer	Renders the appearance of the feeds by geographic location.
SettingsManager	Accepts and notifies FeedManager of changes made by the user through UserInterface.
UserInterface	Translates user's actions, to the operation that needs to be performed, to UserInterface.

Table 2-1 Weather RSS Alert MVC Architecture Description

There are four main subsystems in the WeatherRSS Alert System. The first two are both a part of the controller that the user uses. The first is the settings manager, where the user may manage the various settings related to the feeds and alerts. The user may manage their alerts, their feed list, and manage the alert frequency. The second is the user interface, where the user may sort their alerts, set their select the

URL of the feed, and expand the pin by clicking on it. The third is the feed manager, which downloads, parses, and manages the feeds. This controller will manipulate the model and change the display on the

map viewer, which is the fourth subsystem. Once the controller has manipulated the model, the model updates the map viewer and the updated map is returned to the user.

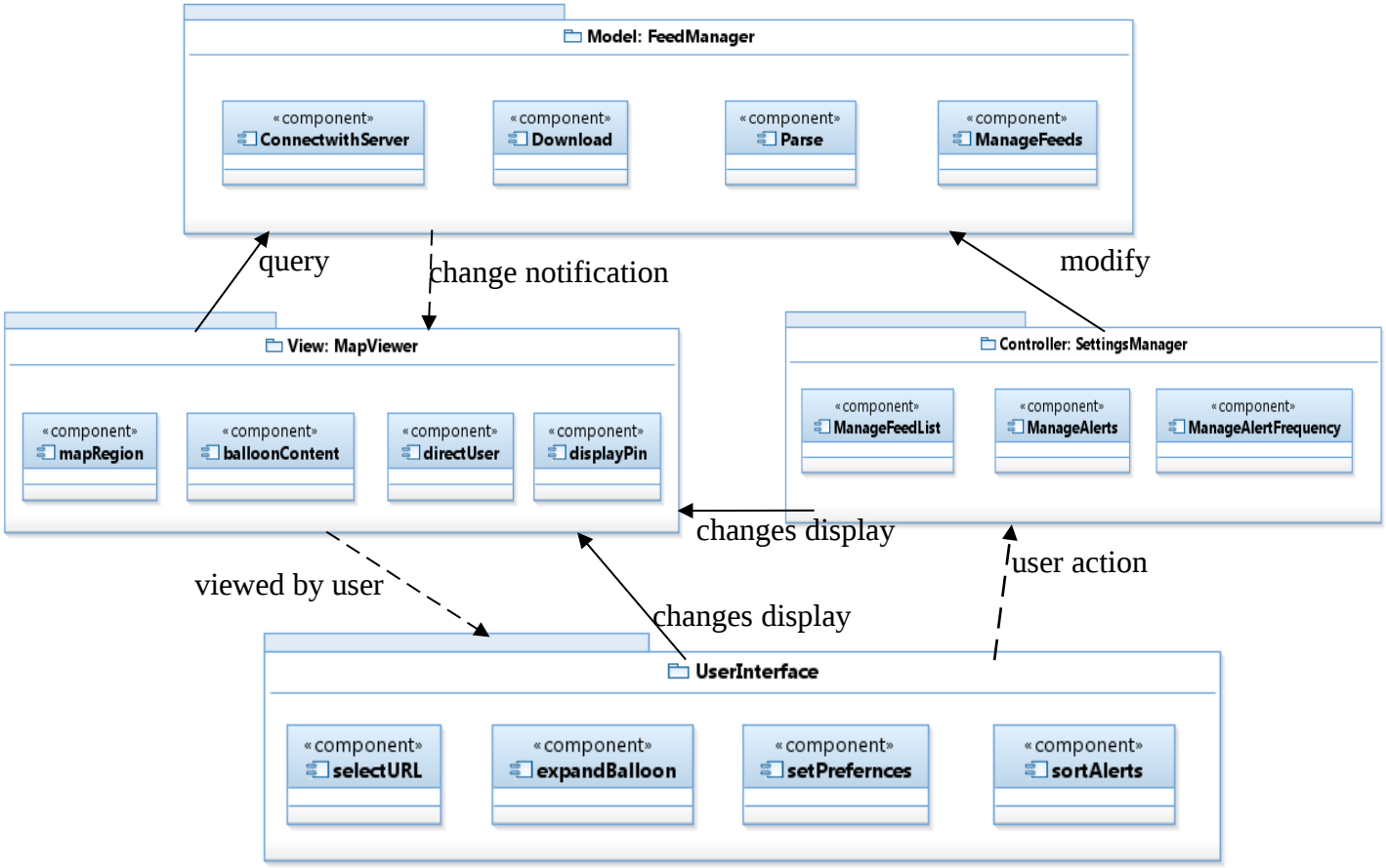


Figure 2-1 Weather RSS Alert MVC Architecture Diagram

2.2 Subsystem Decomposition

2.2.1 FeedManager Package

This subsystem interacts with the server and handles RSS feeds from http://alerts.weather.gov. This includes, directing users to the website when they select a feed's URL.

Name	Summary
ConnectWithServer	Connects with http://weather.gov to fetch and download feeds. Verifies users internet connection.
Download	Downloads user specified feeds.
Parse	Parses the RSS XML files by severity, location, and event type.
ManageFeeds	Contains list of parsed feeds.

2.2.2 MapViewer Package

This subsystem handles the GUI portion of the Weather RSS system. It displays feeds based on user preferences and feed information, such as, alert summary, URL, and location.

Name	Summary
mapRegion	Responsible for locating the alert location, given the latitude longitude coordinates.
balloonContent	Sends alert information to be shown inside balloon.
directUser	Notifies and provides URL to the model to connect with the weather.gov server.
displayPin	Displays a pin on the alert location.

2.2.3 SettingsManager Package

This subsystem accepts user input from the UserInterface and communicates those changes with the MapViewer. It also sends any changes that need to be saved to FeedManager.

Name	Summary
ManageFeedList	Allows user to add, delete, or view a feed in the feed list. Any changes are saved to a file.
ManageAlerts	Allows user to add, delete, or view weather alerts. Any changes are saved to a file.

ManageAlertFrequency	Allows user to set, edit, or remove an alert frequency. Any changes are saved to a file.

2.2.4 UserInterface Package

This subsystem handles user input from the user's keyboard and communicates any request, such as sorting alerts by location, to the SettingsManager.

Name	Summary
selectURL	Notifies and provides MapViewer with a link if a user selects a URL.
expandBalloon	Notifies MapViewer to display balloon with feed summary.
setPreferences	Accepts user input, such as feed and alert management, and sends it to SettingsManager to be saved.
sortAlerts	Notifies MapViewer to display alerts by location, time or severity.

2.3 Hardware/Software Mapping

The user will need to have a Windows or Linux operating system in order to run the WeatherRSS Alert System. In the future, the software may be ported to other devices, such as phones, but for now, it is only available on those operating systems. A screen and a mouse/track pad is necessary in order to interact with the software, and access to the Internet is required to receive feed data and alerts.

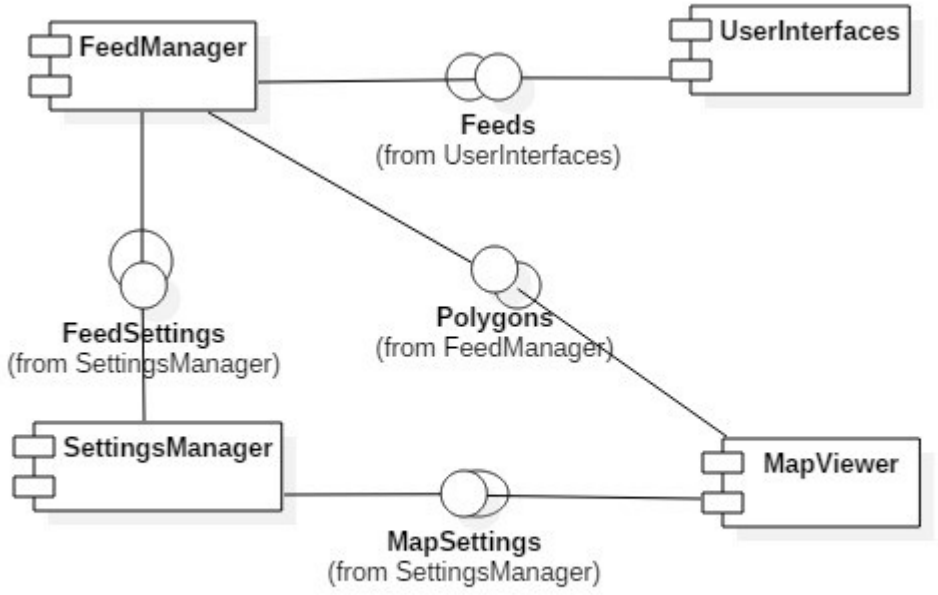
2.4 Persistent Data Storage and Management

The persistent data that is stored by the software includes mostly the user settings. The amount of time before feeds and alerts are updated is an example. Another example is how feeds are sorted, whether they be sorted by name, severity, region, or time. Another point of data stored is how long to display an inactive feed. The feeds that were displayed before system shut down, if the software was closed properly, will also be saved.

This persistent data will be stored on a file system on the user's device. A database storage system would be impractical, as the amount of data stored is small, and there is little to no harm done if data is lost. Moreover, since settings and viewed feeds are personal data items, a file system storage works well, as there is no risk of concurrent access to the data, thus preventing the pitfall file systems have where they are locked while a user is accessing them. Another advantage the file system has is that it has less of a cost associated with it. A disadvantage of the file system strategy would be that it does not ever create a

snapshot of the data for backup purposes, so data could be lost. However, as stated above, the settings and feeds are can easily be redone, so that disadvantage is quite minimal.

3. Subsystem Services



The feed manager depends on the input from the user from the user interface. The map viewer depends on the feed manager to put pins on the map, the settings manager depends on nothing, although one could argue that the map viewer and feed manager, while not depending on the settings manager, do rely on it somewhat for certain information.

The feed manager should provide that map viewer with the locations and data information of the feeds. This could be called provideFeedInfo and would send over feed information to the map viewer. The map would then update with information from the feed.

The user interface should provide the feed manager with input as to which types of feeds, regions of feeds, severity of feeds, or time of feeds should be displayed. This could be called modifyFeed, as this would entail adding and removing feeds from a list.

The feed manager, while it does not rely on it, does receive some information from the settings manager. This could be called something along the lines of exportSettings. It would, of course, export the list of feeds to display to the feed manager, which would then forward the list to the map viewer. It would also have settings for how long to display feeds and alerts, and when to remove them from the list.