Database Management System for Monitoring the Effect of Climate on Wildlife Interactions
Final Project Report

Final Requirements, Design, & Implementation/Testing

NCSU College of Veterinary Medicine
Dr. Suzanne Kennedy-Stoskopf
Dr. Lauren Charles-Smith

CSC 492 Team 12:

Bo Chulindra
Chris King
Scott Gentry

North Carolina State University
Department of Computer Science
Senior Design Center

December 13, 2011

I. Sponsor Background

Dr. Suzanne Kennedy-Stoskopf and Dr. Lauren Charles-Smith of the NCSU College of Veterinary Medicine are conducting research in order to determine the effect climate has on wildlife infectious diseases. They currently have collars on various wildlife in order to collect data such as the animal's locations throughout the day, their home range, the time spent in proximity to other collars, and more. This data is also combined with topographical maps and weather station data to give a more detailed picture of the collar's environment.

- Dr. Suzanne Kennedy-Stoskopf, Research Professor of Wildlife Infectious Diseases
  suzanne_stoskopf@ncsu.edu

- Dr. Lauren Charles-Smith, Fisheries, Wildlife, and Conservation Biology
  drlaurencharles@gmail.com

II. Problem Statement

The sponsors are collecting data from animal collars and weather stations, as well as maps.

This data presents the following problems:

- **The researchers don't have a way of automatically linking data together.** For example, interaction data, or data of where and for how long two collars have been within a certain range of each other, is important to the researchers. The researchers don't have a system that calculates interaction data.
- **The researchers can't share data with other researchers very well.** Data is retrieved from the collars in one format, and then may be saved in some other format. Because of this, two researchers working with different equipment will often have their data in different formats.
- **The researchers want to see their data visualized in ArcGIS.** In order to get their data in ArcGIS correctly, the researchers have to use some tool, or create on themselves, that imports their data.

III. Project Goals & Benefits

Our project goal is to create a system that manages the collar and weather station data, integrates it with map data and ArcGIS, and provides a UI for researchers to analyze portions of the data. This will save researchers time and facilitate analysis with less errors.

IV. Requirements

The following are a list of our requirements, broken up into functional and non-functional, and then broken up into components and the requirements for each of those components.

Functional Requirements:

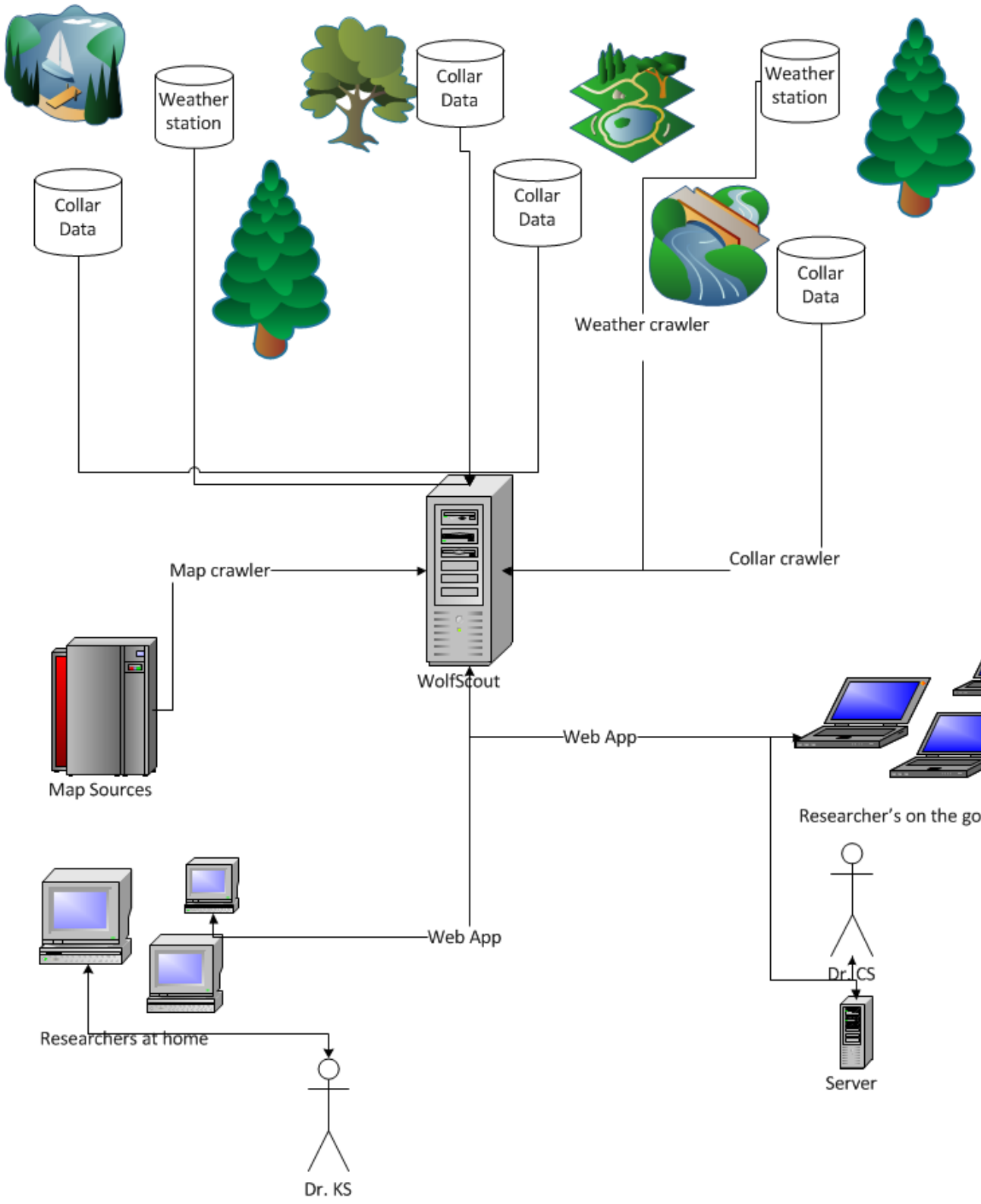
- Graphical UI
  - May be a web app or stand-alone GUI
    - Will depend on if a web app can provide the same functionality
  - Retrieves/store data from/to the database:
    - Collar data (temperature, time, location)
    - Climate data
    - Maps
  - Integrates with ArcGIS 10
  - Authenticates the user
- Database
  - Meets OpenGIS standards
  - Links data according to their time and locations
- Server
  - Reads the following data from a file:
    - Collar data
    - Weather station data
  - Stores data in and serves data from the database
  - Calculates the following:
    - Interaction data
    - Home-range

Nonfunctional Requirements:

- Database
  - Can handle 2 million GPS data points per month
  - PostgreSQL 8.4
  - PostgreSQL-8.4-PostGIS
- Server
  - Can handle 50-60 users at once
  - If fails, can be brought back up within a day.
  - Ubuntu 11.04+
  - Python 2.7.x
  - Supporting Packages:
    - Django==1.3
    - Fabric==1.2.0
    - PIL==1.1.7
    - South==0.7.3
    - coverage==3.5.1b1
    - django-nose==0.2
    - gunicorn==0.13.1
    - logilab-astng==0.22.0
    - logilab-common==0.56.1
    - nose==1.1.2
    - nosexcover==1.0.7
    - paramiko==1.7.7.1
    - psycopg2==2.4.1
    - pycrypto==2.3
    - pylint==0.24.0

        - unittest2==0.5.1
        - wsgiref==0.1.2
- Documentation
    - Non-trivial code is documented
    - Administrative server management is documented

V. Design

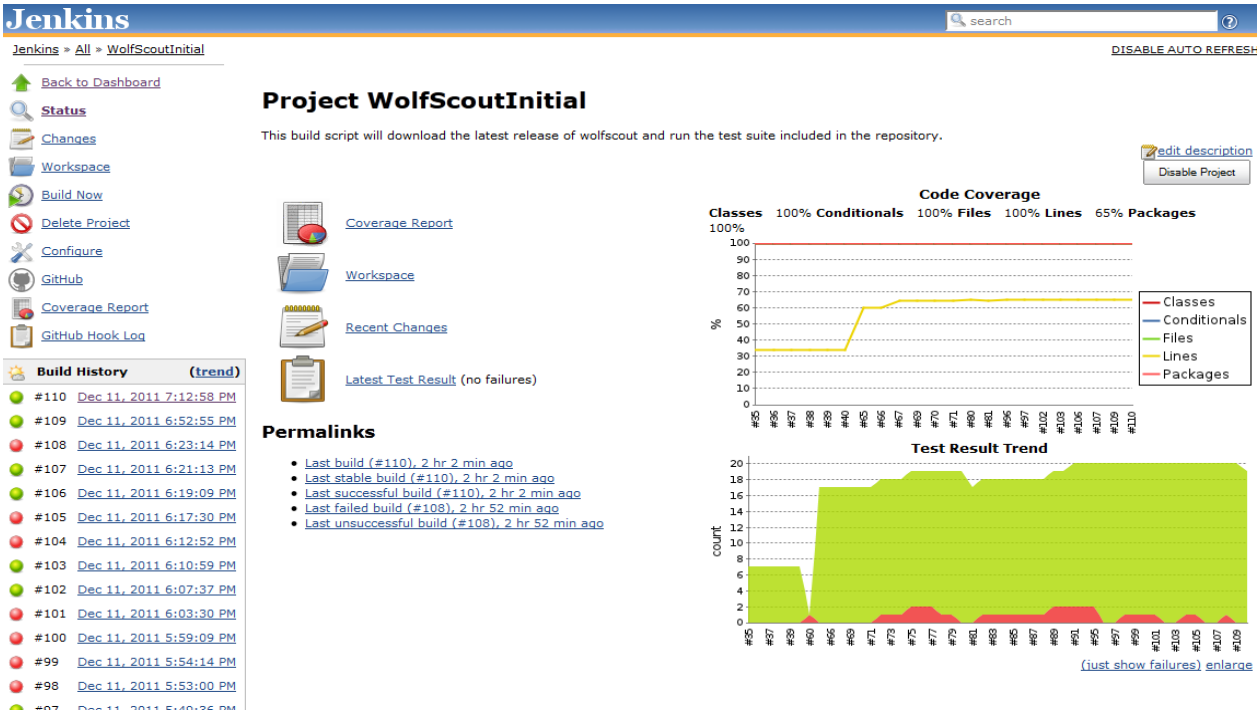

Weather station
Collar Data
Collar Data
Collar Data
Weather station
Collar Data
Weather crawler
Map crawler
Collar crawler
WolfScout
Map Sources
Web App
Researcher's on the go
Web App
Dr. CS
Researchers at home
Server
Dr. KS

Above is a simplified diagram of our overall architecture. The server, in the center and named WolfScout, grabs all the collar data, weather station data, and maps with crawlers. Users can then access that data via a web application.

VI. Test Plan/Cases

Automated tests are written for collar.py. All tests are run and must pass before any code can be committed or merged in the repository. We use a tool called Fabric to help enforce/facilitate this.

Our test plan is to have 70% statement coverage. We help track this through the use of a tool called Jenkins. Jenkins can be accessed via http://<serverIP>:8080, where <serverIP> is our server's IP address (currently 152.14.104.35).



Below is a graph of our project's test coverage.

Project Coverage summary

Name	Classes	Conditionals	Files	Lines	Packages
Cobertura Coverage Report	100% 51/51	100% 0/0	100% 51/51	65% 928/1421	100% 9/9

Coverage Breakdown by Package

Name	Classes	Conditionals	Files	Lines
apps	N/A	N/A	N/A	N/A
apps.crawler	N/A	N/A	N/A	N/A
apps.crawler.cronos	100% 4/4	N/A	100% 4/4	59% 120/204
apps.crawler.cronos.migrations	100% 7/7	N/A	100% 7/7	61% 175/285
apps.crawler.gpscollar	100% 6/6	N/A	100% 6/6	59% 183/310
apps.crawler.gpscollar.migrations	100% 2/2	N/A	100% 2/2	88% 23/26
apps.general	100% 3/3	N/A	100% 3/3	90% 27/30
apps.study	100% 4/4	N/A	100% 4/4	53% 101/192
apps.study.migrations	100% 18/18	N/A	100% 18/18	85% 206/242
apps.wildlife	100% 3/3	N/A	100% 3/3	60% 48/80
apps.wildlife.migrations	100% 4/4	N/A	100% 4/4	87% 45/52

In addition to the automated tests, we have the following test scripts which should be manually run through periodically.

Test ID	Description	Expected Results	Actual Results
login	Preconditions: You are not logged in to the UI. The server is up and running. 1. Open the UI home page.	You are shown the login page.	
login_validUser	Preconditions: You are not logged in to the UI. The server is up and running. User ‘user’ and password ‘password’ is registered in the system. 1. Open UI home page 2. Enter username ‘user’ and password ‘password’. 3. Submit.	You are logged in.	
login_invalidUser	Preconditions: You are not logged in to the UI. The server is up and running. User ‘bad’ is not registered in the system. 1. Open the UI home page. 2. Enter username ‘bad’ and password ‘bad’. 3. Submit.	You are not logged in. You are still shown the login page. You are shown a message that explains that login failed.	
login_CSS	Preconditions: You are not logged in to the UI. The server is up and running. 1. Open the UI home page.	An alert does not appear.	

	2. Enter the following for the username and password: ‘<<SCRIPT>alert("XSS");//<</SCRIPT>’. 3. Submit.		