**Red Team**

**Stargazer**

**Automatic Telescope Control System**

**Project Plan**

**Team Members:**

Rob Grmek

Robert Smith

**Instructor:**

**Youry Khmelevsky**

**Course:**

**COSC 470**

**Date:**

**January 7th, 2010**

# ****Revision History****

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 10/16/2009 | 1.0 | Document created. | Robert, Rob, Jason |
| 11/1/2009 | 1.1 | Added information for 2nd iteration. | Rob |
| 11/8/2009 | 1.2 | Added additional information for release #1 and #2. | Rob |
| 11/13/2009 | 1.3 | Minor updates. | Rob |
| 11/16/2009 | 1.4 | Added metrics for release #1 | Rob |
| 11/24/2009 | 1.5 | Added metrics for release #2 | Rob, Robert |
| 11/29/2009 | 1.6 | Added some changes. | Rob |
| 1/7/2010 | 1.7 | Plans for the second semester. | Rob |
| 1/20/2010 | 1.8 | Updated plans. | Rob |

# ****Table of Contents****

**Page**

|  |  |
| --- | --- |
| Introduction | 1 |
| Risk Management Plan | 2 |
| Iteration 1: Inception, Partial Elaboration and Initial Design | 3 |
| Iteration 2: SPIKE Project | 5 |
| Iteration 3: Prototype (Release #1) | 6 |
| Iteration 4: Prototype (Release #2) | 7 |
| Iteration 5: Prototype (Release #3) | 8 |
| Iteration 6: Prototype (Release #4) | 10 |
| Iteration 7: Prototype (Release #5) | 11 |
| Iteration 8: The Final Release | 12 |

# ****Project Plan****

This document covers the planning process for the design, development and deliverance of our Stargazer projectfor our customers Alexander and Nan.

Note:

Durations in our planning process is measured in units. 1 unit is equal to 15 minutes.

Task durations tracked with Time Tracker: <http://timetracker.wrconsulting.com>

Furthermore, the analysis of possible risks with corresponding risk management solutions which attempt to either minimize or avoid the risk from becoming an issue is included as well in this document.

Also, further project management can be done using the following tool: <http://project.youry.net>

Additional metrics for the web application can be dynamically attained through the following website: <http://getcaliper.com/> using the application’s Git repository (git://github.com/RedTeamCOSC470/Stargazer.git).

# Risk Management Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Probability | Effect | Solution |
| Underestimate the time the project will take. | Medium to high. | Serious – we run out of time. | Remove non-essential features. |
| Illness to team members. | Low to medium. | Catastrophic – We are low on manpower as it is. | Assign and distribute their tasks to the other team member. |
| Not understanding requirements. | Medium to high. | Serious -The system behaves incorrectly. | Keep in close contact with the customer. |
| Scheduling conflicts between team members. | Low to medium. | Serious – Individual programming is not XP programming. | Schedule well before hand. |
| No available lab space. | Medium. | Tolerable – need to find other areas to work. | Use laptops in meeting rooms or other areas. |
| Server or technical issues. | Low to medium. | Catastrophic – May lose work. | Backup regularly; use a versioning control system. |
| Campus network problems. | High. | Serious – Reduced productivity, difficulty accessing development server. | Work directly on development server or work locally on a laptop computer. |
| Lack of time at end of semester. | High. | Tolerable – Reduced ability to work outside of class/lab time. | Use class/lab time as productively as possible. |
| Client unavailable or unresponsive. | Medium. | Catastrophic – No customer input. | Use alternative methods of communication (phone or email).  May have to select a different project. |

Probabilities can be: low, moderate, high.

Effects can be: tolerable, serious and catastrophic.

# Iteration 1: Inception, Partial Elaboration and Initial Design

We have selected our project and have begun drafting initial documentation related to the project and set up a development environment.

Also, we planned to run through several Ruby on Rails related tutorials to learn how to use the development, how to use versioning control with Git and how to use unit testing with Ruby’s built-in unit testing.

### Task Duration and Dependencies:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **#** | **Estimated Duration** | **Dependency** | **Actual Duration** |
| Create initial vision | 1 | 2 | - | 2 |
| Create project plan document | 2 | 2 | - | 3 |
| Add business case and product info to vision | 3 | 4 | 1 | 3 |
| Add architecture to vision | 4 | 1 | 1 | 1 |
| Add risks to project plan | 5 | 2 | 2 | 1 |
| Research image recognition technologies | 6 | 24 | - | - |
| Research hardware I/O specifications | 7 | 24 | - | - |
| Research development environment components | 8 | 24 | - | 12 |
| Set up development environment | 9 | 16 | 8 | 8 (inc.) |
| Create tutorial plan | 10 | 2 | - | 2 |
| Create initial developer's guide | 11 | 8 | - | 6 |
| Create use case diagram | 12 | 2 | - | 2 |
| Write fully dressed use cases | 13 | 6 | 12 | 4 |
| Complete tutorials using our framework | 14 | 32 | 8, 9 | - |
| Draft acceptance test | 15 | 4 | - | - |
| Add installation guide information | 16 | 4 | 11 | 4 |

### Tutorial Plan:

Setting up a development environment on Windows:

<http://www.akitaonrails.com/2009/1/13/the-best-environment-for-rails-on-windows>

Setting up a development environment on Fedora 11:

<http://www.technetra.com/2009/04/22/howto-setting-up-ruby-on-rails-for-fedora-10-and-11/>

Rails Guides:

1. <http://guides.rubyonrails.org>
2. <http://www.tutorialspoint.com/ruby-on-rails-2.1/index.htm>
3. <http://www.meshplex.org/wiki/Ruby/Ruby_on_Rails_programming_tutorials>

Rails Development Tutorials:

1. <http://guides.rubyonrails.org/getting_started.html>
2. <http://www.rails4days.pwp.blueyonder.co.uk/Rails4Days.pdf>
3. <http://www.tutorialized.com/view/tutorial/Image-uploads-and-resizing-for-Rails-models-with-mini-magick/19070>

Git Tutorial:

<http://harryseldon.thinkosphere.com/2009/01/14/git-and-rails-a-detailed-tutorial-including-plugins-submodules-development-and-production>

# Iteration 2: SPIKE Project

We have researched the technologies related to the project in iteration 1 and can now begin a short one week SPIKE project to learn the Ruby on Rails framework. The SPIKE project we created was that of a blogging site.

The SPIKE project must:

* Have login system and authentication; since this is important to learn how to do for our real project.
* Be committed regularly to our GitHub repository and use source control management (SCM).
* Have unit testing; we’ll need to use test-driven development (TDD) in our real project, so it is key to learn the built-in ruby testing tool.

### List of Tasks:

|  |  |  |
| --- | --- | --- |
| **Task** | **Estimated Duration** | **Actual Duration** |
| SPIKE: Unit testing tutorials | 4 | 3 |
| SPIKE: Use Git to push/pull project to/from GitHub | 2 | 2 |
| SPIKE: Post scaffolding tutorial | 6 | 5 |
| SPIKE: Comment and tag functionality tutorial | 6 | 6 |
| SPIKE: Collect SPIKE code for submission | 2 | 3 |
| Design: Create domain model | 4 | 4 |
| Design: Create architecture diagram | 2 | 2 |
| Documentation: Add using Git information to developer's guide | 4 | 3 |
| Documentation: Update project plan for release #1 and #2 | 4 | 6 |
| Documentation: Add architecture diagram to developer's guide | 2 | 2 |
| Documentation: Get extra user stories from customer | 3 | 4 |
| Documentation: Prioritize the user stories | 1 | 1 |
| Documentation: Update installation information | 4 | 4 |
| Documentation: Update vision | 2 | 2 |
| Set up development environment | 8 | 6 |
| Test telescope interface/drivers | 12 | 26 |
| Test telescope programs (Autostar, ASCOM) | 8 | 10 |
| Total: |  | 89 |

### Milestones:

* SPIKE completion: Nov. 6, 2009

# Iteration 2: Prototype (Release #1)

From our knowledge gained from the SPIKE project, we can begin development of the web application portion of our project.

In this one week iteration, we plan to implement the following functionality:

* Authentication; force the user to login in order to access the system.
* Scheduling telescope position; show a form to allow the user to input a time, date, coordinates, etc which contains all the information necessary to properly position the telescope.
* Review scheduling logs; show entries for all schedules that have been created in the past, present and future.
* Online help documentation; gives users information on using the system.

### List of Tasks:

|  |  |  |
| --- | --- | --- |
| **Task** | **Estimated Duration** | **Actual Duration** |
| Documentation: Update testing plan | 3 | 2 |
| Create web system: Add authentication | 12 | 11 |
| Create web system: Add scheduling form (with validation/tests) | 16 | 18 |
| Create web system: Create scheduling review logs | 2 | 1 |
| Create web system: Add online help documentation | 12 | 4 |
| Create web system: Add authorization | 16 | 23 |
| Documentation: Create change requests document | 1 | 1 |
| Documentation: Update all other documents. | 24 | 18 |
| Design: Create class diagram | 2 | 8 |
| Design: Update use case diagram | 1 | 1 |
| Refactoring | 4 | 4 |
| Total: | 93 | 89 |

### Milestones:

* Release #1: Nov. 18, 2009

#### Metrics:

Unit Testing:

* Unit tests: 2
* Test cases: 18
* Assertions: 31

Implementation:

* Views: 15 (including partials)
* Models: 3
* Controllers: 4

# Iteration 4: Prototype (Release #2)

In this iteration, we are to primarily build onto the web systems functionality. We plan to create:

* Stylesheets; create separate CSS for both mobile device users and PC users for easier visibility and usability.
* An image gallery; show latest composite images from the telescope.
* HTTPS/HTTP for admin/user
* Set up production server (use Apache as production web server)
* Changes to schedule input forms and model

Also, extensive refactoring must be done in this iteration.

### List of Tasks:

|  |  |  |
| --- | --- | --- |
| **Task** | **Estimated Duration** | **Actual Duration** |
| Add separate stylesheets for PC users and mobile device users | 8 | 8 (inc.) |
| Add gallery section - display latest images from telescope | 16 | 22 |
| HTTPS/HTTP for admin/user | 8 | 6 |
| Set up production server | 8 | 10 |
| Update all documentation and create design diagrams | 40 | 16 (inc.) |
| Refactoring | 20 | 6 |
| Change schedule input forms | 12 | 16 |
| Fix unit tests | 4 | 4 |
| Fix CSS issues | 4 | 5 |
| Create PowerPoint Presentation | 6 | 10 |
| Total: | 100 | 103 |

### Milestones:

* Release #2: Nov. 25, 2009
* Refactored Release #2: Dec. 4, 2009
* Final Presentation: Dec. 4, 2009

#### Metrics:

Unit Testing:

* Unit tests: 3
* Test cases: 17
* Assertions: 29

Implementation:

* Views: 26 (including partials)
* Models: 4
* Controllers: 6

# Iteration 5: Prototype (Release #3)

Web system functions to be implemented and tasks to be completed:

* Job scheduling; telescope positioning schedules created through the web system should correspond to a job schedule on the OS to run at the specific time as entered by the user.
* Integrate libraries which contain celestial objects of interest (e.g. certain planets) so as to allow users an alternative to entering coordinates manually.
* Some refactoring left from last iteration needs to be completed.
* Current online help documentation needs to be added.

Tasks centered on the telescope controlling application:

* Allow ability to park the telescope.
* Allow telescope movement to specific celestial coordinates.

Modeling that needs to be initially created for the telescope-controlling application:

* Sequence diagrams.
* Class diagram.

### List of Tasks:

|  |  |  |
| --- | --- | --- |
| **Task** | **Estimated Duration** | **Actual Duration** |
| Documentation updates | 16 |  |
| Add new online help documentation to website | 12 |  |
| Create sequence diagrams | 8 |  |
| Create class diagram | 4 |  |
| Include telescope parking ability | 1 |  |
| Include telescope movement ability from entered celestial coordinates | 4 |  |
| Integrate libraries (which have celestial objects of interest) with web site | 20 |  |
| Minor web site code refactoring | 6 |  |
| Draft user acceptance test | 1 |  |
| Research Rails plugin to handle scheduling | 2 |  |
| Create triggers/batch file to run application and additional actions | 1 |  |
| Total: | 75 |  |

### Milestones:

* Release #3: Jan. ?, 2010

#### Metrics:

Unit Testing:

* Unit tests:
* Test cases:
* Assertions:

Implementation:

* Views: (including partials)
* Models:
* Controllers:

# Iteration 6: Prototype (Release #4)

Web system functions to be implemented and tasks to be completed:

* Add mail server with accompanying mail system.
* Pagination added to tables such as the schedule log.
* Increase usability for mobile users by removing dropdown select boxes which are used for filters; always show current day with arrows to switch between days.

Tasks centered on the telescope controlling application:

* Coordinate the camera with the telescope; take pictures once the telescope has positioned itself to the celestial coordinates of interest and then begin taking images until time has elapsed or the number of pictures input from the user is reached.

### List of Tasks:

|  |  |  |
| --- | --- | --- |
| **Task** | **Estimated Duration** | **Actual Duration** |
| Documentation updates | 16 |  |
| Implement mail server/mailing system | 14 |  |
| Add pagination to website tables | 10 |  |
| Changes to schedule log filtering for mobile device views | 4 |  |
| Fix functional tests | 12 |  |
| Integrate camera to telescope-controlling application | 18 |  |
| Total: | 74 |  |

### Milestones:

* Release #4: Feb. ?, 2010

#### Metrics:

Unit Testing:

* Unit tests:
* Test cases:
* Assertions:

Implementation:

* Views:
* Models:
* Controllers:

# Iteration 7: Prototype (Release #5)

Web system functions to be implemented and tasks to be completed:

* Ability for live streaming of the telescope to see current images of that specific position.
* Ability for manual positioning of the telescope; users may remotely control the telescope from the web site by slewing the telescope north, south, east, or west.

Tasks centered on the telescope controlling application:

* Image processing: once the many images have been taken, a composite image is assembled.
* Image processing: image recognition needs to be realized. New objects are found by comparing composite images to previous images or that of those in libraries.
* Take images and send it to the web system in real time.

### List of Tasks:

|  |  |  |
| --- | --- | --- |
| **Task** | **Estimated Duration** | **Actual Duration** |
| Documentation updates | 16 |  |
| Implement manual telescope positioning via web system | 14 |  |
| Research image processing options and components | 12 |  |
| Implement ability for streaming images live to the web system | 24 |  |
| Implement functionality for compiling composite images | 18 |  |
| Implement functionality for image recognition | 24 |  |
| Total: | 108 |  |

### Milestones:

* Release #5: Feb. ?, 2010

#### Metrics:

Unit Testing:

* Unit tests:
* Test cases:
* Assertions:

Implementation:

* Views:
* Models:
* Controllers:

# Iteration 8: The Final Release

Extensive refactoring must be done in this iteration and the final product needs to be deployed. Furthermore, all project-related metrics and statistics need to be gathered and documentation needs to be finalized.

### List of Tasks:

|  |  |  |
| --- | --- | --- |
| **Task** | **Estimated Duration** | **Actual Duration** |
| Finalize documentation | 24 |  |
| Compile final metrics and project statistics | 4 |  |
| Migrate database to PostgreSQL | 3 |  |
| Deploy systems to production environment | 4 |  |
| Final refactoring | 20 |  |
| Final user acceptance test | 4 |  |
| Create PowerPoint presentation | 10 |  |
| Total: | 69 |  |

### Milestones:

* Final Release: Apr. ?, 2009
* Final Presentation: Apr. ?, 2009

#### Metrics:

Unit Testing:

* Unit tests:
* Test cases:
* Assertions:

Implementation:

* Views:
* Models:
* Controllers: