# Software Requirements Specification
for the
# Rotation Curve Modeler (RoCM)

Version 2.0

Robert Moss, Alex Clement
Wentworth Institute of Technology

June 5, 2014

# Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

| **Signature** | **Printed Name** | **Title** | **Date** |
|---|---|---|---|
| | | | |
| | | | |

# Contents

# 1 Introduction

## 1.1 Purpose

The Software Requirement Specification for the Rotation Curve Modeler will explain in detail necessary features that the client purposes and the developers provide. Astrophysicists will be the main target for this web application. It will expedite the process of accessing galactic data (from the project Scholarly Observed Celestial Measurements) and provide a tool to model galaxies using many different theories to solve the rotation curve problem.

## 1.2 Scope

1. Rotation Curve Modeler
   (a) Interact with SOCM (Scholarly Observed Celestial Measurements)
      i. Use the repository of observable galactic data to model hundreds/thousands of different galaxies.
   (b) Interact with RoCS (Rotation Curve Simulation)
      i. RoCS visualizes the spin of star clusters around the center of a galaxy.
      ii. Include scale and legend for the visualization.
   (c) Allow users to import their own model (via JavaScript code)
      i. Following the defined v(R) input/output standard.
      ii. Observable galactic parameters from SOCM will be available as constants.
      iii. User defined constants will need to be implemented in the user's function.
   (d) Import LaTeX equation for each model (optional)
      i. The user can import their own LaTeX equation to be displayed during the data plotting.
      ii. Aids in understanding the behavior of each parameter.
   (e) Dynamic parameter sliders
      i. For every parameter in the individual models, a dynamic slider with user defined ranges can be created.
2. www.RoCMSOCM.com
   (a) The website will host the RoCM, SOCM, and ROCS components.
      i. This website, hosted on WIT servers, will allow astrophysicists to access a large database of observed galactic data (via SOCM), plot their own rotation curve models (via RoCM), and simulate their rotation curves (via RoCS).

## 1.3 Definitions, Acronyms, and Abbreviations

- Application Specific Definitions
  - RoCM - Rotation Curve Modeler
  - SOCM - Scholarly Observed Celestial Measurements
  - RoCS - Rotation Curve Simulator

- Industry Definitions
  - WIT - Wentworth Institute of Technology
  - SRS - Software Requirements Specification
  - JavaScript - A web based programming language.
  - D3 - Data Driven Documents: A JavaScript library for data visualization.
  - Ruby on Rails - A web development framework written in the Ruby programming language
  - JQuery - A JavaScript library for easy UI development.
  - LaTeX - A document preparation system used widely throughout science and mathematics.
  - SVG - Scalable Vector Graphics: A loss-less graphics format.
- Technical Definitions
  - MoND - Modification of Newtonian Dynamics
  - TeVeS - Tensor-vector-scalar gravity
  - MATLAB - A mathematical programming language (MATrix LABoratory)
  - Mathematica - A mathematical programming language.
  - DB - Database
  - GUI - Graphical User Interface
  - HTML - HyperText Markup Language
  - div - HTML tag to define a division in a document

## 1.4 References

The list of references below are software documentation that we will be using:

1. Data Driven Documents (D3): http://d3js.org/
2. JQuery documentation: http://api.jquery.com/
3. JQuery UI documentation: http://api.jqueryui.com/

## 1.5 Overview

The rest of the SRS will contain:

1. Specific features RoCM and the RoCMSOCM website
2. System Requirements
3. Design Constraints for the application.
4. Risks within the scope.
5. Any additional information about the development of the application.

# 2 General Description

This section will explicitly lay out the product's functionality and constraints as well as compare it to existing products. Possible set backs will be discussed. Details about each requirement will be laid out in Section 3: Specific Requirements.

## 2.1 Product Perspective

Astrophysicists will need to model galaxies in programs like MATLAB or Mathematica, but there doesn't exist a singular tool to expedite this process in a universal format. Currently, scholars must sift through peer-reviewed articles and gather data one-by-one. RoCM will help generalize the work being done on galaxy research. With observable data as the input, any galaxy can be imported into the tool.

## 2.2 Product Functions

There will be many available functions of RoCM users can utilize. The modular design of the software will allow for additional functions to be easily implemented.

1. A rotation curve plotting tool that overlays different models of the galaxy on top of the observational data. Gives the user the ability to select which curve to plot based on the models available. **TODO**
2. Value ranged sliders that can update each individual parameter within specific galactic models. Each slider can be dynamically created with user defined ranges. This enables the user to visualize the behavior of each parameter within the each model. Allows for the testing of uncertainty within the galactic parameters.S

## 2.3 User Characteristics

Any computer user who has trouble with screen space will find this software useful. The general user will be a laptop owner who has a built in camera. Anyone from developers to web-surfers with a screen space or battery issue will want to use this software.

## 2.4 General Constraints

The main functionality of the TM will rely heavily on the availability of an input camera; preferably stationed at the center-top of the screen. If said camera is unavailable, the user can select the mouse as a functional alternative. When using the TM, visibility of the person in control will be crucial.

## 2.5 Assumptions and Dependencies

A camera is assumed to be installed, due to the main functionality incorporating a computer vision tracking module. However, there are alternatives that have been discussed in the General Constraints section.

The windows control module (WCM) is dependent on C++ due to the Windows API written in that language. It is also assumed that the user is running a Windows operating system.

# 3 Specific Requirements

This section will explicitly lay out the purposed system design and it's individual requirements.

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

Users must interface with a computer or other device capable of accessing and displaying internet contents. The system will require a cursor device (e.g. mouse or touchpad on a PC or laptop, touch-enabled screen on mobile device). The website and its contents were designed for a modern PC.

### 3.1.2 Hardware Interfaces

Users will need a device capable of supporting a modern browser and an Internet connection. Experience may depend on hardware capabilities and Internet speeds.

### 3.1.3 Software Interfaces

The system will require an Internet browser capable of running Javascript.

### 3.1.4 Communications Interfaces

The system demands an Internet connection to access www.RoCMSOCM.com and the SOCM database.

## 3.2 Functional Requirements

### 3.2.1 Tracking Module (TM)

#### 3.2.1.1 Introduction

The TM simply tracks the position of the user. Encapsulating it enables the developers to easily swap alternatives for how the user is tracked. A C++ implementation of OpenCV will be used.

#### 3.2.1.2 Inputs

OpenCV will send raw camera data as input to the TM in the form of an $nxm$ matrix that specifies each cell as a pixel with RGB values.

#### 3.2.1.3 Processing

There are several ways in which to process the camera feed. Two main implementations of the TM will be as followed: an object tracker (OT), and a facial recognition tracker (FRT). The OT can apply filters to the matrix of colored pixels coming in as input in a variety of ways. It can filter a specific RGB color in the (0:255, 0:255, 0:255) range and apply a noise reduction algorithm to help pinpoint the target object. The option of HSV filters with the same ranges can yield better results especially in a physical environment that is noisy. The noise tends to increases the uncertainty in the tracked object. The settings for which filter works best in the users physical environment will be sent to the SM to be added to the SQLite DB. The user will have to wear a specific item on their head that can be easily tracked. An alternative to this will be implemented with the FRT.

The FRT, taking advantage of the FaceRecognizer 0.05 API that OpenCV provides, will enable the user to not require any external dependencies for the TM to track. An easy to use API, FaceRecognizer

can be a replacement for the OT if the user chooses. As stated above, all data settings will be saved to the SQLite DB.

If the user does not have a camera, or wished not to have their camera on, they can use the mouse as a suitable replacement for the tracked object.

#### 3.2.1.4 Outputs

A simple $(x, y)$ coordinate system of the tracked object, whether it be through the OT, FRT, or by mouse, will be given as output of the TM.

#### 3.2.1.5 Error Handling

Uncertainty in the OT will be minimized using filtering algorithms that reduce the tracked object's noise. If the user selects the FRT, then the uncertainty will be lower, due to light restrictions being less harsh, and the specification of the color of your tracked object not necessary. If a clear tracked object isn't present and the mouse tracking isn't enabled, then the system will assume no object and the TM will not output any coordinates.

### 3.2.2 Windows Control Module (WCM)

#### 3.2.2.1 Introduction

The WCM encompasses each real-estate utilizing feature; whether it's space or battery life you're trying to maximize, the WCM will handle everything. All sub-modules will be written in C++ using the Windows API build date: 3/25/2010.

#### 3.2.2.2 Inputs

The outputs of the TM will act as direct inputs to the WCM, namely the $(x, y)$ coordinates of the tracked object. The WCM doesn't care about how the coordinates were calculated, it only worries about manipulating the operating system.

#### 3.2.2.3 Processing

There are three sub-modules to the WCM (with additional features easily added by the developers):

- Windows Grid Organizer

  The user will be able to select five open windows to be placed in a grid structure around their screen. The top, bottom, left, right, and middle portions will be available for configuration. The user can define the ratio of each window size proportional to the total screen size. This is true for each section except the middle cell; which fills the entire screen once all five windows are chosen. The user can activate the predefined hot-key, defaulted to Alt, and peer in the four outer directions. The middle window will move in the opposite direction to simulate the "peering" action of the user. The speed at which the middle cell moves can be altered and saved using the SM.

- Windows Perspective (3D)

  This feature will organize all open windows into a layered view (with the illusion of 3D windows) for the user to "look" around their desktop and see which window they want to select. The windows in the back layer will move at a much slower rate than the windows in the front layer; to give the illusion of a 3D perspective.

- Sleep Manager

If the Sleep Manager is on and the user is away from the computer, the TM will detect that nobody is present and relay that message to the Sleep Manager. The Sleep Manager will then assess if the user is in fact away from the keyboard (via the absence of key presses or mouse movements). If all requirements are aligned, the computer will either lock or sleep after a user defined delay. All of the settings for this configuration will be handled by the SM.

#### 3.2.2.4 Outputs

The output of the WCM will be a structure that contains the settings applied to that feature. This structure will be defined in later documents pertaining to design.

#### 3.2.2.5 Error Handling

Each sub-module will have their own set of error handlers. The key component to all of them will be the integrity of the Desktop.

- If the Windows Grid Organizer does not meet all the necessary cells, it will default to empty cells and still be able to function.
- The Windows Perspective will have minimal errors associated with it, due in part to the simplistic nature of the feature; only window resizing and repositioning is being applied.
- The Sleep Manager will be handled with care and will default to staying awake with any input from the TM, mouse or keyboard (as to minimize intrusive actions).

### 3.2.3 Settings Module (SM)

#### 3.2.3.1 Introduction

The SM will be written in C++ as a transparent module to deal with organizing and saving users data. A local SQLite DB will hold the user's settings for quick access and increased fidelity in TACS.

#### 3.2.3.2 Inputs

The output of the WCM will be a structure of settings for each WCM feature. This structure will be taken in as input to the SM.

#### 3.2.3.3 Processing

The SM, with the predefined structure, will separate the settings by their respective feature. Once separated, the data will be saved to an already created SQLite DB that can be queried at any time.

#### 3.2.3.4 Outputs

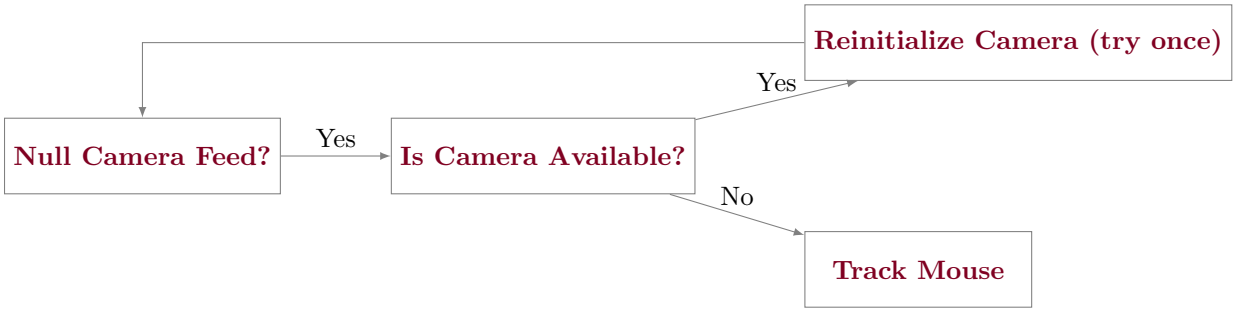
A binary variable output will declare if the transaction between the SM and the SQLite DB was successful.

#### 3.2.3.5 Error Handling

If settings were not specified, a list of initial defaults will be available to fall back on.

## 3.3 Use Cases

### 3.3.1 Uncooperative/Nonexistent Camera




## 3.4 Classes/Objects

1. RoCM.html
   (a) Plot div
   (b) Parameter Table div
   (c) Parameter Slider div
   (d) LaTeX Equation div
   (e) JavaScript Model Input div
2. RoCM.js
   (a) PlotView.js
   (b) ParamTable.js
   (c) ParamSlider.js
   (d) LaTeXEquationViewer.js
   (e) JSModelInput.js
   (f) GalacticModels.js
3. RoCMWebApplication Ruby on Rails Application
   (a) RoCMModel.rb
   (b) RoCMView.rb
   (c) RoCMController.rb

## 3.5 Non-Functional Requirements

### 3.5.1 Performance

The TM should be able to track an object with a 90% certainty. This can be monitored via an algorithm that takes the users given position, and compares it to the previous one to look for large deviations. If the variance is above a certain threshold, the percentage of certainty will be marked down. A camera feedback of 1 second will be necessary for the TM to talk to the WCM effectively.

### 3.5.2 Reliability

The users customized configuration of their TACS environment will be saved using the SM. This will enable the user to relaunch the program with the same settings as previous sessions.

### 3.5.3 Availability

A internet connection is not necessary for TACS to run, allowing the system to be act independently offline. TACS replies heavily on the installation of a camera, but is not completely dependent on one.

### 3.5.4 Security

The settings that are saved will not require sensitive personal information, so security is a low priority. The use of the camera is the only vulnerability in TACS; however, the lack of internet connectivity means a secure network connection is not necessary.

### 3.5.5 Maintainability

The modular nature of the design will help developers maintain each encapsulated portion of TACS. Each module can be updated separately, without intruding on it's neighboring modules. Only if there is a redefinition of inputs and outputs will each module need to be updated accordingly.

### 3.5.6 Portability

This software will be built on several machines ranging from Windows 7 (6.1.7601) to Windows 8.1 (6.3.9600). The intended system for use will be a Windows operating system within that range. Because TACS relies heavily on the Windows API, the Linux and Macintosh machine will not be supported.

## 3.6 Inverse Requirements

TACS stresses that the user shouldn't need to learn any proprietary material in order to use the system. Therefore, TACS does not have any inverse requirements.

## 3.7 Design Constraints

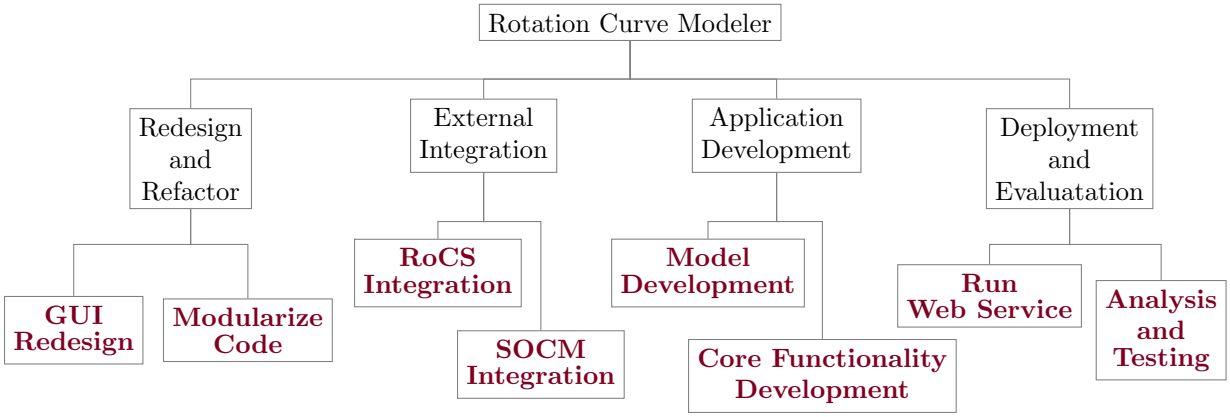
Given most laptops come standard will a built-in camera, the design behind TACS is safe. The modular design organizes the individual system features so they can be separately worked on.

## 3.8 Logical Database Requirements

A low maintenance database will be used as a simple data storage facility. This SQLite DB won't need any specific size requirements, due to the minimal amount of data that's required to save.

# 4 Project Planning and Risk Management

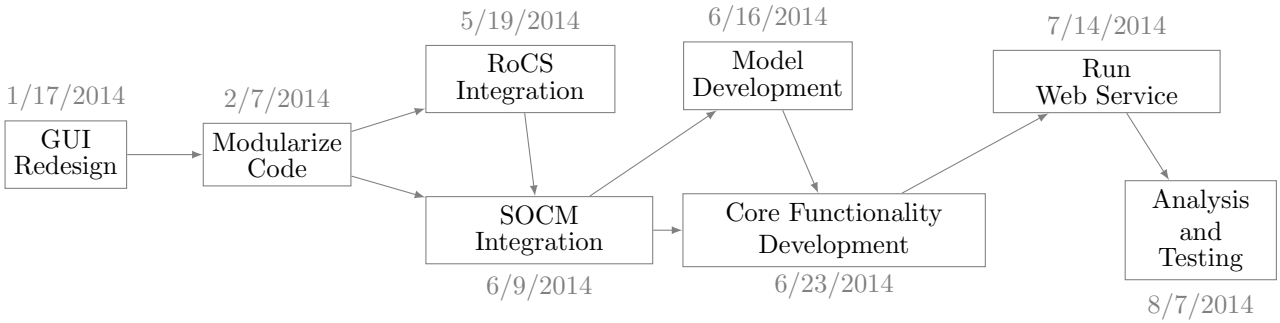
## 4.1 Work Breakdown Structure



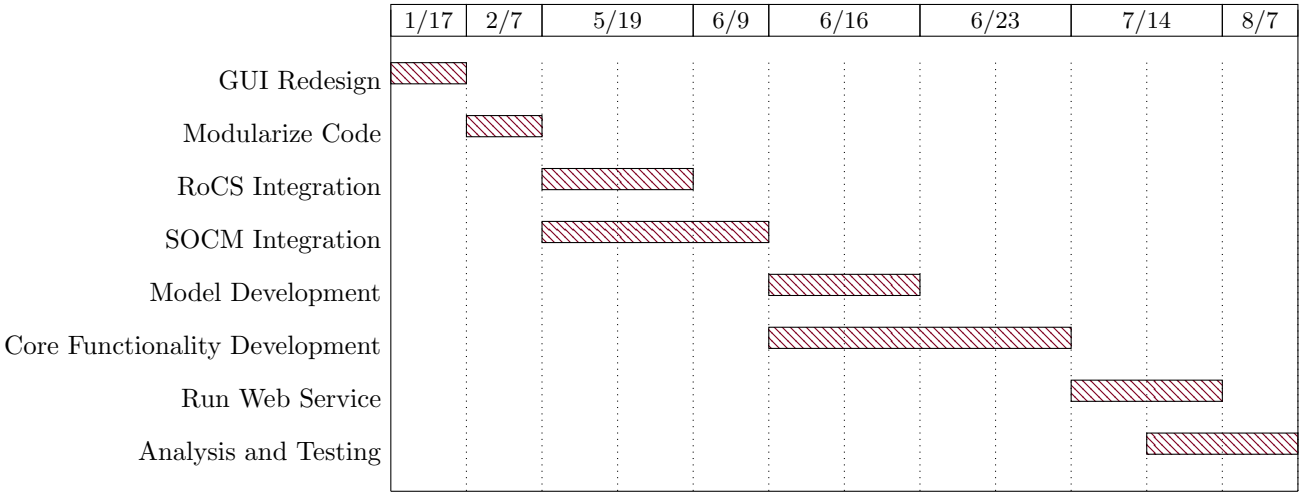

## 4.2 Time Estimation

| Task | Number of Weeks |
|---|---|
| GUI Redesign | 1.5 |
| Modularize Code | 3 |
| RoCS Integration | 1 |
| SOCM Integration | 2 |
| Model Development | 2 |
| Core Functionality Development | 5 |
| Run Web Service | 1 |
| Analysis and Testing | 1 |
| **Total** | **16.5** |

## 4.3 Activity Sequencing Diagram




## 4.4 Gantt Chart




## 4.5 Risk Management

| Risk | Likelihood | Consequences | Total | Fix |
|---|---|---|---|---|
| No camera | 1 | 3 | **3** | The user can use the mouse as a replacement for the head tracking. |

# A Appendix

## A.1 Design Concept

A tentative conceptual design of the system is as followed: