Software Requirements Specification
for the
Track and Control System

Version 1.0

Robert Moss, Aaron Periera, Matthew Shrago

February 6, 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 Track and Control System will explain in detail necessary features that the client purposes and the developers provide. The user base will be focused on anyone who will want to use their computer in a more efficient way and take advantage of their full screen space.

1.2 Scope

1. Track and Control System
   (a) The software will track a user's movements and be able to control numerous features around their Desktop with the movement of their head.
      i. The everyday computer user will utilize this software to organize their cluttered Desktop.
   (b) The application will also act as a security monitor to recognize when you're present at your computer and lock your screen accordingly.
      i. A benefit of this will be a sense of security for the user when they're away from their computer.

1.3 Definitions, Acronyms, and Abbreviations

- Application Specific Definitions
  - TACS - Track and Control System
  - TM - Tracking Module
    - OT - Object Tracker
    - FRT - Facial Recognition Tracker
  - WCM - Windows Control Module
  - SM - Settings Module
- Industry Definitions
  - SRS - Software Requirements Specification
  - OpenCV - Open Computer Vision: An open source library for object tracking via the camera.
  - SQLite - A lightweight, low maintenance, self contained database.
  - DB - Database
  - RGB - Red, Green, Blue color values.
  - HSV - Hue, Saturation, Value.
  - API - Application Programming Interface
  - C++ - An object oriented programming language.
  - GUI - Graphical User Interface
  - QT - An API for building GUIs
- Technical Definitions
  - i:j - The range of integers starting at i, increasing to j by 1.

1.4 References

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

1. OpenCV documentation: http://opencv.org/
   - FaceRecognizer API: http://docs.opencv.org/trunk/modules/contrib/doc/facerec/facerec_api.html
2. Windows API Index: http://msdn.microsoft.com/en-us/library/hh920508(v=vs.85).aspx
3. QT C++ documentation: http://qt-project.org/

1.5 Overview

The rest of the SRS will contain:

1. Specific features of TACS and their details.
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 in the Specific Requirements section of the SRS.

2.1 Product Perspective

A vast array of other products that utilize OpenCV are available, but no product will control your Desktop with the movement of your head. There has been much research in the field of physical interactions between the user and their computer. However, the application of organizing and securing your Desktop with this interaction is the novel portion of this software.

2.2 Product Functions

There will be many available functions the users can utilize. The modular design of the software will allow for additional functions to be easily implemented. *It should be noted that the window functions will be activated with a user-defined hot key.*

1. A proportional snap to grid set up for users to place desired windows in five different locations. Namely, left, right, top, bottom, and middle. Once all the windows are selected, the middle window will fill the screen and the user will be able to "peer" in the direction of the four other windows to show a preview of them.
2. The software will organize all open windows into a 3D view (with the illusion of layered windows) for the user to "look" around their desktop and see which window they want to select.

3. The software will be able to detect when the user is away from the screen, and with a user defined delay-time, be able to lock or put your computer to sleep (all settings can be changed from the SM).

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 **??** 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

A GUI will be built around the two main modules; the TM, and WCM (which both incorporate the SM). It will be written using the open source C++ framework QT 5.2. The GUI will allow the user to change their settings and act as a liaison between each sub-module. Two tabs views for the TM and WCM will give the user a framework for customizing their TACS environment.

3.1.2 Hardware Interfaces

An installed camera will send raw data to the TM to be processed. No other external hardware dependencies are necessary.

3.1.3 Software Interfaces

The TM will rely on the open source computer vision library, OpenCV 2.4.8, which will be necessary to track the user's movements. For facial recognition, the FRT uses the FaceRecognizer 0.05 API. In addition, the WCM will utilize the easy system calls with the Windows API build date: 3/25/2010.

3.1.4 Communications Interfaces

The SM will have direct communication to the local SQLite 3.8.3 database to save user settings. The DB will contain past window configurations of the user, specified delays and hot keys, as well as TM configurations.

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 3D view (with the illusion of layered 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 that 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

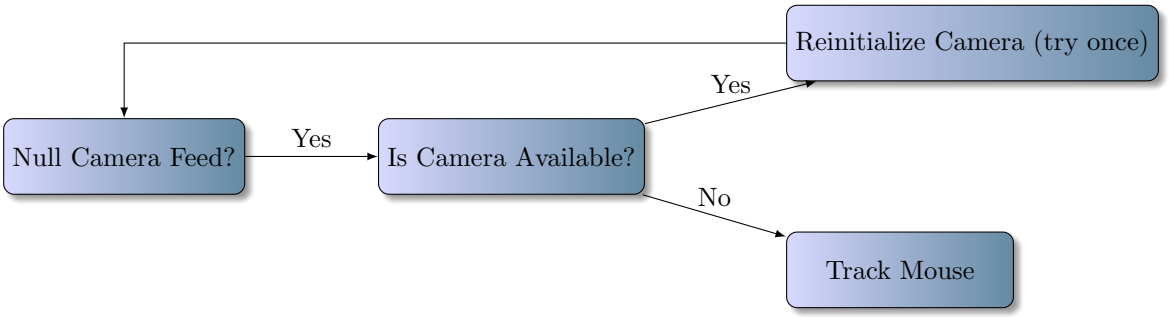
A binary variable output will declare if the transaction between the SM and the SQLite DB was successful.

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. <TrackingModule>
   (a) <ObjectTracker>
   (b) <FacialRecognitionTracker>
   (c) <MouseTracker>

2. <WindowsControlModule>
    (a) <WindowsGridOrganizer>
    (b) <WindowsPerspective>
    (c) <SleepManager>
3. <SettingsModule>

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

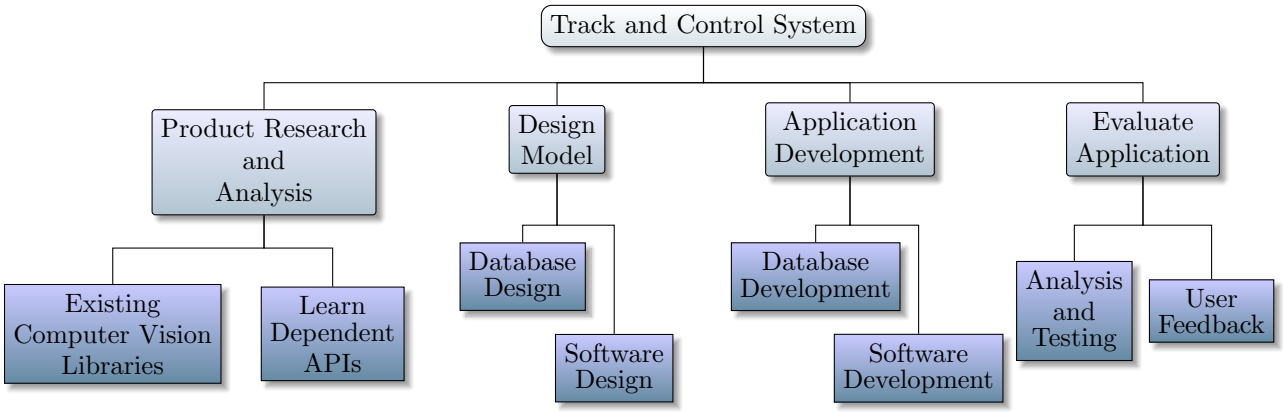
Given most laptops come standard will a built-in camera, the design behind TACS is safe. The modular design organizes the individual system features so they can be separately worked on.

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

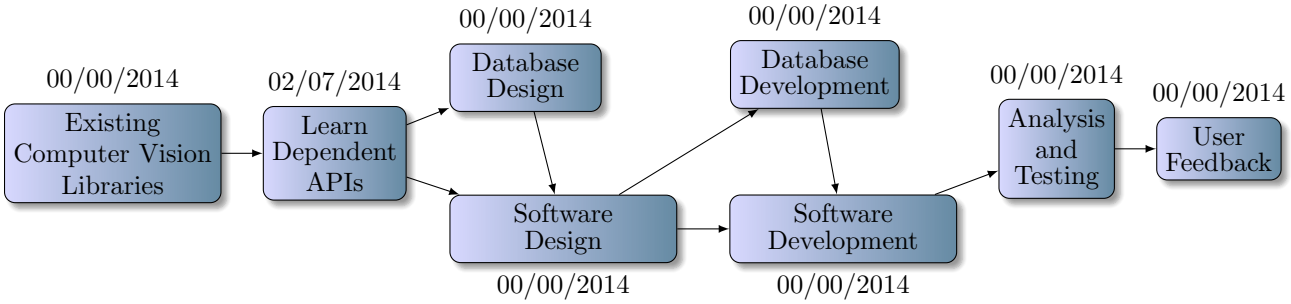
4.1 Work Breakdown Structure



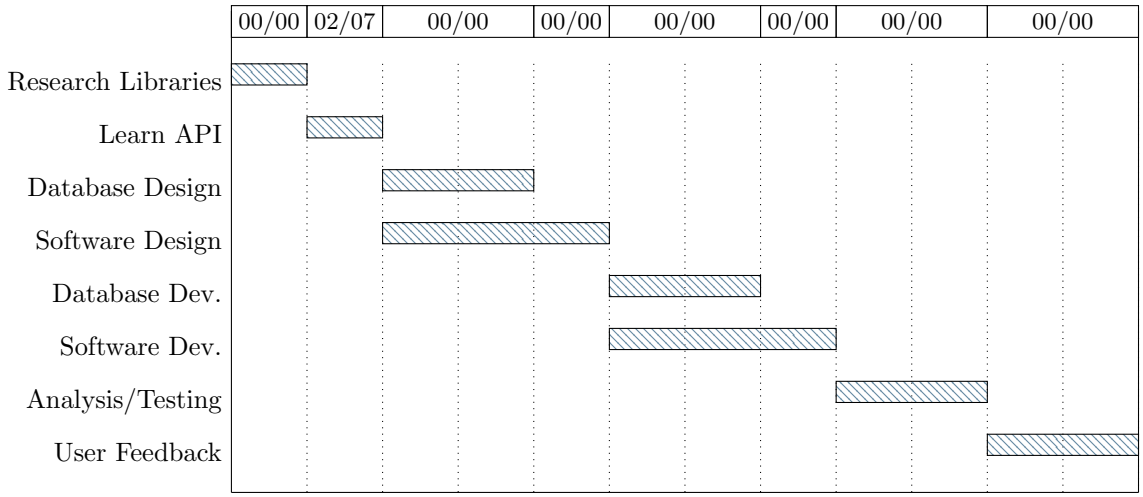
4.2 Time Estimation

Task	Weeks
Research Libraries	i
Learn API	i
Database Design	i
Software Design	i
Database Dev.	i
Software Dev.	i
Analysis/Testing	i
User Feedback	i
Total	**X**

4.3 Activity Sequencing Diagram



4.4 Gantt Chart



4.5 Risk Management

Risk	Likelihood	Consequences	Total	Fix
No camera	1	2	2	The user can use the mouse as a replacement for the head tracking.
Risk2	i	i	10-	i
Risk3	i	i	5-9	i
Risk4	i	i	1-4	i

A Appendix

A.1 Design Concept

A tentative conceptual design of the system is as followed: