Swift Arm of Justice

O R A N G S - 3D Robot Arm Simulator

Software Requirements Specification

Version 1.0

9/21/2012

U2

Brandon Kirby

Stephanie Chouinard

Peter O’Connor

Ryan Downey

Bobby Staples

Prepared for

Principles of Software Engineering

Fall 2012

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 9/23/12 | Version 1 | U2 | First Rev. and Submission |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Document Approval

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | Peter O’Connor | Point of Contact | 9/23/2012 |  |
|  | Ryan Downey | Team Member | 9/23/2012 |  |
|  | Bobby Staples | Team Member | 9/23/2012 |  |
|  | Stephanie Chouinard | Team Member | 9/23/2012 |  |
|  | Brandon Kirby | Team Member | 9/23/2012 |  |

**Table of Contents**

[**Revision History iii**](#id.gjdgxs)

[**Document Approval iii**](#id.30j0zll)

[**1. Introduction 1**](#id.1fob9te)

[1.1 Purpose - Complete 1](#id.3znysh7)

[1.2 Scope](#id.2et92p0) [- Complete](#id.3znysh7) [1](#id.2et92p0)

[1.3 Definitions, Acronyms, and Abbreviations](#id.tyjcwt) [- Evolving](#id.3znysh7) [1](#id.tyjcwt)

[1.4 References](#id.3dy6vkm) [- Evolving](#id.3znysh7) [1](#id.3dy6vkm)

[1.5 Overview](#id.1t3h5sf) [- Complete](#id.3znysh7) [2](#id.1t3h5sf)

[**2. General Description 2**](#id.4d34og8)

[2.1 Product Perspective](#id.2s8eyo1) [- Incomplete - Downey](#id.3znysh7) [2](#id.2s8eyo1)

[2.2 Product Functions](#id.17dp8vu) [- Incomplete - Downey](#id.3znysh7) [2](#id.17dp8vu)

[2.3 User Characteristics](#id.3rdcrjn) [- Incomplete - Downey](#id.3znysh7) [2](#id.3rdcrjn)

[2.4 General Constraints](#id.26in1rg) [- Incomplete - Downey](#id.3znysh7) [2](#id.26in1rg)

[2.5 Assumptions and Dependencies](#id.lnxbz9) [- Evolving - Downey](#id.3znysh7) [2](#id.lnxbz9)

[**3. Specific Requirements 2**](#id.35nkun2)

[3.1 External Interface Requirements 3](#id.1ksv4uv)

[*3.1.1 User Interfaces*](#id.44sinio) [- Complete](#id.3znysh7) [*3*](#id.44sinio)

[*3.1.2 Hardware Interfaces*](#id.2jxsxqh) [- Complete](#id.3znysh7) [*3*](#id.2jxsxqh)

[*3.1.3 Software Interfaces*](#id.z337ya) [- Complete](#id.3znysh7) [*3*](#id.z337ya)

[*3.1.4 Communications Interfaces*](#id.3j2qqm3) [- Complete](#id.3znysh7) [*3*](#id.3j2qqm3)

[3.2 Functional Requirements 3](#id.1y810tw)

[*3.2.1 <Forward Kinematics>*](#id.4i7ojhp) [- Complete](#id.3znysh7) [*3*](#id.4i7ojhp)

[*3.2.2 <Inverse Kinematics>*](#id.2xcytpi) [- Complete](#id.3znysh7)

[*3.2.3 <Zooming>*](#kix.c1u993ydaj4j) [- Complete](#id.3znysh7)

[*3.2.4 <Logging>*](#kix.93o5qcngpjxg) [- Complete](#id.3znysh7) [*3*](#kix.93o5qcngpjxg)

[3.3 Use Cases 3](#id.1ci93xb)

[*3.3.1 Standard forward kinematics input.*](#id.3whwml4) [Complete](#id.3znysh7) [*3*](#id.3whwml4)

[*3.3.2 Standard reverse kinematics input.*](#id.3whwml4) [Complete](#id.3znysh7) [*3*](#id.3whwml4)

[*3.3.3 Input of angles produces inaccessible point.*](#id.2bn6wsx) [Complete](#id.3znysh7) [*3*](#id.2bn6wsx)

*3.3.4 Input of point is inaccessible by manipulator.* [- Complete](#id.3znysh7)  [*3*](#id.3whwml4)

[*3.3.5.*](#id.3whwml4)[*Saving a single motion to file*- Complete](#id.3znysh7) [*3*](#id.3whwml4)

[*3.3.6 Loading and playback of replay data*](#id.3whwml4)  [- Complete](#id.3znysh7) [*3*](#id.3whwml4)

[3.4 Classes / Objects](#id.qsh70q) [- (Basic Descriptions)](#id.3znysh7) [4](#id.qsh70q)

[*3.4.1 Main.java / Driver Object 4*](#id.3as4poj)

[*3.4.2 Arm.java / Arm Object (abstract object)*](#id.1pxezwc)

[*3.4.3 Flange.java / Flange (extends Arm) 4*](#id.1pxezwc)

[*3.4.4 Waist.java / Waist interface*](#id.1pxezwc)[*(extends Kinematics) 4*](#id.1pxezwc)

[*3.4.5 Wrist.java / FwdKinematics Object (extends Arm) 4*](#id.1pxezwc)

[*3.4.6 Shoulder.java / FwdKinematics Object (extends Arm) 4*](#id.1pxezwc)

[*3.4.7 Elbow.java / FwdKinematics Object (extends Arm) 4*](#id.1pxezwc)

[*3.4.8 Grabber.java / FwdKinematics Object (extends Arm) 4*](#id.1pxezwc)

[*3.4.9 DataLogger.java / VerboseLog Object 4*](#id.1pxezwc)

[*3.4.10 ReplayLogger.java / Replay Object 4*](#id.1pxezwc)

[*3.4.11 ArmGUI.java / ArmGUI Object 4*](#id.1pxezwc)

[*3.4.12 3DRender.java / 3DRenderer 4*](#id.1pxezwc)

[*3.4.13 ControlPanelGUI.java / ControlPanelGUI Object 4*](#id.1pxezwc)

[*3.4.14 ReplayPanelGUI.java / ReplayPanelGUI Object 4*](#id.1pxezwc)

[3.5 Non-Functional Requirements 4](#id.49x2ik5)

[*3.5.1 Performance*](#id.2p2csry) [-](#id.3znysh7) Complete [*4*](#id.2p2csry)

[*3.5.2 Reliability*](#id.147n2zr) [-](#id.3znysh7) Complete [*4*](#id.147n2zr)

[*3.5.3 Availability*](#id.3o7alnk) [-](#id.3znysh7) Complete [*4*](#id.3o7alnk)

[*3.5.4 Security*](#id.23ckvvd) [-](#id.3znysh7) Complete [*4*](#id.23ckvvd)

[*3.5.5 Maintainability*](#id.ihv636) [-](#id.3znysh7) Complete [*4*](#id.ihv636)

[*3.5.6 Portability*](#id.32hioqz) [-](#id.3znysh7) Complete [*4*](#id.32hioqz)

[3.6 Inverse Requirements](#id.1hmsyys) [- Incomplete, when multiple inverse possibilities we will use first one calculated](#id.3znysh7) [4](#id.1hmsyys)

[3.7 Design Constraints](#id.41mghml) [- Complete](#id.3znysh7) [4](#id.41mghml)

[3.8 Logical Database Requirements](#id.2grqrue) [-](#id.3znysh7) Complete [4](#id.2grqrue)

[3.9 Other Requirements](#id.vx1227) [- Evolving](#id.3znysh7) [4](#id.vx1227)

[**4. Analysis Models 4**](#id.3fwokq0)

[4.1 Sequence Diagrams](#id.1v1yuxt) [- Complete](#id.3znysh7) [5](#id.1v1yuxt)

[4.3 Data Flow Diagrams (DFD)](#id.4f1mdlm) [- Complete](#id.3znysh7) [5](#id.4f1mdlm)

[4.2 State-Transition Diagrams (STD)](#id.2u6wntf) [- Complete](#id.3znysh7) [5](#id.2u6wntf)

[**5. Change Management Process**](#id.19c6y18) [- Complete](#id.3znysh7) [**5**](#id.19c6y18)

# 1. Introduction

## 1.1 Purpose

This Software Requirements Specification document, or SRS, will provide a clear and definite outline and layout of the functional and nonfunctional specifications for the Puma 560 Robot Manipulator that is to be developed by Group U2 for Principles of Software Engineering, CS 424/524, Fall 2012, Hood College. This software will emulate the use of the robot arm in such as way that allows the user to input both a set of angles for each arm or a point in space and determine the moves the arm will make, displaying them to user. This program will also allow the user to record their actions to replay later.

The intended audience of this document is Professor George Dimitoglou, professor of CS 424/524, the class for which this project is being developed.

## 1.2 Scope

The software product is a three dimensional robot arm simulator for the PUMA 560 robot. The program will model and animate the movement of the robot based on one of two things: joint angles specified by the user, or a user-entered point in space that the robot arm will navigate to. The robot simulator will alert the user if the angles or coordinates entered are out of the robot’s reach. The simulator will be able to “record” and play back a set of angle/coordinate instructions given, and will be able to save and load them for future use or playback. The simulator will also allow the user to change the dimensions and colors of the arm. A user will be able to run this product on a machine using Windows or Unix operating systems, and should be able to use it fully with two hours of training.

## 1.3 Definitions, Acronyms, and Abbreviations

|  |  |  |
| --- | --- | --- |
| *Term* | *Abbreviation (if one exists)* | *Definition* |
| *Forward Kinematics* | *FK* | *This is the process by which a user inputs angles for the arm joints and the calculation to find the point it reaches is performed.* |
| *Reverse Kinematics* | *RK* | *This is the process by which a user inputs the point for the arm to reach and the calculation to find the angles required to reach that point.* |
| *User Interface* | *UI or GUI* | *The is the graphical interface the user will directly access to view and input data.* |
| *Data Logger* | *DL* | *The data logger is a component of the software that records actions performed and simulations that run during the program run-time.* |
| *Kinematics Engine* | *KE* | *The kinematics engine is the core class that data is processed in and out of to/from the UI and the instance of Kinematics that is occuring.* |
| *Comma Separated Values* | *CSV* | *This is the file type (.csv) that the application will save position data for future replay.* |
| *Operating System* | *OS* | *This is the Operating System installed on the user’s computer* |
| *Gigahertz* | *GHz* | *This is the user’s processor speed* |
| *Sequence Diagram* | *SD* | *The sequence diagram for the application* |
| *Data Flow Diagram* | *DFD* | *The data flow diagram for the application* |
| *State Transition Diagram* | *STD* | *The state transition diagram for the application* |

## 1.4 References

1. Kumar, V. Introduction to Robot Kinematics. Obtained by Principle Investigator (Dr. D)

*This subsection should:*

*(1) Provide a complete list of all documents referenced elsewhere in the SRS, or in a separate, specified document.*

*(2) Identify each document by title, report number - if applicable - date, and publishing organization.*

*(3) Specify the sources from which the references can be obtained.*

*This information may be provided by reference to an appendix or to another document.*

## 1.5 Overview

The rest of this SRS contains a general description of the software, followed by all the functions it has and requirements it fulfills, as well as any dependencies or assumptions made during the development process. It also includes a profile of the software’s prospective users and how the requirements change to accommodate them. There are lists of specific requirements, external interface requirements, functional requirements, use cases, and the software’s classes and objects. Also listed are the nonfunctional requirements, design constraints, analysis models, and the process to make and submit changes to the SRS (i.e.: when a requirement changes, etc). There are appendices with additional software and SRS information at the end of the document.

# 2. General Description

*This section of the SRS should describe the general factors that affect 'the product and its requirements. It should be made clear that this section does not state specific requirements; it only makes those requirements easier to understand.*

## 2.1 Product Perspective

*This project is a completely self contained system and does not interface with any other internal or external systems.*

## 2.2 Product Functions

*2.2.1. The application should have a user-friendly and appealing GUI.*

*2.2.2. The application should be based on a 6 DOF PUMA 560 robot manipulator.*

*2.2.3. The application should enable the user, given the joint angles and link lengths of the*

*robot arm to determine the orientation and position of the end eﬀector coordinates.*

*2.2.4. The application should enable the user, given the orientation and position of the end*

*eﬀector coordinates to determine the joint angles.*

*2.2.5. The application should enable the user to change the link lengths of the manipulator.*

*2.2.6. The application should enable the user to change the link colors of the manipulator.*

*2.2.7. The application should move the manipulator in real time (for forward kinematics).*

*2.2.8. The application should enable the user to select a point in space for the manipulator to*

*reach (for inverse kinematics).*

*2.2.9. The application should enable the user to change the speed of execution (for inverse*

*kinematics).*

*2.2.10. The application should enable the user to zoom-in and zoom-out.*

*2.2.11. The application should provide both a textual and a graphical/animated representation of what is happening during execution.*

*2.2.12. The application should clearly indicate to the user when activities begin and end.*

*2.2.13. The application should provide detailed logging about the state of each shape, the overall*

*environment, information about properties and activities at each increment of execution.*

*2.2.14. The application should provide text file logging capability so users can retrieve the results*

*and trace of their execution.*

*2.2.15. The application should be able to replay executed application runs.*

*2.2.16. The application should be developed so it is platform independent (MS-Windows, Unix,*

*Apple is optional)*

*2.2.17. The application should be developed in such a way that the rendering engine and the*

*GUI are independent, allowing developers to use other GUIs*

## 2.3 User Characteristics

The user should have a general understanding of using GUIs.

## 2.4 General Constraints

*The customer has provided any other constraints.*

## 2.5 Assumptions and Dependencies

2.5.1: The robot is sitting on a surface (similar to a table) that the arm will not be able to reach below

2.5.2: The object that the robot will “grab” in space (when the user specifies its coordinate) is a sphere

2.5.3: The software will be installed and run on a computer that satisfies all the conditions listed in the hardware interfaces section (3.1.2).

# 3. Specific Requirements

*This will be the largest and most important section of the SRS. The customer requirements will be embodied within Section 2, but this section will give the D-requirements that are used to guide the project’s software design, implementation, and testing.*

*Each requirement in this section should be:*

* *Correct*
* *Traceable (both forward and backward to prior/future artifacts)*
* *Unambiguous*
* *Verifiable (i.e., testable)*
* *Prioritized (with respect to importance and/or stability)*
* *Complete*
* *Consistent*
* *Uniquely identifiable (usually via numbering like 3.4.5.6)*

*Attention should be paid to the carefuly organize the requirements presented in this section so that they may easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.*

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

The user interface will have at least:

* Simple User Interface (performance noted in Secondary Requirements)
* Animated 3D Robot Arm
* User will be able to input points to find angles
* User will be able to click on sliders to alter robot arm position and find new end point
* User will be able to change robot arm dimensions
* User will be able to save and load data for replay usage

### 3.1.2 Hardware Interfaces

|  |  |
| --- | --- |
| Specification | Requirement |
| OS | Windows, Linux |
| Memory | >40MB + Memory assets |
| CPU | 1 GHz |
| Graphics Card | ATI Radeon 9500, NVIDIA GeForce 5 FX, Intel GMA 4500, or better supporting OGL 2.0 or better (latest driver recommended) |

### 3.1.3 Software Interfaces

* User will need to have Java 6 or later running on their OS.

### 3.1.4 Communications Interfaces

* User will have no extra need for communications hardware.

## 3.2 Functional Requirements

### 3.2.1 <Forward Kinematics>

3.2.1.1 Introduction

The simulator should enable the user to input both link lengths and link colors at start-up. It should also allow for the user to specify joint angles and orientation, as well as starting position.

3.2.1.2 Inputs

* Link Lengths
* Link Colors
* Joint Angles/Orientation
* Starting position in space
* All points are changed via scrolling bars

3.2.1.3 Processing

Each movement of the scrolling bars changes the way that the GUI looks. This is done by a reoccurring, step-wise process.

3.2.1.4 Outputs

The GUI should reflect the users choices and show the robot as specified (In other words, move the robot manipulator in real time).

The simulator will also determine and show the coordinates of the end effectors of each joint, as well as their angles.

3.2.1.5 Error Handling

The simulator will change angle maximum’s and minimum’s according to link lengths and other joint angles in the robot to best account for and stop potential non-allowable situations (such as the arm hitting the ground).

### 3.2.2 <Inverse Kinematics>

3.2.2.1 Introduction

The simulator should allow for the user to select a point in space for the robot arm to reach. The user will also be allowed to change the execution speed of the process.

3.2.2.2 Inputs

* Point in space (x, y, z)
* Execution Speed

3.2.2.3 Processing

The simulator finds a way to get to a user specified point in space via an algorithm.

3.2.2.4 Outputs

The simulator will show both a graphical and a textual representation of what is happening during execution. It will clearly show when the activity begins and ends.

3.2.2.5 Error Handling

The simulator will tell the user if the point cannot be reached after the algorithm has finished. The simulator will also make sure that the point specified is not within restricted bounds.

### 3.2.3 <Zooming>

3.2.3.1 Introduction

The simulator will allow the user to zoom in and out to the GUI.

3.2.3.2 Inputs

* Magnification (Slider)

3.2.3.3 Processing

GUI changes in a state by state basis as the user changes the magnification.

3.2.3.4 Outputs

Certain aspects of the GUI get bigger and smaller as the user changes the magnification.

3.2.3.5 Error Handling

The simulator will have maximum and minimum value magnifications.

### 3.2.4 <Logging>

3.2.4.1 Introduction

The simulator will provide detailed logging about the state of each part of the arm, the overall

environment, information about properties and activities at each increment of execution. The application should provide text ﬁle formatted in a CSV format logging capability so users can retrieve the results and replay execution. The application should be able to replay executed application runs.

3.2.4.2 Inputs

* Whether or not the user would like to log their execution.
* Takes in a replayed execution run when the user wants.

3.2.4.3 Processing

Processes the execution of statements to the program and records them, allowing for reuse later.

3.2.4.4 Outputs

If wanted, the simulator will record the detailed logging information in a text file for reuse later. Upon reuse, the simulator should output the graphical and textual animation of the replay execution.

3.2.4.5 Error Handling

The simulator will make sure that replay files are usable for reuse, and records the information in a good enough way.

## 3.3 Use Cases

### 3.3.1 Standard Forward Kinematics Input

A user will open the application, then will enter an angle for one or more of the arm segments, using the graphical sliders on screen. After this, the user will press the “Go” button located in the same panel area as the slider buttons. Now, the user will see the arm shown in the application move to match the criteria that they entered. They will also see the values in the XYZ Coordinates area change to match the new end point of the robot manipulator.

**3.3.2 Standard Inverse Kinematics Input**

A user will open the application, then will enter a valid XYZ coordinate using the three text fields on the application, labelled as such, “X”, “Y”, and “Z”. Then, the user will see the robot manipulator move to reflect the values that they entered into the coordinate area. They will also see the sliders on the bottom of the application move to reflect the new angles at which each arm of the manipulator now resides.

**3.3.3 Input of Angles Produces Inaccessible Point**

The user will open the application, then will attempt to enter an invalid series of angles using the sliders on the GUI. They will not be allowed to by the application because each of the sliders will change its maximum and minimum values as any one of them is moved.

**3.3.4 Input of Point is Inaccessible by Manipulator**

A user will open the application, then will enter an invalid XYZ coordinate using the three text fields on the application, labelled as such, “X”, “Y”, and “Z”, and then hit the “Go” button located in the same area of the GUI. Then the user will see an error prompt making them aware that the point that they have entered is invalid and unreachable. They will then exit that prompt, and be returned to the previous valid state of the manipulator.

**3.3.5 Saving A Single Motion to File**

A user will open the application and will enter either a valid XYZ coordinate. The user will then check the “Record” check box, and then hit the “Go” button located in the same area of the GUI. Then the user will hit the “Dump to File” button and the program will output to the working directory, a file that is usable as a replay file in this simulator, replaying the actions which they saved to the file.

**3.3.6 Loading and Playback of Replay Data**

(In this use case, it is assumed that a valid replay file exists in the working directory.)

The user will open the application, then will click on a file in the “Files” area in the GUI. They will then click the “Load” button. The user will then hit the “Play” button, and the actions that were stored in that replay file will be played out in the application, by the image of the manipulator moving, as well as the sliders and the numeric text boxes for XYZ coordinates. There will be one second between each motion.

## 3.4 Classes / Objects

### 3.4.1 Main.java / Driver

3.4.1.1 Attributes:

1. JFrame - MainFrame
2. JPanel - ControlPanel
3. JPanel - RecorderPanel
4. Arms[] - armpieces
5. int - height
6. int - width

3.4.1.2 Functions

1. Coordinates the GUI with the Kinematics Engine

### 3.4.2 Arm.java / Arm (abstract object)

3.4.2.1 Attributes:

1. Length of Arm
2. Angle of Arm
3. X & Y of Arm End Point

3.4.2.2 Functions:

1. Calculates relative angle of arm if needed
2. Calculates the XYZ of the arm given a angle

### 3.4.3 Flange.java / Flange (extends Arm)

3.4.3.1 Attributes:

1. Min and Max Angle Movement

3.4.3.2 Functions:

1. Calculates relative angle of arm if needed
2. Calculates the XYZ of the arm given a angle
3. Paint itself onto panel.

### 3.4.4 Waist.java / Waist (extends Arm)

3.4.4.1 Attributes:

1. Min and Max Angle Movement

3.4.4.2 Functions:

1. Calculates relative angle of arm if needed
2. Calculates the XYZ of the arm given a angle
3. Paint itself onto panel.

### 3.4.5 Wrist.java / Wrist (extends Arm)

3.4.5.1 Attributes:

1. Min and Max Angle Movement

3.4.5.2 Functions:

1. Calculates relative angle of arm if needed
2. Calculates the XYZ of the arm given a angle
3. Paint itself onto panel.

### 3.4.6 Shoulder.java / Shoulder (extends Arm)

3.4.6.1 Attributes:

1. Min and Max Angle Movement

3.4.6.2 Functions:

1. Calculates relative angle of arm if needed
2. Calculates the XYZ of the arm given a angle
3. Paint itself onto panel.

### 3.4.7 Elbow.java / Elbow (extends Arm)

3.4.7.1 Attributes:

1. Min and Max Angle Movement

3.4.7.2 Functions:

1. Calculates relative angle of arm if needed
2. Calculates the XYZ of the arm given a angle
3. Paint itself onto panel.

### 3.4.8 Grabber.java / Grabber (extends Arm)

3.4.8.1 Attributes:

1. Min and Max Angle Movement

3.4.8.2 Functions:

1. Calculates relative angle of arm if needed
2. Calculates the XYZ of the arm given a angle
3. Paint itself onto panel.

### 3.4.9 DataLogger.java / VerboseLog

3.4.9.1 Attributes:

1. File To be written/read from
2. Data to be written

3.4.9.2 Functions:

1. Writes to file
2. Opens File
3. Closes File

### 3.4.10 Replay.java / Replay

3.4.10.1 Attributes:

1. File to be replayed

3.4.10.2 Functions:

1. Returns replay file information in an ArrayList.
2. Turn recording on and off
3. Accept data from the Arm object about functions ongoing.

### 3.4.11 ArmGUI.java / ArmGUI

3.4.11.1 Attributes:

1. Height and Width of Panel
2. Arm[] of arms
3. Houses the 3D Renderer

3.4.11.2 Functions:

1. Draws the Arm
2. Animates the arm through the process.

### 3.4.12 3DRenderer.java / 3DRenderer

3.4.12.1 Attributes:

1. Houses JMonkey implementation

3.4.12.2 Functions:

1. Rotates image
2. Zooms image
3. Resizes image

### 3.4.13 ControlPanel.java / ControlPanel

3.4.13.1 Attributes:

1. 6 JSlider(s) for Angles
2. 3 JTextField(s) for XYZ
3. 2 JCheckBox(s) for recording options
4. JButton for “Go” on XYZ

3.4.13.2 Functions:

1. Triggers arm changeAngle function.
2. Triggers repainting of ArmGUI
3. Triggers updating of sliders
4. Triggers XYZ update with new angle
5. Triggers Angle update with clicking of “Go” button
6. Triggers Recording/Stopping of Replay Object

### 3.4.14 ReplayPanelGUI.java / ReplayPanelGUI

3.4.14.1 Attributes:

1. JFileChooser of Replay files
2. JButton for “Load”
3. JButton for “Stop”

3.4.14.2 Functions:

1. Loads values to Main and then into Arm.
2. Triggers repaint of Arm

## 3.5 Non-Functional Requirements

*Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, > 30 day MTBF value, etc).*

### 3.5.1 Performance

The simulator must change states within, at most, 5 seconds of changing an angle or link length via a slider. Inverse Kinematics should be calculated and shown within a window of 20 seconds.

### 3.5.2 Reliability

The simulator should offer precise and accurate results for every calculation and record that it makes 99.9% of the time. This window accounts for potential crashes as well.

### 3.5.3 Availability

The simulator will be available for use in both Linux and Windows.

### 3.5.4 Security

Considering the fact that the simulator does not use a database of any kind, and no personal information is required or even mentioned, security is not really needed. The only security needed will be that of personal need when protecting replay files.

### 3.5.5 Maintainability

The modules of the simulator will be created and maintained until the deadline of the project. Prior to December 12, 2012, any maintenance performed will not be the responsibility of the members of U2.

### 3.5.6 Portability

The project will be made compatible with both Windows and Linux OS’s. No extra portability will be made in initial development.

## 3.6 Inverse Requirements

*State any \*useful\* inverse requirements.*

## 3.7 Design Constraints

3.7.1 To be supported on Windows and Linux OS.

3.7.2 Has to have a soft installation (i.e. not .exe)

3.7.3 Bases on 6 DOF PUMA 560 robot manipulator specifications.

## 3.8 Logical Database Requirements

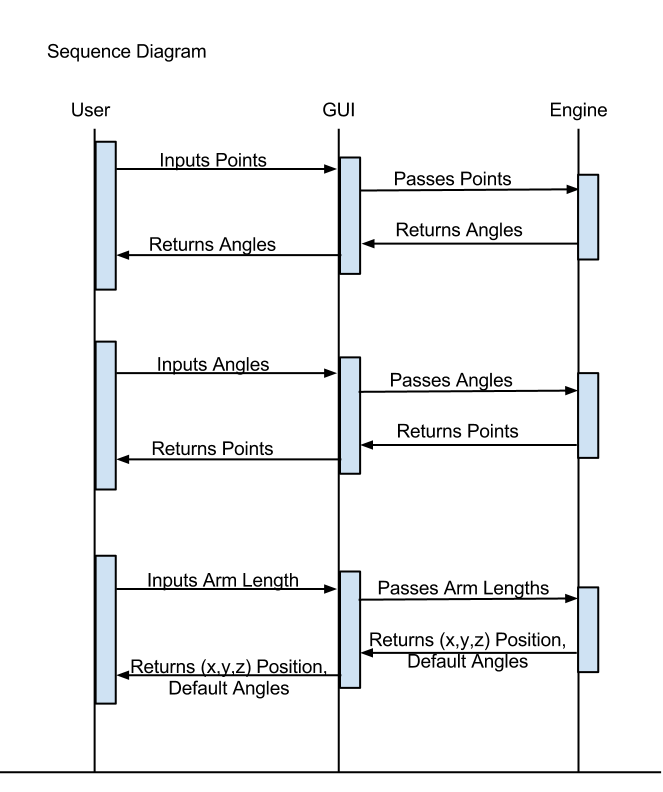
This program will not use a database; it will save all the data it needs in a formatted CSV file (.csv). It will save the coordinates, angles, and lengths for playback as the user specifies.

## 3.9 Other Requirements

*Catchall section for any additional requirements.*

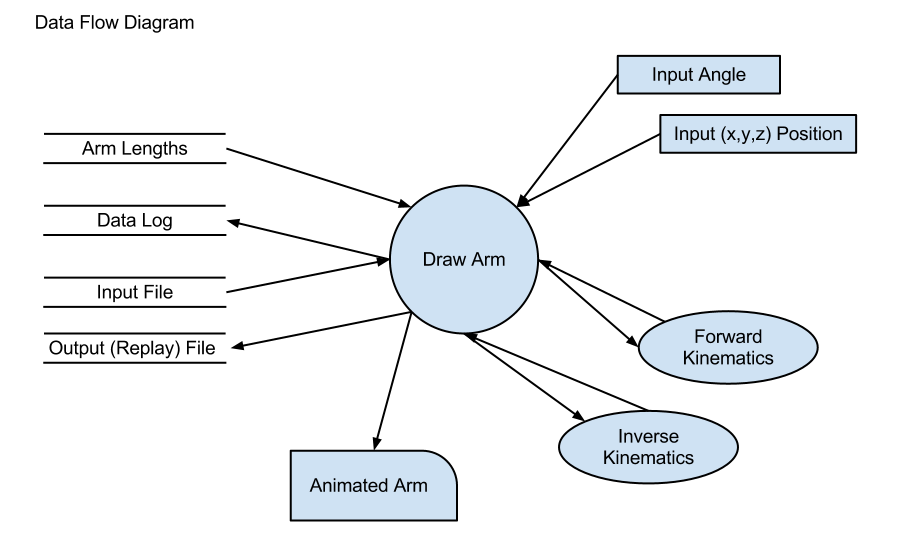
# 4. Analysis Models

## 4.1 Sequence Diagrams (SD)



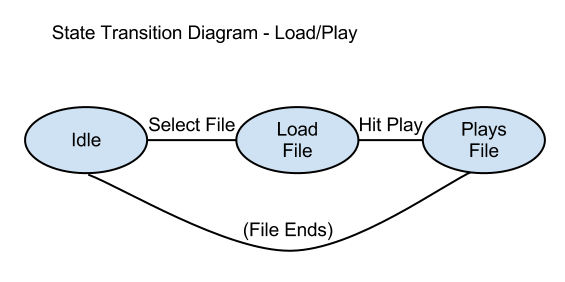
This sequence diagram shows three possible user choices upon application startup. The first is if they choose to input the coordinate point in the boxes. The user will input the desired point and click “Go”, at which time the points are passed through the GUI to the engine for validation and angle calculation. If the point is valid (is reachable by the arm), the angles will be calculated and returned to the GUI and user. The next choice is to use the sliders to input new angles for the arm. After the mouse release of each slider input, the angles will be passed through the GUI to the engine which will then calculate the new coordinate points and updated angle ranges for the remaining sliders. The information is then passed to the GUI, giving the sliders a new range and updating the point listed in the boxes. The last option is to change a dimension in the arm. The user will input the new dimension(s) in the appropriate boxes and click “Save”. The information is passed from the GUI to the engine, which resets all the angles to the default (adjusting sliders accordingly) after adjusting the proper dimensions, and returns the new coordinate to the GUI boxes and hence to the user.

## 4.3 Data Flow Diagrams (DFD)

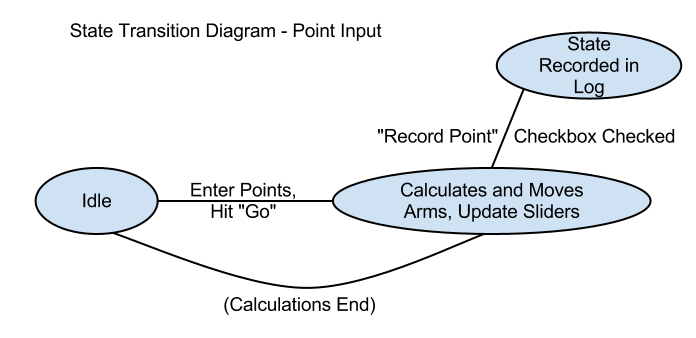


The arm is dependent on the arm lengths. The arm animation is dependent on the input angles and position that the user inputs. Forward and inverse kinematics both receive information from the arm and give it back. The data log is a continuous log of anything and everything (startup, shutdown, any points visited / angles set, errors, etc.) and will continue to receive data from the program. Output files are saved from the arm drawing steps that the user selects. Input files are loaded into the program for animation replay after being saved in the program. The final result that the user sees is the animated arm that accepts these inputs and gives back these outputs.

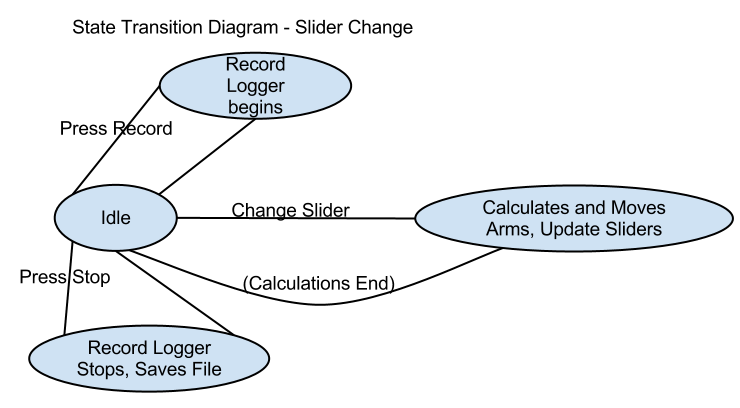
## 4.2 State-Transition Diagrams (STD)



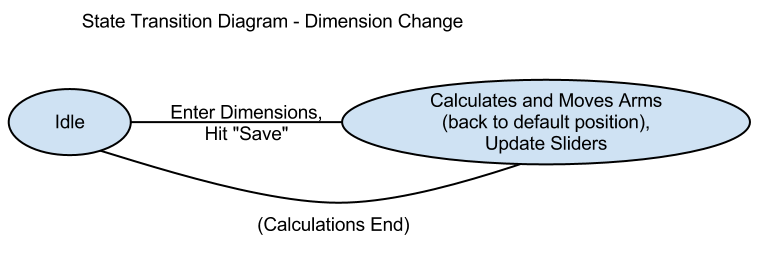
The application begins at idle; the user will select a file to load, and hit play. The animation will play, and when the file ends, the application will return to idle.



The application will begin at idle; the user will enter new coordinate points and click “Go”. The application will calculate the angles for the arm and update the slider ranges. It will record the arm’s “state” in the replay log if the checkbox for recording the point is checked. Once the calculations are done, the arm will animate to its new position and then the application will return to idle.



The application begins at idle, if the user presses record then any changes to the sliders will be recorded until they click stop and the logger will save the data. Once the user changes a slider, the new position will be calculated, the remaining slider ranges will be updated, the arm will animate and then the application will return to idle.



The application begins at idle, the user will enter new dimension(s) in the appropriate boxes and click “Save”, when the application will reset the arm angles and animate it back with the entered dimensions to the new position. The sliders will also be updated with new ranges, and the application will return to idle.

# 5. Change Management Process

For any changes to this document, members of U2 will communicate the concern or change via e-mail to all other members and wait for a response from all. The change will be considered accepted if at least three out of the five members agree with it and there are no major objections. The change will be added to the document by the person who initiates it.