# Overview

This document includes an initial detailed description of the system architecture. This includes a structural view of the major components, their connections, and a justification for using. It also contains a detailed description of the system’s behavior with textual use cases and activity diagrams. We also have included our initial mock-ups of the user interface for both the client and server systems. Finally we have included our tentative development plan, which includes a general description of our iterations, a description of our configuration management system, team member roles and responsibilities, key milestones of the project, and an initial cut at our meeting schedule.

# Structural View



# Behavioral View

## Use Case Diagrams

|  |  |  |
| --- | --- | --- |
| Name: | **Create Session** | |
| Actors: | User (initiates) | |
| Pre-Conditions: | User signed in on MultiDraw | |
| Flow of Events: | **Actor Action** | **System Response** |
| 1) In the session pane the user selects “Create Session” from file menu. | New session created(canvas, session log, etc) |
| Post-Conditions: | New session has been created | |
| Quality Requirements: | Session should persist without impacting current live sessions | |

|  |  |  |
| --- | --- | --- |
| Name: | **Leave Session** | |
| Actors: | User (initiates) | |
| Pre-Conditions: | User is in a session | |
| Flow of Events: | **Actor Action** | **System Response** |
| 1) User clicks on change session | User is transferred to newly selected session |
| Post-Conditions: | User has left old session and is in new session | |
| Quality Requirements: | No sessions have been changed except transferring the user | |

|  |  |  |
| --- | --- | --- |
| Name: | **Join Session** | |
| Actors: | User (initiates) | |
| Pre-Conditions: | User must be logged in | |
| Flow of Events: | **Actor Action** | **System Response** |
| 1) User clicks the join session button | User is transferred to selected session |
| Post-Conditions: | User is in selected session | |
| Quality Requirements: | No sessions have been changed except transferring the user | |

|  |  |  |
| --- | --- | --- |
| Name: | **Pass Drawing Control** | |
| Actors: | User (initiates) | |
| Pre-Conditions: | User has to be in control of drawing session, and has selected a recipient of the drawing control | |
| Flow of Events: | **Actor Action** | **System Response** |
| 1) User clicks give control button | Selected user gains control of the drawing canvas |
| Post-Conditions: | Selected user now has exclusive control of canvas | |
| Quality Requirements: | Only one user may have drawing control per session | |

|  |  |  |
| --- | --- | --- |
| Name: | **Receive Drawing Control** | |
| Actors: | User (initiates) | |
| Pre-Conditions: | User must not have drawing control and must be in the session | |
| Flow of Events: | **Actor Action** | **System Response** |
| 1) User accepts drawing control | User is given drawing control |
| Post-Conditions: | Only the user has control | |
| Quality Requirements: | Only one user may have drawing control per session | |

|  |  |  |
| --- | --- | --- |
| Name: | **Save Drawing** | |
| Actors: | User (initiates) | |
| Pre-Conditions: | User must be owner of session | |
| Flow of Events: | **Actor Action** | **System Response** |
| 1) In canvas pane user selects “Save” from the file menu | System stringifies objects on canvas then stores on server in a file |
| Post-Conditions: | File exists on server | |
| Quality Requirements: | File must persist | |

|  |  |  |
| --- | --- | --- |
| Name: | **Open Drawing** | |
| Actors: | User (initiates) | |
| Pre-Conditions: | User must be logged in as owner of the drawing | |
| Flow of Events: | **Actor Action** | **System Response** |
| 1. In canvas pane, user selects “Open” in file menu 2. Users selects drawing to open | Session is created with the opened drawing as the canvas |
| Post-Conditions: | Session now contains opened canvas | |
| Quality Requirements: | Opened drawing has not changed during storage/retrieval | |

## Activity Diagrams















# GUI Mock-ups













# Development Plan

## Iteration Plan

**Iteration 0 -** Architecture Model and Development Plan

**Expected Completion Date:** 04/04/2011

**Objective:** Produce the initial plans for developing the *MultiDraw* application.

**Iteration 1 –** Saving and Opening Drawings

**Expected Completion Date:** 04/10/2011

**Objective:** Provide single user capability to open and close drawings. This will involve serializing the canvas data in some way and saving it to a file. Our initial plan is to incorporate an xml library into our project.

**Iteration 2 –** Create Network Architecture

**Expected Completion Date:** 04/17/2011

**Objective:** Create the architecture to allow network access. This will allow others to view the drawing canvas that a document owner is working on. Other users will not be able to modify the canvas, but will be able to see the owner’s modifications in real time.

**Iteration 3 –** Passing of Control

**Expected Completion Date:** 04/24/2011

**Objective:** Add the capability of users other than the owner to make modifications to the canvas. The owner will initiate this by passing a user in the session control. The owner will still maintain ownership of the canvas.

**Iteration 4 –** Tool Design/Dynamic Class Loading

**Expected Completion Date:** 05/01/2011

**Objective:** Add the capability of users to add new tools and share them across their sessions. This way all users will be able to use these new tools. This will involve dynamic class loading for the creation of functionality at runtime.

**Iteration 5 –** Testing and Clean up and Documentation

**Expected Completion Date:** 05/06/2011

**Objective:** We will use this final week for any refactoring needed and testing network loads, security, and that our application is platform independent. We will also prepare our final documentation.

## Configuration Management

We will be using a Git repository for our versioning system. We will be tagging iterations so that the state of our iterations is preserved in case any new functionality needs to be reverted. This will make the adding of features easier as well making so that we don’t have to merge branches.

## Assignments and Responsibilities

For the project, our team is well rounded and will be sharing the responsibilities of coding and documentation. Since we are using Git, we will be able to simultaneously code and document files. This will improve the rate at which our iterations get completed. Cyrus Coffman will be acting as our team’s coordinator and will write weekly status updates for the team. Team members will keep track of their individual performance in a personal log.

## Meeting Schedule

We plan to hold weekly meetings on Sunday nights to discuss the results of the previous week’s iteration and to also split up tasks and create a plan of attack for the next week’s iteration. These meetings may be brief or can last several hours depending on the difficulties that arose during a particular iteration.