# SOFTWARE ARCHITECTURE DOCUMENT

# Version 1.0

## LIQUID: 2-D FLUID DYNAMIC SIMULATOR

## TEAM PRYMM

Pavan Kumar Gade, Radhika Panchal, Yashraj Sinha, Minghua Liu and Manoj Mathe

# SAD Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 10/13/2015 | 1.0 | Initial version of Software architecture Document based on 4+1 Architectural model. | Team PRYMM<br>Individual contribution along with respective section number :<br>1. Yashraj, Minghua and Radhika<br>2. Yashraj, Radhika and Minghua<br>3. Yashraj and Minghua<br>4. Minghua and Yashraj<br>5. Minghua and Yashraj<br>6. Minghua and Yashraj<br>7. Manoj and Pavan<br>8. Manoj and Pavan<br>9. Radhika, Pavan and Manoj |
| | | | |

# Table of Contents

# 1. Introduction

This document comes as an architecture design document for the project LIQUID: 2-D Fluid Dynamic Simulator. In this document the high level structures of project LIQUID, the discipline of creating such structures, and the documentation of these structures. These are the set of structures needed to reason about the software system

| Architectural activities | Software Architecture Document |
|---|---|
| Step 1 - Identify and prioritize significant Use-Cases | Section 4 |
| Step 2 - Define the candidate architecture | Section 2,3, 5.1, 9 |
| Step 3 - Define the initial Deployment Model | Section 6 |
| Step 4 - Create an Analysis Model | Section 5 |
| Step 5 - Create the Design Model | Section 5 |
| Step 6 - Document concurrency mechanisms | Section 6, 7 |

## 1.1 Purpose

The Software Architecture Document (SAD) provides a comprehensive architectural overview of the project LIQUID: 2-D Fluid Dynamic Simulator. It presents a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

In order to depict the software as accurately as possible, the structure of this document is based on the "4+1" model view of architecture.

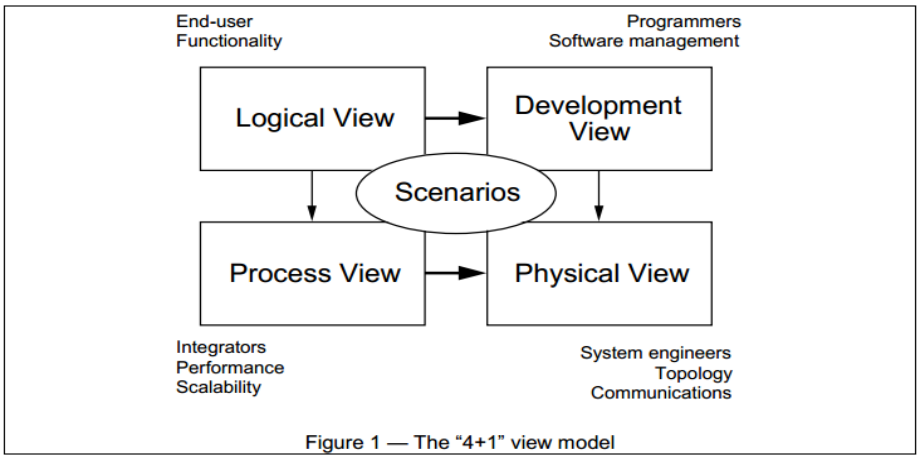



Figure 1 — The "4+1" view model

The "4+1" View Model allows various stakeholders to find what they need in the software architecture.

### 1.2 Scope

The scope of this Software Architecture Document is to depict the architecture of the project LIQUID: 2-D Fluid Dynamic Simulator. This document provides the high level structure of the project to its stakeholders.

### 1.3 Definitions, Acronyms and Abbreviations

SAD: Software Architecture Document

QoS: Quality of Service

MVC: Model View Controller

GUI: Graphical User Interface

JAVA SE: JAVA Standard Edition

### 1.4 References

[1] "ISO/IEC/IEEE 42010", website: http://www.iso-architecture.org/42010/

[2] "4+1 model Wikipedia", website: https://en.wikipedia.org/wiki/4%2B1_architectural_view_model

### 1.5 Overview

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections.

Section 2: describes the use of each view

Section 3: describes the architectural constraints of the system

Section 4: describes the functional requirements to be considered for architecture design

Section 5: describes the most important Analysis Model and the Design Model

Section 6: describes the layers and subsystems of the application

Section 7: describes the process view of the architecture

Section 8: describes the physical view of the project

Section 9: describes any aspects related to the quality of service (QoS) attributes

## 2. Architectural Representation

This document details the architecture using the views defined in the "4+1" mode. The views used to document the project LIQUID: 2-D Fluid Dynamic Simulator are:

### Logical view

**Audience**: Architecture Designers, Developers, Testers and Project Management team
**Area:** Functional Requirements: describes the design's object model. Also describes the most important use-case realizations.
**Related Artifacts:** Design model

### Process view

**Audience:** Integrators and testers
**Area:** Non-functional requirements: describes the design's concurrency and synchronization aspects.

**Related Artifacts:** (no specific artifact).

### Development view

**Audience**: Developers
**Area**: Non-functional requirements: describes the design's concurrency and synchronization aspects.
**Related Artifacts**: (no specific artifact).

### Physical view

**Audience:** Not applicable to this project
**Area**: This view is related to physical layer of the software and hence does not apply to this project
**Related Artifacts**: (no specific artifact).

### Use Case view

**Audience:** all the stakeholders of the system, including the end-users.
**Area:** describes the set of scenarios and/or use cases that represent some significant, central functionality of the system.
**Related Artifacts:** Use-Case Model

## 3. Architectural Goals and Constraints

This section describes the software requirements and objectives that have some significant impact on the architecture. This section also includes the constraints applied on the design and development.

### 3.1 Technical Platform

The application shall be developed as an installable software on MS-Windows platform using Java SE.

### 3.2 Persistence

Data persistence has to be addressed using a log file in the file system.

### 3.3 Reliability/Availability

The availability of the system is a key requirement by nature, hence the application shall be responsive at all times during the simulation. Targeted reliability is MTBF of 100 minutes for the software .Considering an average simulation execution time of 5 minutes, application shall work for 20 cycles without crashing.

### 3.4 Performance

There are set of non-functional constraints related to performance which are considered to design the software architecture.

- The Frame rate of the fluid simulator should be at least 50fps.
- The response time for user to retrieve the log from the system should not be more than 25 seconds.
- The response time for the system to start replay from the entered log file should not exceed more than 30 seconds
- The software should log the data at every one second into the log file.
- The log file of the software should not exceed 5MB.
- The system should not allow user to import a file which is larger than 5MB.

### 3.5 Portability

The application is a portable installable for MS-Windows environment.

### 3.6 Other Constraints

Along with above mentioned constraints two other constraints which impacts the the architecture design of the project are:

- The application shall be developed in such a way that the rendering engine and the GUI are independent. Allowing developers to use other GUI's created on the same operating system (but not necessarily with the same language) with minimal integration.
- The software can include only one third party component.

## 4. Scenarios (Use-Case View)

This section covers scenarios from the use-case model. This section represents significant, central functionality of the system. The two central functionality of the project LIQUID: 2-D Fluid Dynamic Simulator are the fluid simulator and the logging functionality. This has been illustrated by use of a use-case diagram.

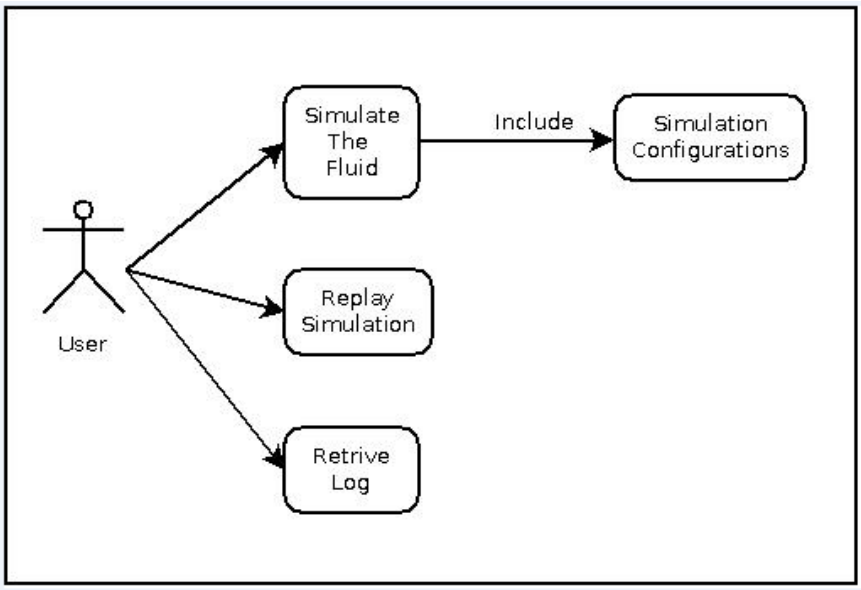



Figure – 2 User Case diagram showing major central system functionalities

## 5. Logical View

### 5.1 Overview

As the system is a small installable desktop application MVC pattern is extended to cover the whole system rather than just GUI. This MVC pattern is used to break down the system, based on the responsibility handled by different units/components of the application.

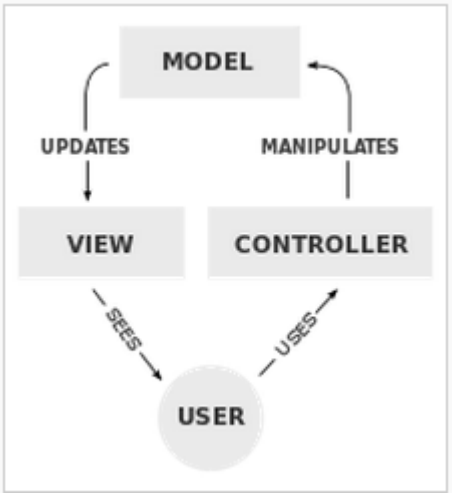



Figure – 3 highlighting the components of MVC pattern

**Controller**: Is used to send commands to the model component in order to update model state. This includes updating configuration parameters, placing flow meters, starting simulation, stopping simulation etc.

**Model:** Stores and updates data passed by controller. Along with the configuration parameters it also stores data getting generated during simulation to provide logging capability.

**View:** This component of the MVC pattern provides the output presentation to the user based on the changes occurred in model component. This includes graphical representation in rectangular simulation box as well as textual in form of values at flow meters and log file.

This strategy has been chosen because it isolates various system responsibilities from one another, so that it improves both system development and maintenance by keeping low coupling and high cohesion.

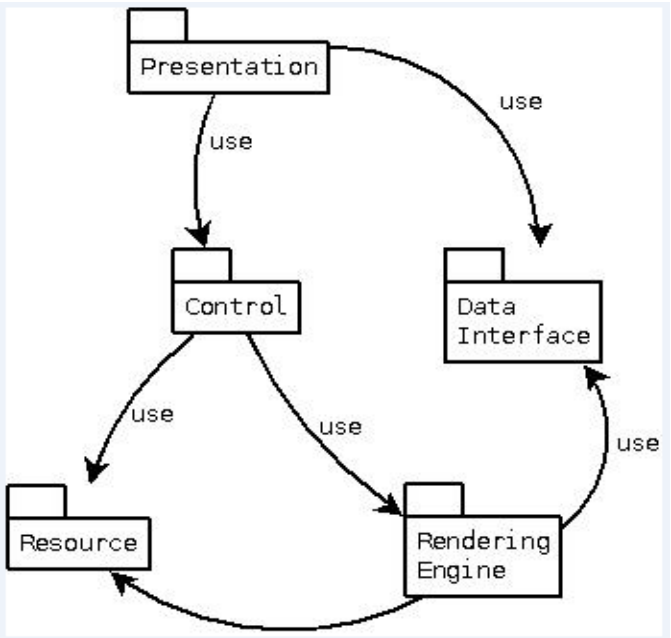



Figure – 4 Architectural layer Dependencies of project LIQUID

Each layer has specific responsibilities.

- The **presentation layer** deals with the presentation logic of the simulation data.
- The **control layer** manages the access to the resource layer and rendering engine layer.
- The **resource layer** is responsible for the access to the logged data during simulation run.
- The **Rendering Engine layer** is related to the calculation logic and manages the accesses to the resource layer.
- The **Data Interface layer** gathers the common objects reused through all the layers.

## 5.2 Class diagram

This section covers classes and functions. The class diagram in Figure – 5 below is a static structure diagram that describes the structure of a system by showing the system's classes, their attributes, methods, and the relationships among objects. The classes covered in this section will have two units in it attributes and functions. Attributes sub-section will have data or properties of the object whereas the other sub-section which is function will contain methods to operate on attributes.

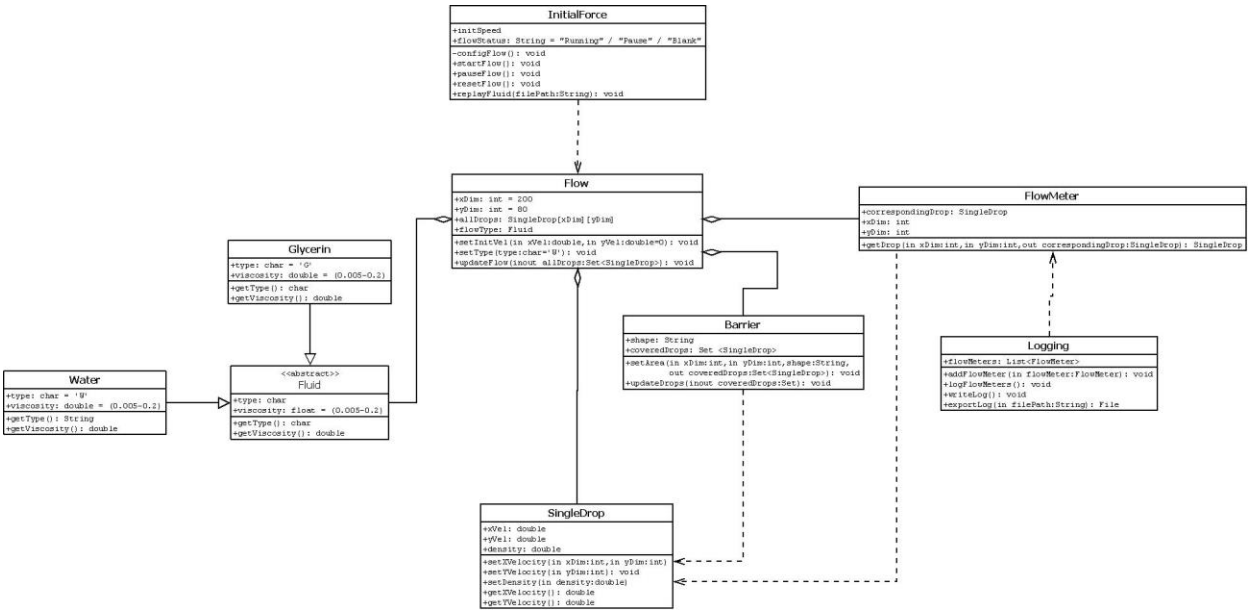
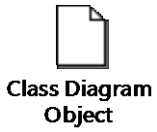


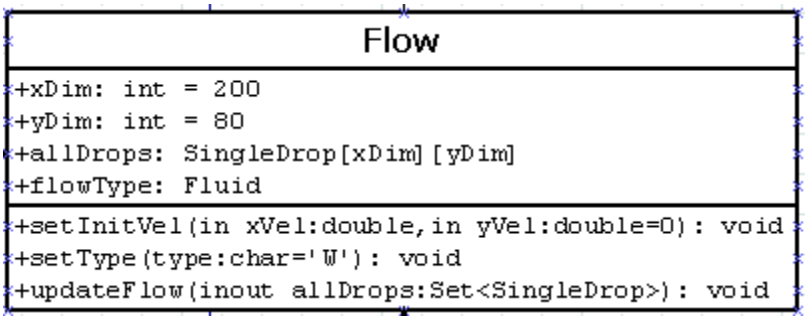

Figure – 5 Class Diagram of project LIQUID

If above image is not clear please refer the attached image:



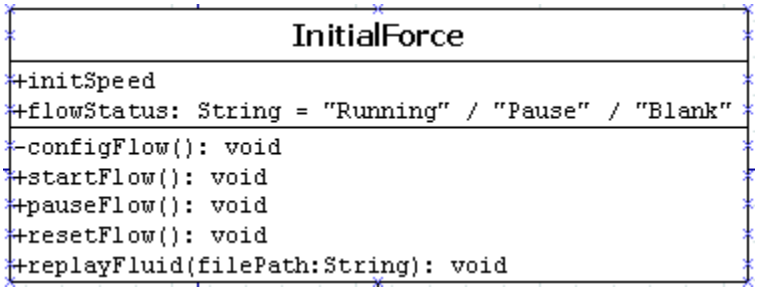

**Flow**

Flow contains all the liquid drops in the canvas, it is used to set type of liquid and initialize velocity of liquid drops at the beginning of simulation.



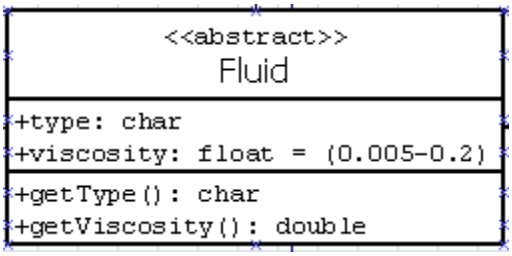

**InitialForce**

Class InitialForce is what we use to generate fluid thus it contains basic information of the fluid itself. It triggers the fluid motion and provide interfaces for pausing, resetting, replaying fluid simulation



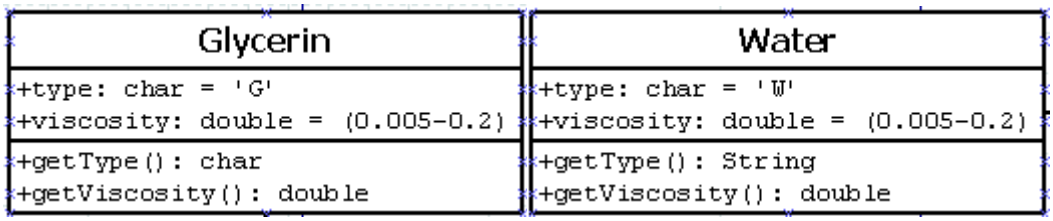

**Fluid**

An abstract class used to define all the essential attributes and methods of a certain type of fluid. It is used to be extended by its subclasses.



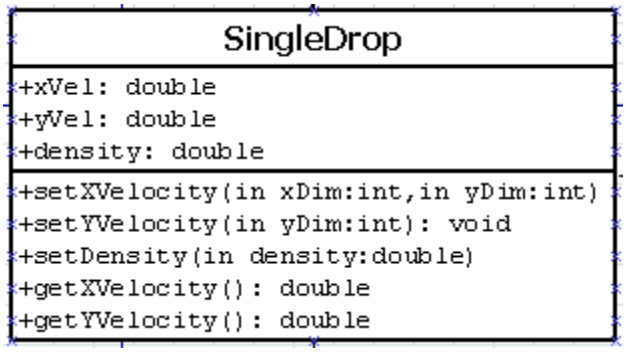

**Glycerin / Water**

Actual types of Fluid with viscosity set by default



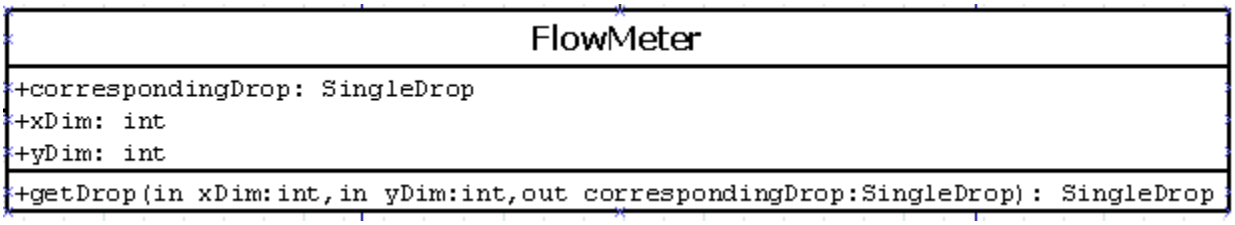
**SingleDrop**

Class SingleDrop contains information of every points defined in the rectangle. For each SingleDrop instances, it should be initialized at the beginning and can be changed as per the fluid simulation



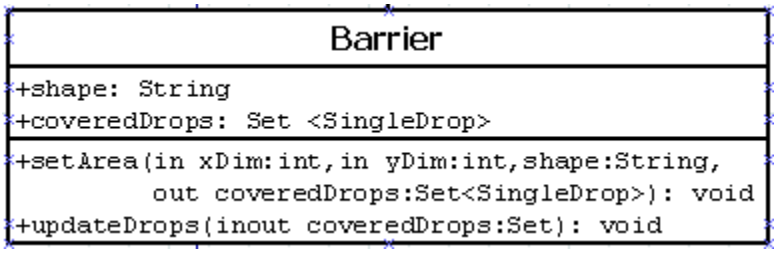

**FlowMeter**

Class FlowMeter is instantiated whenever user place a flow meter so that the values at the flow meter can be monitored. It corresponds to a SingleDrop object and get the dynamic information from it.



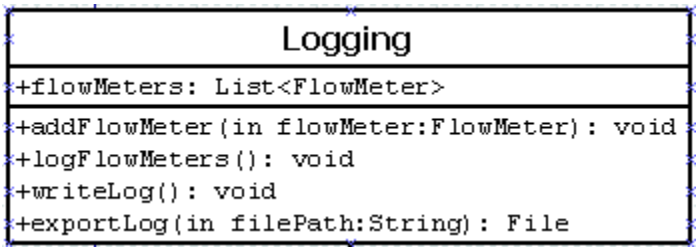

**Barrier**

Class Barrier is instantiated whenever user place a barrier with different shapes to observe the behavior of the fluid under force. It also include the SingleDrop objects underlying and will update the same accordingly if barrier is there.




**Logging**

Class Logging can be used to export log by user and write log to file by the system. Logging will also record dynamic information of flow meters set by user every 1 second




### 5.3 Sequence Diagram

Figure – 6 below shows the sequence diagram of the project LIQUID: 2-D Fluid Dynamic Simulator. The figure is self-explanatory and can be used to understand the sequence of execution.

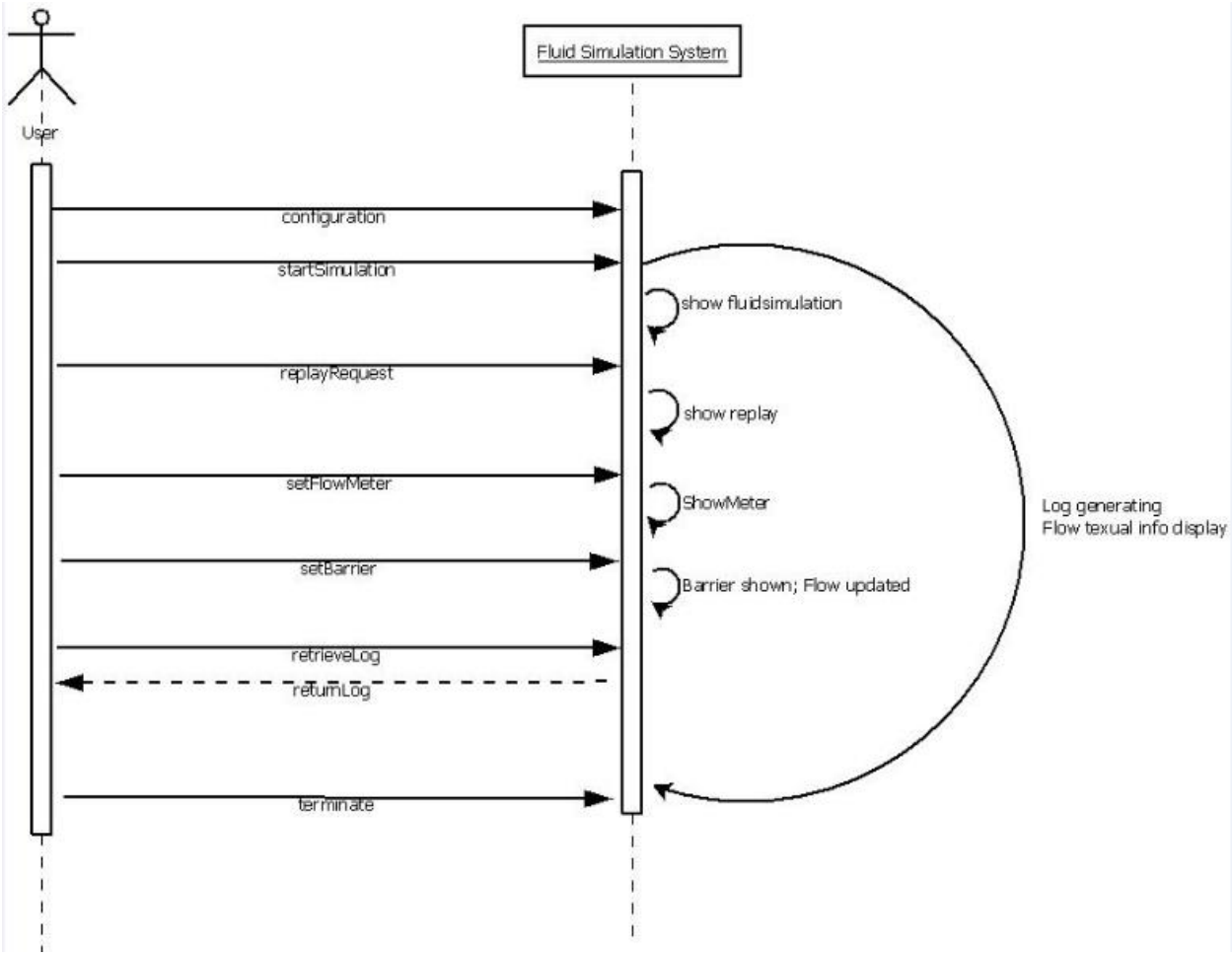



Figure – 6 Sequence diagram showing the sequence of execution in project LIQUID

### 5.4 Communication Diagram

A Communication diagram models the interactions between objects or parts in terms of sequenced messages. The Communication diagram below is used to illustrate the flow and interaction between the classes and objects listed in

class diagram in section 5.2.

Communication diagram below consist of three major steps:

- User does initial configuration and press 'Run' button to start the flow simulation
- Application will process the configuration in three sub-steps as mentioned below:
  a. Configure the initial status of flow
  b. Set the barrier as per configuration
  c. Initialize flow meter accordingly
- Once executing, application will:
  a. Calculating the flow as per the selected algorithm for display use
  b. Writing flow initial info to log
  c. Flow meter information will be written into log at a regular interval

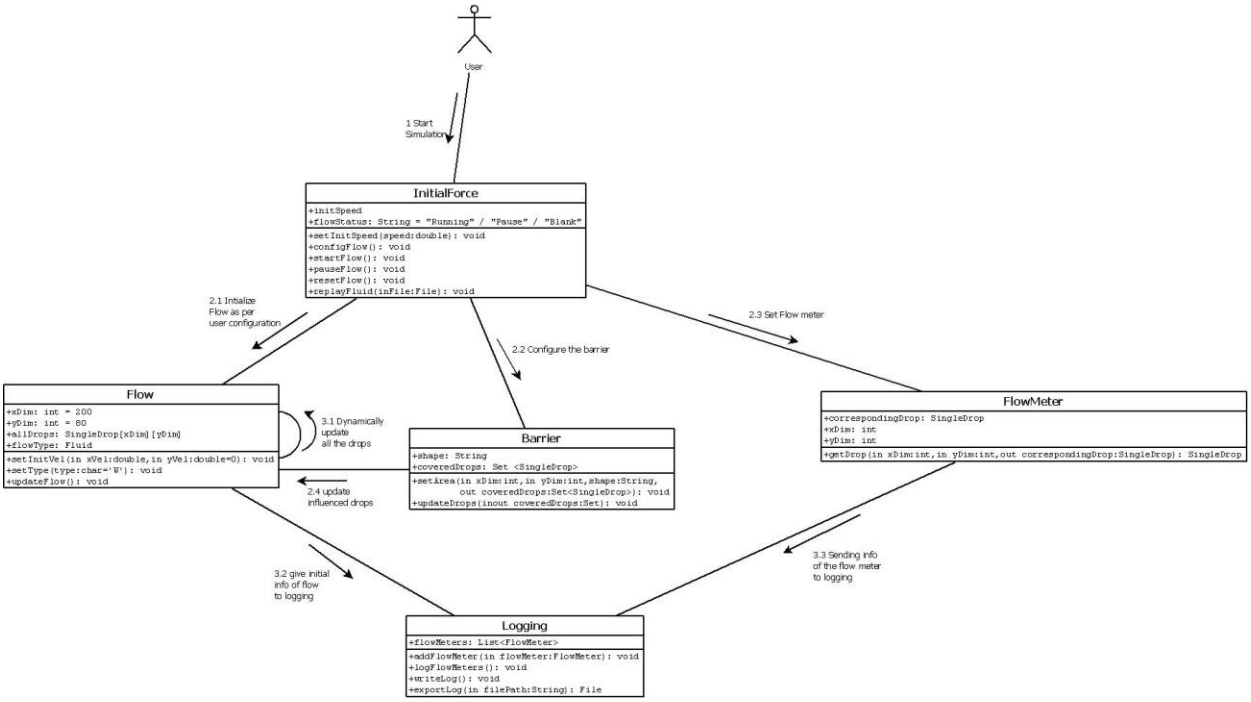



Figure – 7 Communication Diagram of the project LIQUID

If above image is not clear please refer the attached image:


# 6. Development View

The development view or implementation view illustrates the programmer's perspective of the project LIQUID: 2-D Fluid Dynamic Simulator. This section is also concerned with management of the software. Component diagram of the project is used to illustrate the components of the system and its interface.

## 6.1 Component Diagram

A component in project LIQUID: 2-D Fluid Dynamic Simulator is something required to execute a stereotype function (executable, sub-systems and library files). Interfaces are used to link them together.

- In UI part, it receives user input and call FlowConfigAPI to pass the values to process engine.
- After processing engine handled the input, it calculates the data of flow and will call DisplayFlowData API sending it back to UI layer, it also need to call LogWriteAPI for logging essential data.
- Logging file should not exceed a certain size thus it should call LogMaintananceAPI for checking the size of itself at a regular basis.

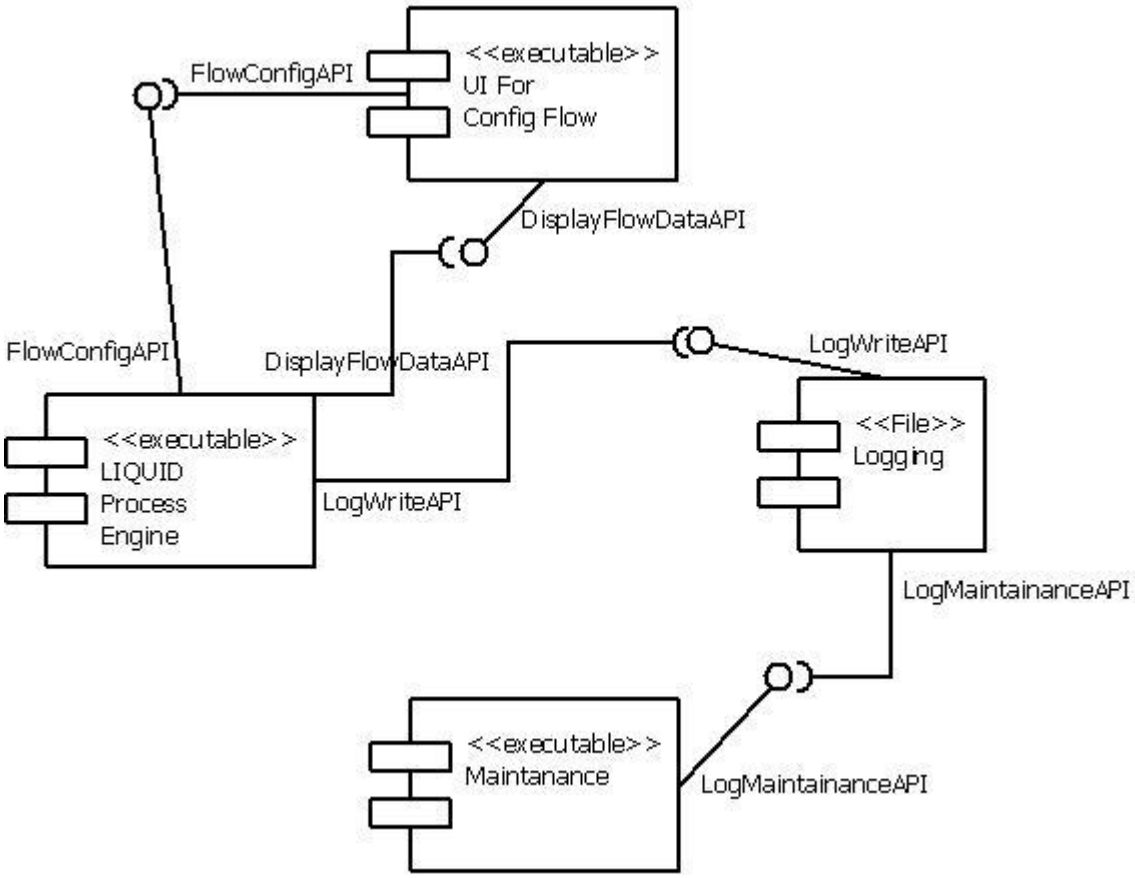



Figure – 8 Component and its interface diagram of project LIQUID

## 7. Process View

The process view section covers the dynamic aspects of the system. It covers the system processes and how they communicate, and focuses on the runtime behavior of the project LIQUID: 2-D Fluid Dynamic Simulator. This can be done by showing the activity diagram which mentions the chain of activities and the respective system functioning.

### 7.1 Activity Diagram

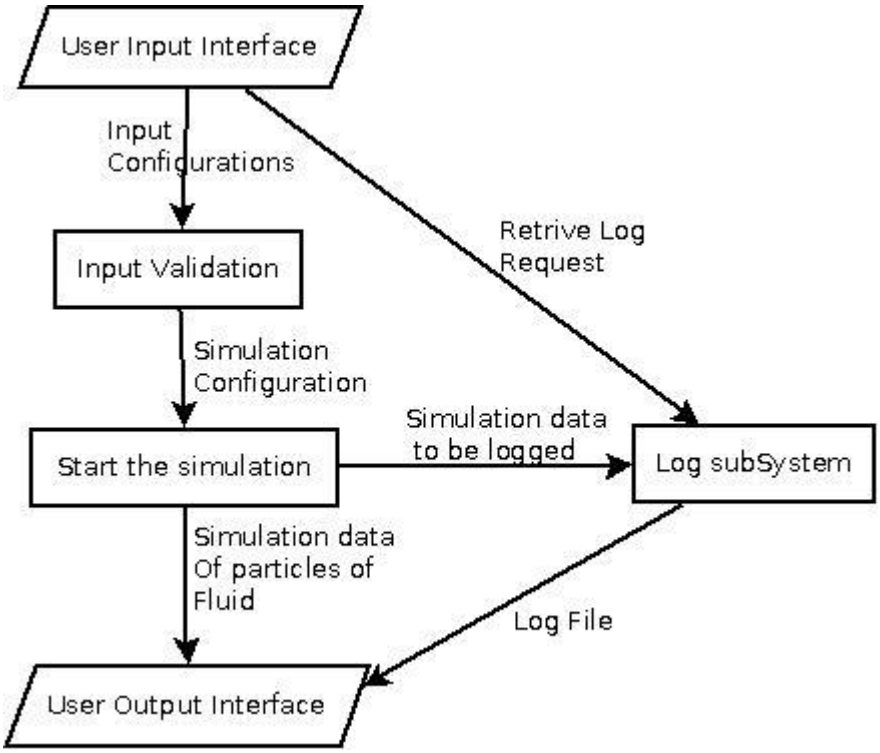



Figure – 9 Activity diagram for the project LIQUID

## 8. Physical View

This view is not applicable to this project.

## 9. Quality

As far as the project LIQUID: 2-D Fluid Dynamic Simulator is concerned, the following quality goals have been identified:

**Performance**:

- **Description:** The Frame rate for the fluid simulation should be at least 50fps and data should be logged every second.
- **Solution:** Next position of each particle in the rectangular canvas should be calculated at least every 20 milliseconds in order to achieve 50fps and data related to flow meters and break points should be logged in log file every second.

**Reliability**:

- **Description**: Targeted reliability is MTBF of 100 minutes for the software .Considering an average simulation execution time of 5 minutes, application shall work for 20 cycles without crashing.
- **Solution:** : Check has to be implemented in software to avoid crash of the application by restricting size of system generated log file and the log file used for replay.

**Portability**:

- **Description** : The application is installable for MS-Windows environment
- **Solution:** Third party library (install4j) can be used to make the software installable on MS-Windows environment so that it is portable to all systems with on MS-Windows environment.

**Security**:

Not applicable to this project