High Level Design & Low Level Design

The purpose of this document is to provide with a template for documenting both HLD & LLD.

**Document Control :**

|  |
| --- |
| **Project Revision History** |

[1. Introduction 3](#_Toc368912248)

[1.1. Intended Audience 3](#_Toc368912249)

[1.2. Acronyms/Abbreviations 3](#_Toc368912250)

[1.3. Project Purpose 3](#_Toc368912251)

[1.4. Key Project Objectives 3](#_Toc368912252)

[1.5. Project Scope and Limitation 3](#_Toc368912253)

[1.5.1. In Scope 3](#_Toc368912254)

[1.5.2. Out of scope 3](#_Toc368912255)

[1.6. Functional Overview 3](#_Toc368912256)

[1.7. Assumptions, Dependencies & Constraints 3](#_Toc368912257)

[1.8. Risks 3](#_Toc368912258)

[2. Design Overview 3](#_Toc368912259)

[2.1. Design Objectives 3](#_Toc368912260)

[2.1.1. Recommended Architecture 3](#_Toc368912261)

[2.2. Architectural Strategies 3](#_Toc368912262)

[2.2.1. Design Alternative 3](#_Toc368912263)

[2.2.2. Reuse of Existing Common Services/Utilities 3](#_Toc368912264)

[2.2.3. Creation of New Common Services/Utilities 3](#_Toc368912265)

[2.2.4. User Interface Paradigms 3](#_Toc368912266)

[2.2.5. System Interface Paradigms 3](#_Toc368912267)

[2.2.6. Error Detection / Exceptional Handling 3](#_Toc368912268)

[2.2.7. Memory Management 3](#_Toc368912269)

[2.2.8. Performance 3](#_Toc368912270)

[2.2.9. Security 3](#_Toc368912271)

[2.2.10. Concurrency and Synchronization 3](#_Toc368912272)

[2.2.11. Housekeeping and Maintenance 3](#_Toc368912273)

[3. System Architecture 3](#_Toc368912274)

[3.1. System Architecture Diagram. (Not Necessary) 3](#_Toc368912275)

[3.2. System Use-Cases 3](#_Toc368912276)

[3.3. Subsystem Architecture 3](#_Toc368912277)

[3.4. System Interfaces 3](#_Toc368912278)

[3.4.1. Internal Interfaces 3](#_Toc368912279)

[3.4.2. External Interfaces 3](#_Toc368912280)

[4. Detailed System Design 3](#_Toc368912281)

[4.1. Key Entities 3](#_Toc368912282)

[4.2. Detailed-Level Database Design 3](#_Toc368912283)

[4.2.1. Data Mapping Information 3](#_Toc368912284)

[4.2.2. Data Conversion 3](#_Toc368912285)

[4.3. Archival and retention requirements 3](#_Toc368912286)

[4.4. Disaster and Failure Recovery 3](#_Toc368912287)

[4.5. Business Process workflow 3](#_Toc368912288)

[4.6. Business Process Modeling and Management (as applicable) 3](#_Toc368912289)

[4.7. Business Logic 3](#_Toc368912290)

[4.8. Variables 3](#_Toc368912291)

[4.9. Activity / Class Diagrams (as applicable) 3](#_Toc368912292)

[4.10. Data Migration 3](#_Toc368912293)

[4.10.1. Architectural Representation 3](#_Toc368912294)

[4.10.2. Architectural Goals and Constraints 3](#_Toc368912295)

[4.10.3. Logical View 3](#_Toc368912296)

[4.10.4. Architecturally Significant Design Packages 3](#_Toc368912297)

[4.10.5. Data model 3](#_Toc368912298)

[4.10.6. Deployment View 3](#_Toc368912299)

[5. Environment Description 3](#_Toc368912300)

[5.1. Time Zone Support 3](#_Toc368912301)

[5.2. Language Support 3](#_Toc368912302)

[5.3. User Desktop Requirements 3](#_Toc368912303)

[5.4. Server-Side Requirements 3](#_Toc368912304)

[5.4.1. Deployment Considerations 3](#_Toc368912305)

[5.4.2. Application Server Disk Space 3](#_Toc368912306)

[5.4.3. Database Server Disk Space 3](#_Toc368912307)

[5.4.4. Integration Requirements 3](#_Toc368912308)

[5.4.5. Jobs 3](#_Toc368912309)

[5.4.6. Network 3](#_Toc368912310)

[5.4.7. Others 3](#_Toc368912311)

[5.5. Configuration 3](#_Toc368912312)

[5.5.1. Operating System 3](#_Toc368912313)

[5.5.2. Database 3](#_Toc368912314)

[5.5.3. Network 3](#_Toc368912315)

[5.5.4. Desktop 3](#_Toc368912316)

[6. References 3](#_Toc368912317)

[7. Appendix 3](#_Toc368912318)

# 

# Introduction

## Acronyms/Abbreviations

|  |  |
| --- | --- |
| UT | Unit Test |
| IT | Integrated Test |
|  |  |
|  |  |

## Project Purpose

Machine job scheduling application is to be developed which will process jobs data and assign requested machines to the jobs. Schedule for each machine is generated. Jobs are of different types. Most of the information on jobs is common; some of the information is specific to each job type.

.

## Key Project Objectives

To schedule without interference between jobs by maintaining time management.

## Project Scope and Limitation

Scheduling without interference and making all jobs done for given machines without leaving any job given in list of job file. Limitations will be some interference if two jobs at a time need same machine it need to be wait state.

### In Scope

It is expected to give a correct schedule for each machine, for each job given and accuracy of code under different circumstances occur while assigning different jobs.

### Out of scope

Required to involve techniques such as concurrency, inheritance to make the project done completely.

## Functional Overview

CPP ATL enables to code the job specifications ,FileIO operations helps to read file from Command line arguments and write schedule into different text files .Valgrind captures the data of memory leak.

## Assumptions, Dependencies & Constraints

Schedule for different machines is done by assuming the job given in the job file is of correct format. Modified application will remove the jobs that are not in correct format.

## Risks

All assumptions, functional overview and design parameters are documented without evaluation which are to be implemented without missing.

# Design Overview

## Design Objectives

Design objectives include different parameters like jobs given to controller and scheduling corresponding machine and providing interference less design while assigning machines to jobs.

### Recommended Architecture

UML architecture.

## Architectural Strategies

No architectural strategies have been employed.

### Design Alternative

Designed sequence diagram and use case diagram as design paradigm but as an alternative selected class diagram to visualize more data that've used in application.

### Reuse of Existing Common Services/Utilities

Design and development is done from scratch using existing sources star UML for design and VI for development.

### Creation of New Common Services/Utilities

Used existing resources to develop the application specific services.

### Error Detection / Exceptional Handling

The files that are sent through command line argument are checked for exception and catches exception if it occurs and at the conditions where the source code may terminate gets checked for exceptions by placing that code in try block if any exceptions it cached by catch block.

### Memory Management

This application involves Dynamic memory allocation which allows us to use only required memory without reserving extra memory. Valgrind application used to check for memory leaks if any and rectified if any memory leaks occur.

### Performance

System performance needs to be accurate while scheduling for each machine. Each machine schedule is to be done in different files as given without interference.

### Security

The source code is available only in read only mode for others who are accessing file and final executable file is only available for further modifications.

### Concurrency and Synchronization

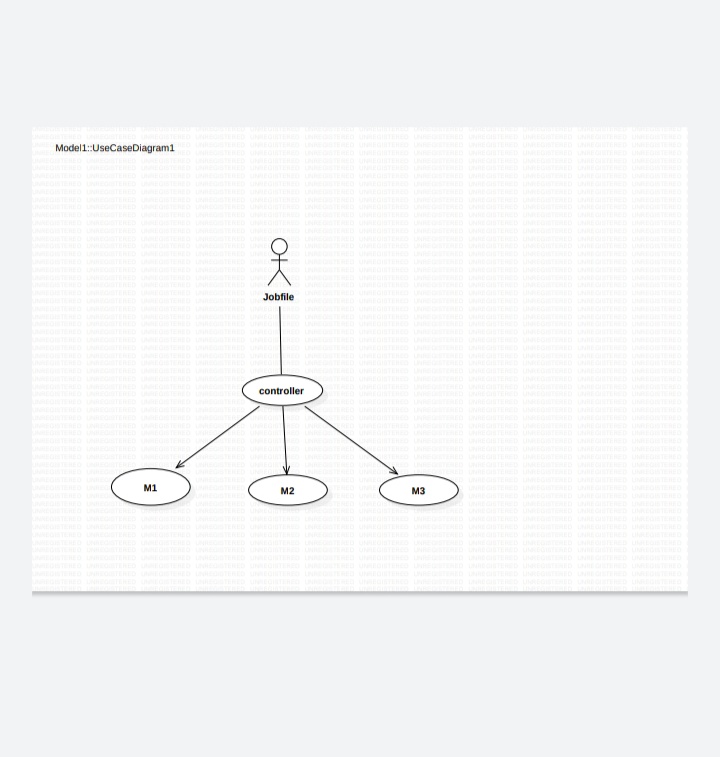
Application is designed to work concurrently without interfering with the function of other part of code and also work simultaneously.

### Housekeeping and Maintenance

Makefile is cleared first to remove the previous cache and text files are used to manage the database.

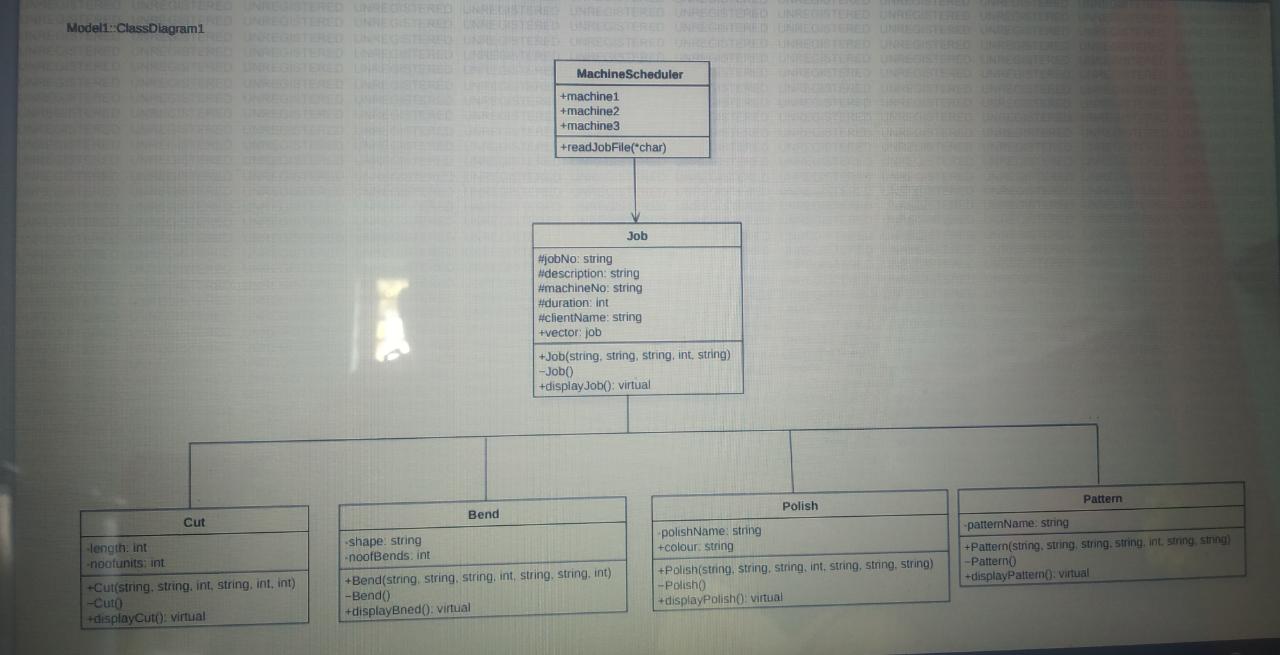
# System Architecture

## System Use-Cases

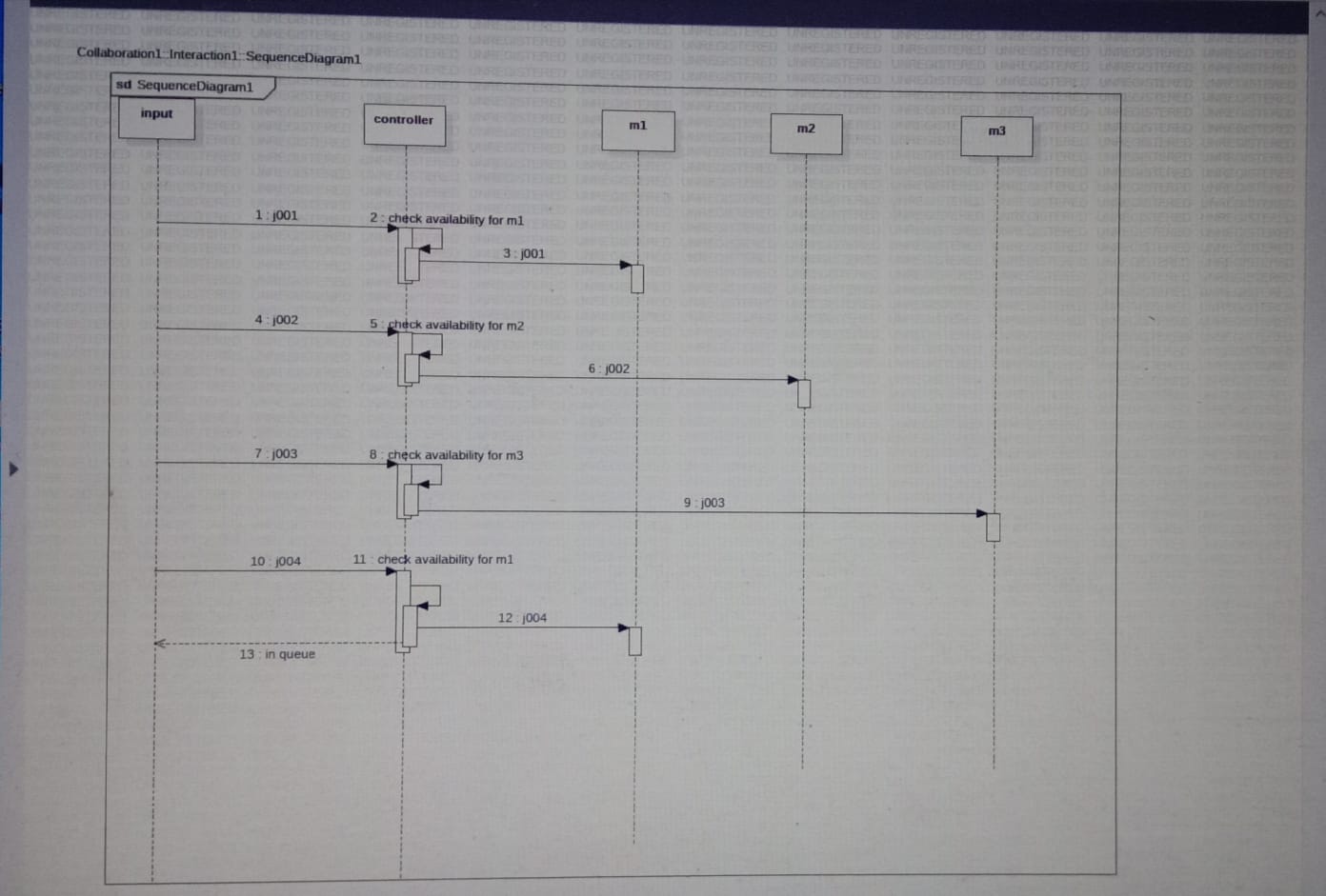


We use use-case diagram to indicate the actors as job members and use case as machines.

## Subsystem Architecture



### Internal Interfaces



# Detailed System Design

## Key Entities

We use key entities as valgrind and gdb.

## Business Process workflow

Business workflow is a repeatable process that consists of a series of tasks that generally need to be completed in a specific sequence.

## Variables

Job No, Description, machine Number, duration, client name

## Activity / Class Diagrams (as applicable)

We have taken job as parent class and child classes as cut, bend, polish, pattern.

## Language Support

CPP on Linux, data structures, object-oriented programming.

## User Desktop Requirements

We use Putty and winscp as desktop Requirements.

## Configuration

Operating system, Processor.

4GB RAM, Processor Intel (R) Core (TM) I3-7020U CPU @, 64 bit operating system, x 64-based processor.

### Database

Operating system, processor, disk space, memory.

### Network

Network is a process of assigning network settings, policies, flows and controls. In a virtual network, its easier to make network configuration changes because physical network devices appliances are replaced by software removing the need for extensive manual configuration.

### Desktop

Linux OS,