Department of Computer Science and Engineering  
The University of Texas at Arlington

Detail Design Specification

Team: Ink3D

Project: 3-D Printer Fabrication System

Team Members:   
*Daniel Lain*

*Timothy Edmonson*

*Shawn Simonson*

*Jesse Bowles*

# Table of Contents

[Table of Contents 2](#_Toc380533687)

[1 Introduction 10](#_Toc380533688)

[1.1 Overview 10](#_Toc380533689)

[1.2 Project Concept 10](#_Toc380533690)

[1.3 Project Scope 10](#_Toc380533691)

[2 Architecture Overview 11](#_Toc380533692)

[2.1 Module Decomposition Tree 13](#_Toc380533693)

[2.2 Module Decomposition Diagram 14](#_Toc380533694)

[2.3 Module Data Flows Table 15](#_Toc380533695)

[2.4 Producer Consumer Table 16](#_Toc380533696)

[2.5 Module Functional Descriptions 17](#_Toc380533697)

[2.5.1 Use Interface Layer Modules 17](#_Toc380533698)

[2.5.2 Preprocessing Layer Modules 17](#_Toc380533699)

[2.5.3 Processing Layer Modules 18](#_Toc380533700)

[2.5.4 Post Processing Layer Modules 18](#_Toc380533701)

[2.5.5 Printer Control Layer Modules 18](#_Toc380533702)

[2.5.6 Printer Feedback Layer Modules 18](#_Toc380533703)

[2.5.7 Communications Layer Modules 18](#_Toc380533704)

[3 Data Descriptions 19](#_Toc380533705)

[3.1 Print Job Configuration Aggregation Hierarchy 19](#_Toc380533706)

[3.1.1 PrintJobConfiguration Class 20](#_Toc380533707)

[3.1.2 PrinterConfiguration Class 21](#_Toc380533708)

[3.1.3 PrinterStatus Class 23](#_Toc380533709)

[3.1.4 SubsectionConfiguration Class 24](#_Toc380533710)

[3.1.5 PrintConfiguration Class 25](#_Toc380533711)

[3.1.6 InfillConfiguration Class 26](#_Toc380533712)

[3.1.7 LayerAndPerimeterConfiguration Class 28](#_Toc380533713)

[3.1.8 SpeedConfiguration Class 29](#_Toc380533714)

[3.1.9 SkirtAndBrimConfiguration Class 30](#_Toc380533715)

[3.1.10 SupportMaterialConfiguration Class 31](#_Toc380533716)

[3.1.11 FileConfiguration Class 32](#_Toc380533717)

[3.1.12 MaterialConfiguration Class 33](#_Toc380533718)

[3.1.13 ExtruderConfiguration Class 35](#_Toc380533719)

[4 User Interface Layer 36](#_Toc380533720)

[4.1 GUI Subsystem Modules 36](#_Toc380533721)

[4.1.1 Import GUI Module 36](#_Toc380533722)

[4.1.2 Material Configuration GUI Module 37](#_Toc380533723)

[4.1.3 Printer Configuration GUI Module 38](#_Toc380533724)

[4.1.4 Extruder Configuration GUI Module 39](#_Toc380533725)

[4.1.5 Print Configuration GUI Module 40](#_Toc380533726)

[4.1.6 Print Job GUI Module 41](#_Toc380533727)

[4.1.7 Status GUI Module 42](#_Toc380533728)

[4.2 Controller Subsystem Modules 43](#_Toc380533729)

[4.2.1 Import Controller 43](#_Toc380533730)

[4.2.2 Material Configuration Controller 44](#_Toc380533731)

[4.2.3 Printer Configuration Controller 45](#_Toc380533732)

[4.2.4 Extruder Configuration Controller 46](#_Toc380533733)

[4.2.5 Print Configuration Controller 47](#_Toc380533734)

[4.2.6 Print Job Controller 48](#_Toc380533735)

[4.2.7 Status Controller 50](#_Toc380533736)

[4.3 Database Subsystem Modules 52](#_Toc380533737)

[4.3.1 Persistence Framework 52](#_Toc380533738)

[4.3.2 Command Structure 54](#_Toc380533739)

[5 Preprocessing Layer 57](#_Toc380533740)

[5.1 Normalization Subsystem Modules 58](#_Toc380533741)

[5.1.1 Object Subsection Module 58](#_Toc380533742)

[5.1.2 File Translation Module 59](#_Toc380533743)

[6 Processing Layer 61](#_Toc380533744)

[6.1 Slicing Subsystem Modules 62](#_Toc380533745)

[6.1.1 Slicing Engine Wrapper 62](#_Toc380533746)

[7 Post processing Layer 64](#_Toc380533747)

[7.1 G-Code Preparation Subsystem Modules 65](#_Toc380533748)

[7.1.1 Parser Module 65](#_Toc380533749)

[7.1.2 Unification Module 66](#_Toc380533750)

[8 Printer control layer 68](#_Toc380533751)

[8.1 Printer State Controller Subsystem Modules 68](#_Toc380533752)

[8.1.1 Printer State Control Module 68](#_Toc380533753)

[9 Printer Feedback Layer 71](#_Toc380533754)

[9.1 State Monitoring Subsystem Modules 71](#_Toc380533755)

[9.1.1 Dispatch Module 71](#_Toc380533756)

[10 Communications Layer 73](#_Toc380533757)

[10.1 Communications Subsystem Modules 73](#_Toc380533758)

[10.1.1 RX/TX Module 73](#_Toc380533759)

[10.1.2 Serialization Module 74](#_Toc380533760)

[10.1.3 De-Serialization Module 75](#_Toc380533761)

[11 Quality Assurance 77](#_Toc380533762)

[11.1 Unit Testing 77](#_Toc380533763)

[11.1.1 User Interface Layer 77](#_Toc380533764)

[11.1.2 Preprocessing Layer 79](#_Toc380533765)

[11.1.3 Processing Layer 79](#_Toc380533766)

[11.1.4 Post Processing Layer 79](#_Toc380533767)

[11.1.5 Printer Control Layer 79](#_Toc380533768)

[11.1.6 Printer Feedback Layer 79](#_Toc380533769)

[11.1.7 Communication Layer 80](#_Toc380533770)

[11.2 Component Testing 80](#_Toc380533771)

[11.2.1 User Interface Layer 80](#_Toc380533772)

[11.2.2 Preprocessing Layer 80](#_Toc380533773)

[11.2.3 Processing Layer 80](#_Toc380533774)

[11.2.4 Post Processing Layer 80](#_Toc380533775)

[11.2.5 Printer Control Layer 81](#_Toc380533776)

[11.2.6 Printer Feedback Layer 81](#_Toc380533777)

[11.2.7 Communications Layer 81](#_Toc380533778)

[11.3 Integration Testing 81](#_Toc380533779)

[11.4 System Verification Testing 81](#_Toc380533780)

[11.4.1 Verify that the system reads STL files 81](#_Toc380533781)

[11.4.2 Verify the database interface 82](#_Toc380533782)

[11.4.3 Verify the system prints a model 82](#_Toc380533783)

[11.4.4 Verify the system stops printing of out of operational range 82](#_Toc380533784)

[12 Requirements Traceability Matrix 83](#_Toc380533785)

[12.1 User Interface Layer 83](#_Toc380533786)

[12.2 Processing Layers 84](#_Toc380533787)

[12.3 Printer State, Communications, and Printer Feedback Layers 85](#_Toc380533788)

[13 Acceptance Plan 86](#_Toc380533789)

[13.1 Package and Installation 86](#_Toc380533790)

[13.2 Acceptance Testing 86](#_Toc380533791)

[13.3 Acceptance Criteria 86](#_Toc380533792)

[14 Appendices 87](#_Toc380533793)

[14.1 Slic3r 87](#_Toc380533794)

[14.2 OpenSCAD 87](#_Toc380533795)

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Revision Date | Description | Rationale |
| 1.0 | 2/20/14 | Gate Review Version | Final changes prior to gate review |
| 2.0 | 2/26/14 | Baseline | Incorporate Gate review changes. |
|  |  |  |  |
|  |  |  |  |

List of Figures

Figure # Title Page #

[Figure 1‑1: System Concept 11](#_Toc380533796)

[Figure 2‑1: Architecture Diagram 12](#_Toc380533797)

[Figure 2‑2: Decomposition Tree 13](#_Toc380533798)

[Figure 2‑3: Module Design Diagram 14](#_Toc380533799)

[Figure 3‑1:Print Job Configuration Class Structure 19](#_Toc380533800)

[Figure 3‑2 : Print Job Configuration UML 20](#_Toc380533801)

[Figure 3‑3 : Printer Configuration UML 21](#_Toc380533802)

[Figure 3‑4 : Printer Status UML 23](#_Toc380533803)

[Figure 3‑5 : Subsection Configuration UML 24](#_Toc380533804)

[Figure 3‑6 : Print Configuration UML 25](#_Toc380533805)

[Figure 3‑7 : Infill Configuration UML 26](#_Toc380533806)

[Figure 3‑8 : Layer and Perimeter Configuration UML 28](#_Toc380533807)

[Figure 3‑9 : Speed Configuration UML 29](#_Toc380533808)

[Figure 3‑10 : Skirt and Brim Configuration UML 30](#_Toc380533809)

[Figure 3‑11 : Support Material Configuration UML 31](#_Toc380533810)

[Figure 3‑12 : File Configuration UML 32](#_Toc380533811)

[Figure 3‑13 : Material Configuration UML 33](#_Toc380533812)

[Figure 3‑14 : Extruder Configuration UML 35](#_Toc380533813)

[Figure 4‑1: Database Subsystem UML 52](#_Toc380533814)

[Figure 5‑1: Preprocessing Layer UML 57](#_Toc380533815)

[Figure 5‑2 : Normalization Subsystem UML 58](#_Toc380533816)

[Figure 6‑1 : Processing Layer UML 61](#_Toc380533817)

[Figure 6‑2 : Slicing Subsystem UML 62](#_Toc380533818)

[Figure 7‑1: Post Processing Layer UML 64](#_Toc380533819)

[Figure 7‑2 : G-Code Preparation Subsystem UML 65](#_Toc380533820)

List of Tables

Table # Title Page #

[Table 2‑1: Module Data Flows 15](#_Toc380533821)

[Table 2‑2: Producer Consumer Table 16](#_Toc380533822)

[Table 3‑1 : Print Job Configuration Data Elements 20](#_Toc380533823)

[Table 3‑2 : Printer Configuration Data Elements 22](#_Toc380533824)

[Table 3‑3 : Printer Status Data Elements 23](#_Toc380533825)

[Table 3‑4: Subsection Configuration Data Elements 24](#_Toc380533826)

[Table 3‑5 : Print Configuration Data Elements 25](#_Toc380533827)

[Table 3‑6 : Infill Configuration Data Elements 27](#_Toc380533828)

[Table 3‑7 : Layer and Perimeter Configuration Data Elements 28](#_Toc380533829)

[Table 3‑8 : Speed Configuration Data Elements 29](#_Toc380533830)

[Table 3‑9 : Skirt and Brim Configuration Data Elements 30](#_Toc380533831)

[Table 3‑10 : Support Material Configuration Data Elements 31](#_Toc380533832)

[Table 3‑11 : File Configuration Data Elements 32](#_Toc380533833)

[Table 3‑12 : Material Configuration Data Elements 35](#_Toc380533834)

[Table 3‑13 : Extruder Configuration Data Elements 35](#_Toc380533835)

[Table 4‑1 : Import GUI Module Interfaces 36](#_Toc380533836)

[Table 4‑2 : Import GUI Module External Data Dependencies 36](#_Toc380533837)

[Table 4‑3 : Material Configuration GUI Module Interfaces 37](#_Toc380533838)

[Table 4‑4 : Material Configuration GUI Module External Data Dependencies 37](#_Toc380533839)

[Table 4‑5 : Printer Configuration GUI Module Interfaces 38](#_Toc380533840)

[Table 4‑6 : Printer Configuration GUI Module External Data Dependencies 38](#_Toc380533841)

[Table 4‑7 : Extruder Configuration GUI Module Interfaces 39](#_Toc380533842)

[Table 4‑8 : Extruder Configuration GUI Module External Data Dependencies 40](#_Toc380533843)

[Table 4‑9 : Print Configuration GUI Module Interfaces 40](#_Toc380533844)

[Table 4‑10 : Print Configuration GUI Module External Data Dependencies 40](#_Toc380533845)

[Table 4‑11 : Print Job GUI Module Interfaces 41](#_Toc380533846)

[Table 4‑12 : Print Job GUI Module External Data Dependencies 41](#_Toc380533847)

[Table 4‑13 : Status GUI Module Interfaces 42](#_Toc380533848)

[Table 4‑14 : Status GUI Module External Data Dependencies 42](#_Toc380533849)

[Table 4‑15 : Import Controller Module Interfaces 43](#_Toc380533850)

[Table 4‑16 : Import Controller Module External Data Dependencies 43](#_Toc380533851)

[Table 4‑17 : Import Controller Module Internal Data Dependencies 43](#_Toc380533852)

[Table 4‑18 : Material Controller Module Interfaces 44](#_Toc380533853)

[Table 4‑19 : Material Controller Module Internal Data Descriptors 44](#_Toc380533854)

[Table 4‑20 : Printer Controller Module Interfaces 45](#_Toc380533855)

[Table 4‑21 : Printer Controller Module External Data Dependencies 45](#_Toc380533856)

[Table 4‑22 : Printer Controller Module Internal Data Descriptors 45](#_Toc380533857)

[Table 4‑23 : Extruder Configuration Controller Interfaces 46](#_Toc380533858)

[Table 4‑24 : Extruder Configuration Controller Internal Data Descriptors 46](#_Toc380533859)

[Table 4‑25 : Print Configuration Controller Interfaces 47](#_Toc380533860)

[Table 4‑26 : Print Configuration Controller Internal Data Descriptors 48](#_Toc380533861)

[Table 4‑27 : Print Job Controller Interfaces 49](#_Toc380533862)

[Table 4‑28 : Print Job Controller Internal Data Descriptors 49](#_Toc380533863)

[Table 4‑29 : Status Controller Interfaces 50](#_Toc380533864)

[Table 4‑30 : Status Controller Internal Data Desctiptors 50](#_Toc380533865)

[Table 4‑31 : Persistence Framework Interfaces 53](#_Toc380533866)

[Table 4‑32 : Persistence Framework Internal Data Descriptors 54](#_Toc380533867)

[Table 4‑33 : Command Structure Interfaces 55](#_Toc380533868)

[Table 4‑34 : Command Structure External Data Dependencies 55](#_Toc380533869)

[Table 4‑35 : Command Structure Internal Data Descriptors 56](#_Toc380533870)

[Table 5‑1 : Object Subsection Module Interfaces 58](#_Toc380533871)

[Table 5‑2 : Object Subsection Module External Data Dependencies 58](#_Toc380533872)

[Table 5‑3 : Object Subsection Module Internal Data Descriptors 59](#_Toc380533873)

[Table 5‑4 : File Translation Module Interfaces 59](#_Toc380533874)

[Table 5‑5 : File Translation Module Internal Data Descriptors 59](#_Toc380533875)

[Table 6‑1 : Slicing Engine Wrapper Interfaces 62](#_Toc380533876)

[Table 6‑2 : Slicing Engine Wrapper External Data Descriptors 62](#_Toc380533877)

[Table 6‑3 : Slicing Engine Wrapper Internal Data Descriptors 63](#_Toc380533878)

[Table 7‑1 : Parser Module Interfaces 65](#_Toc380533879)

[Table 7‑2 : Parser Module Internal Data Descriptors 66](#_Toc380533880)

[Table 7‑3 : Unification Module Interfaces 66](#_Toc380533881)

[Table 7‑4 : Unification Module Internal Data Descriptors 67](#_Toc380533882)

[Table 8‑1 : Printer State Control Module Interfaces 68](#_Toc380533883)

[Table 8‑2 : Printer State Control Module Internal Data Descriptors 68](#_Toc380533884)

[Table 9‑1 : Dispatch Module Interfaces 71](#_Toc380533885)

[Table 9‑2 : Dispatch Module Internal Data Descriptors 71](#_Toc380533886)

[Table 10‑1 : Rx/Tx Module Interfaces 73](#_Toc380533887)

[Table 10‑2 : Rx/Tx External Module Data Dependencies 73](#_Toc380533888)

[Table 10‑3 : Rx/Tx Module Internal Data Descriptors 74](#_Toc380533889)

[Table 10‑4 : Serialization Module Interfaces 75](#_Toc380533890)

[Table 10‑5 : Serialization Module Internal Data Descriptors 75](#_Toc380533891)

[Table 10‑6 : De-Serialization Module Interfaces 75](#_Toc380533892)

[Table 10‑7 : De-Serialization Module Internal Data Descriptors 76](#_Toc380533893)

[Table 11‑1 : User Interface Layer Unit Tests 78](#_Toc380533894)

[Table 11‑2 : Preprocessing Layer Unit Tests 79](#_Toc380533895)

[Table 11‑3 : Processing L ayer Unit Tests 79](#_Toc380533896)

[Table 11‑4 : Post Processing Layer Unit Tests 79](#_Toc380533897)

[Table 11‑5Printer Control Layer Unit Tests 79](#_Toc380533898)

[Table 11‑6 : Printer Feedback Layer Unit Tests 79](#_Toc380533899)

[Table 11‑7 : Communications Layer Unit Tests 80](#_Toc380533900)

[Table 12‑1 : User Interface Layer Requirement Tracibility Matrix 83](#_Toc380533901)

[Table 12‑2 : Processing Layers Requirement Tracibility Matrix 84](#_Toc380533902)

[Table 12‑3 : Printer Stste, Communications, and Printer Feedback Layers Requirement Tracibility Matrix 85](#_Toc380533903)

# Introduction

## Overview

The Detailed Design Specification (DDS) provides a detailed specification of the subsystem and module breakdown of the 3-D Printer Fabrication System. The document is intended to use detailed constructs to represent implementation. The document focuses on low level concepts such as algorithms, module definitions, and data flows in such a way that the system can be implemented without requiring any further design work.

## Project Concept

Mainstream 3-D printers and their respective tool chains are suited for printing a 3-D model from an abstract model developed from Computer Aided Drafting (CAD) software. The software tool chain is capable of issuing instructions to the printer in the form of G-Codes. The printer then prints the object layer by layer as an additive process by depositing a heated polymer or plastic onto the print bed iteratively until the final object is realized. The limitation of this style of tool chain, however, is that it is not suited for depositing multiple materials interleaved with each other on a single print run.

The 3-D Printer Fabrication System aims to remove this limitation by providing a software tool chain that will read and interpret Stereo Lithography (STL) files with varying material parameters into a single stream of G-Codes that will be interpreted and executed by a device to be implemented by a Mechanical Engineering team at the University of Texas at Arlington. This device will be capable of depositing multiple materials within a single print run.

## Project Scope

The scope of the 3-D Printer Fabrication System is to develop software that will produce suitable machine code for a 3-D printer head that is capable of depositing multiple materials within a single print run. The system will present the user with an interface that will allow them to specify which STL files are to be loaded and specify the material properties of the respective STL files. The system will then use this information to process the geometry such that a suitable set of G-Codes can be issued to the device. The system will also provide a method for streaming the information to the printer control hardware via a serial interface. The system is intended to be used by 3-D printer operators, CNC operators, Dr. Shiakolas, and other experienced operators in the research field. The system is not intended for the consumer market.



Figure 1‑1: System Concept

# Architecture Overview

The 3-D Printer Fabrication System is intended to be a software system that initially provides the core functionality necessary to print multi-material objects on a multi-extruder 3-D printer, but can be easily expanded in the future to become a feature rich multi-material 3-D printing software suite. This goal dives the need for a modular, configurable, extensible, and portable system. To meet these needs the architecture has been broken in to seven layers with the design constraint that each layer should be replaceable and modular for future expansion. The User Interface Layer is responsible to providing a GUI for the user to enter configuration date, select print options, load and save configuration, load object files, and start the print process. The Preprocessing Layer takes the configuration and object files provided by the User Interface Layer and translate the files into a correct form for the processing layer. The Processing Layer takes the configuration and translated object files and produces an instruction set in the form of G-Codes for the Post Processing Layer. The Post Processing Layer uses the instruction sets created by the Processing layer and the configuration data to build the complete series of G-Codes in the proper format for the Printer Control Layer to use. The Printer Control Layer uses the G-Codes and configuration to stream the instructions one by one to the Communications Layer. The Printer Control Layer also uses a pause, stop, resume command from the User Interface Layer based on the user input. The Printer Control Layer must also provide printer state monitoring and maintain bounds control to ensure safe and correct printer operation. The Communications Layer is responsible for serialization, deserialization, and direct serial communication with the printer. The Communications Layer uses the instructions issued by the Printer Control Layer to serialize those instructions then send them to the printer through the serial interface. The Communications Layer poles the printer for state information then de-serializes this information and sends it to the Printer Feedback Layer. The Printer Feedback Layer is responsible for sending printer state information to the Printer Control Layer and the User Interface Layer.



Figure 2‑1: Architecture Diagram

## Module Decomposition Tree

The 3-D Printer Fabrication System illustrates its modular design. The system is decomposed into seven layers. Each layer is decomposed to one or more subsystems that have related functionality. Each subsystem is then decomposed into one or more modules that support that subsystems function.



Figure 2‑2: Decomposition Tree

## Module Decomposition Diagram



Figure 2‑3: Module Design Diagram

## Module Data Flows Table

|  |  |  |
| --- | --- | --- |
| Layer | Data Flow ID | Data |
| Outside Inputs | OI1 | Object file of the STL file to be printed |
| OI2 | Printer configuration data entry values |
| OI3 | Material configuration data entry values |
| OI4 | Print configuration data entry values |
| OI5 | Print selections and button press |
| OI6 | Pause, resume, and stop button Presses |
| OI7 | Disk reads of XML and Directory structure |
| OI8 | Printer state information |
| OI9 | OS Driver information |
| OI10 | Extruder configuration data entry values |
| User Interface Layer | UI1 | Import file name |
| UI2 | Success state of import |
| UI3 | Loaded printer configuration |
| UI4 | Save printer configuration request |
| UI5 | Save material configuration request |
| UI6 | Loaded material configuration |
| UI7 | Loaded print configuration |
| UI8 | Save print configuration request |
| UI9 | Loaded configuration data |
| UI10 | Run print job |
| UI11 | Pause, resume, and stop button Presses |
| UI12 | Printer state information |
| UI13 | File Configuration object |
| UI14 | Printer configuration object |
| UI15 | Material configuration object |
| UI16 | Print Configuration object |
| UI17 | All requested configuration objects |
| UI18 | Object save/load requests |
| UI19 | Object save/load results |
| UI20 | XML File writes to disk |
| UI21 | Save print job configuration |
| UI22 | Print package object |
| UI23 | Print, Pause, Resume request |
| UI24 | Loaded Extruder Configuration Data |
| UI25 | Save Extruder Configuration Request |
| UI26 | Extruder Configuration Object |
| Preprocessing Layer | PR1 | Print package object |
| PR2 | Print package object |
| Processing Layer | PO1 | Print package object |
| Post-Processing Layer | PP1 | Print package object |
| PP2 | Print package object |
| Printer Control Layer | PL1 | G-Code |
| Communications Layer | CL1 | Serialized G-Codes |
| CL2 | Serialized printer state |
| CL3 | Command stream |
| Printer Feedback Layer | PF1 | Printer State Object |
| PF2 | Printer State Object |

Table 2‑1: Module Data Flows

## Producer Consumer Table

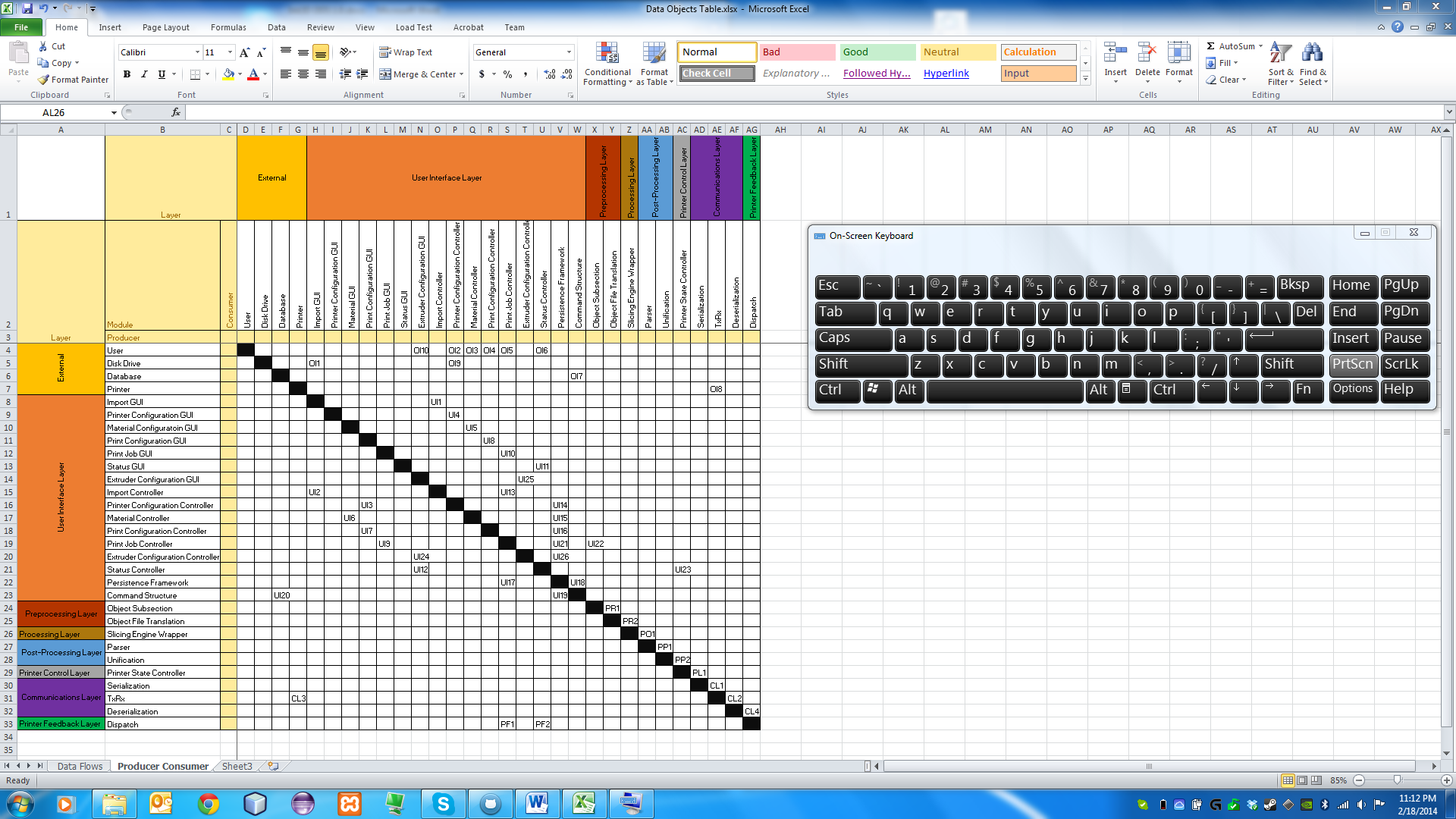


Table 2‑2: Producer Consumer Table

## Module Functional Descriptions

### Use Interface Layer Modules

* GUI Subsystem
  + Import GUI Module
    - GUI Module to allow the user to import STL files.
  + Printer Configuration GUI Module
    - GUI module to allow the user to enter printer configuration data.
  + Material GUI Module
    - GUI module to allow the user to enter material configuration data.
  + Print Configuration GUI Module
    - GUI module to allow the user to enter print configuration data.
  + Print Job GUI Module
    - GUI module to allow the user to set the print job configuration and initiate the print run.
  + Extruder Configuration GUI Module
    - GUI module to allow the user to enter extruder configuration data.
  + Status Module
    - GUI module to allow the user to view current status of the print run as well as pause, resume or stop the current print run.
* Controller Subsystem
  + Import Controller
    - Business logic for the importing o STL files.
  + Printer Configuration Controller
    - Logical controller for storing and updating printer configuration files.
  + Material Controller
    - Logical controller for storing and updating material configuration files
  + Print Configuration Controller
    - Logical controller for storing and updating print configuration files
  + Print Job Controller
    - Logical controller for creating and initiating print runs.
  + Extruder Configuration Controller
    - Logical controller for storing and updating extruder configuration files.
  + Status Controller
    - Logical controller for updating the printer status display and pause, resume, stop print run actions.
* Database Subsystem
  + Persistence Framework
    - Provides the controllers with a single point of contact for the data store.
  + Command Structure
    - Implements the commands required to talk to the data store to store and retrieve configuration objects.

### Preprocessing Layer Modules

* Normalization Module
  + Object Subsection Module
    - Creates subsection STL files.
  + Object File Translation Module
    - Combines the subsection collection into an AMF file.

### Processing Layer Modules

* Slicing Engine
  + Slicing Engine Wrapper
    - Generates G-Code from the AMF file for each subsection.

### Post Processing Layer Modules

* G-Code Preparation
  + Parser Module
    - Reads g-codes and modifies them as required for the current printer configuration
  + Unification Module
    - Combines the subsection g-code sets into one series of g-codes to be printed.

### Printer Control Layer Modules

* Printer State Controller
  + Printer State Controller
    - Issues g-code commands to the printer in sequence and monitors the operational state of the printer.

### Printer Feedback Layer Modules

* State Monitoring
  + Dispatch Module
    - Combines information about the printer state received from the communications layer and sends the state object to the Printer State Controller and the Status Controller.

### Communications Layer Modules

* Communications Subsystem
  + Serialization Module
    - Serializes the g-codes from the Printer State Controller in the correct format for the printer.
  + RxTx Module
    - Transmits and receives information directly with the printer.
  + Deserialization Module
    - De-serializes printer state information and passes the data to the Dispatch Module.

# Data Descriptions

The framework of this system relies heavily on the Print Job Configuration object. This object is an instantiation of the PrintJobConfiguration class, which encapsulates all objects and data necessary for the processing and hardware communication layers. The Print Job Configuration object is instantiated in the User Interface Layer and passed down through all other layers where it is modified by each layer. This section is intended to define, in detail, the PrintJobConfiguration class and the classes encapsulated within.

## Print Job Configuration Aggregation Hierarchy

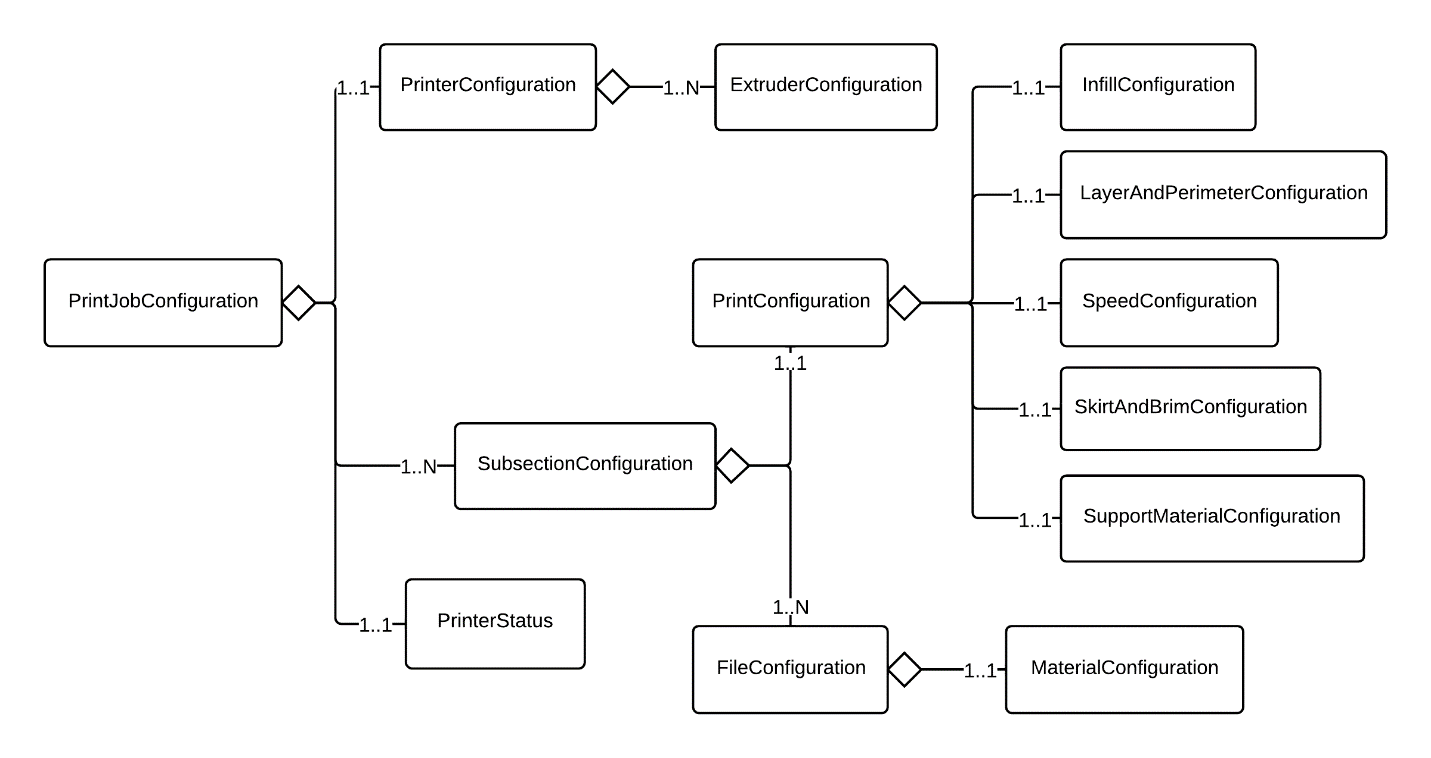


Figure 3‑1:Print Job Configuration Class Structure

### PrintJobConfiguration Class

#### Aggregation Relationships



Figure 3‑2 : Print Job Configuration UML

#### Data Elements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Description | Units | Boundaries |
| finalizedGCode | File | The final G-Code to print. This is set in the Post Processing Layer. | N/A | N/A |

Table 3‑1 : Print Job Configuration Data Elements

### PrinterConfiguration Class

#### Aggregation Relationships

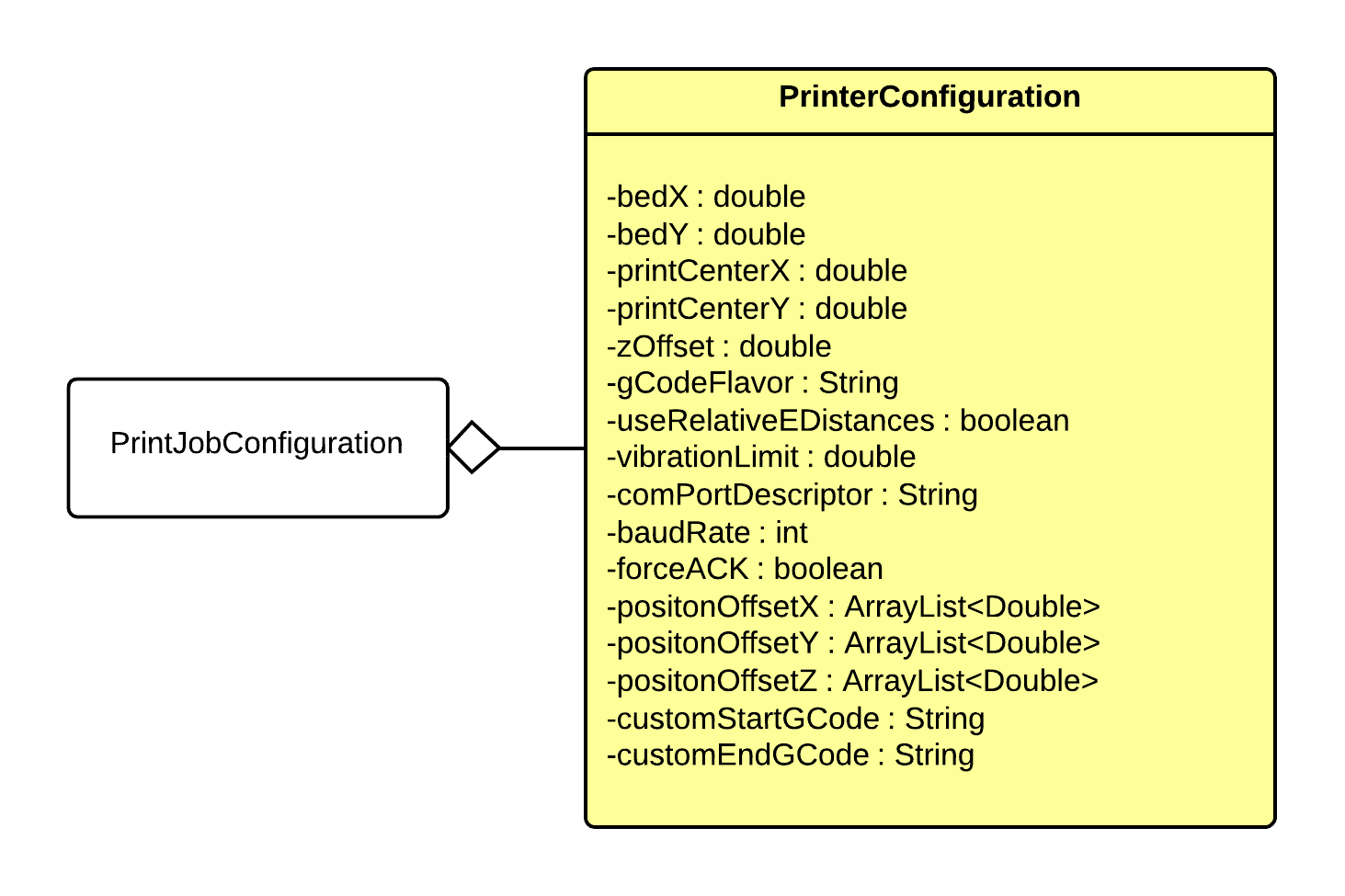


Figure 3‑3 : Printer Configuration UML

#### Data Elements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Description | Units | Boundaries |
| bedX | double | The max x of the bed. | mm | Greater than 0. |
| bedY | double | The max y of the bed. | mm | Greater than 0. |
| printCenterX | double | The x of the center of the print. | mm | 0 – bedX |
| printCenterY | double | The y of the center of the print. | mm | 0 – bedY |
| zOffset | double | The zOffset of the print bed surfaces. Used if the bed does not sit exactly at z = 0. | mm | Greater than or equal to 0. |
| gCodeFlavor | String | The G-Code flavor to output. | N/A | Within the set of available G-Code flavors. |
| useRelativeEDistances | boolean | When true, uses relative E values (required by some firmwares). | N/A | True or false |
| numExtruders | int | The number of extruders on the printer. | N/A | Greater than 0. |
| vibrationLimit | double | The limit of vibrations (in Hz) where movements will be slowed. If a move hits the specified vibration frequency, the extruder will slow. | Hz | Greater than or equal to 0. |
| comPortDescriptor | String | The descriptor for the com port the printer is connected to. Used to establish a connection to the printer. | N/A | N/A |
| baudRate | int | The baud rate for the printer. | Pulses per second | 0 - 250000 |
| forceACK | boolean | States whether or not to use ACK as part of the protocol when communicating with the printer. | N/A | True or false |
| positionOffsetX | ArrayList<Double> | The x offset to apply to the extruder positions for the printer. These offsets are stored in an ArrayList where index 0 represents the x offset for position 0, etc. | mm | Double precision floating point boundaries. |
| positionOffsetY | ArrayList<Double> | The y offset to apply to the extruder positions for the printer. These offsets are stored in an ArrayList where index 0 represents the y offset for position 0, etc. | mm | Double precision floating point boundaries. |
| positionOffsetZ | ArrayList<Double> | The z offset to apply to the extruder positions for the printer. These offsets are stored in an ArrayList where index 0 represents the z offset for position 0, etc. | mm | Double precision floating point boundaries. |
| customStartGCode | String | The custom G-Code to run when the printer starts. | N/A | G-Codes that are understood by the printer. |
| customEndGCode | String | The custom G-Code to run when the printer shuts down. | N/A | G-Codes that are understood by the printer. |

Table 3‑2 : Printer Configuration Data Elements

### PrinterStatus Class

#### Aggregation Relationships

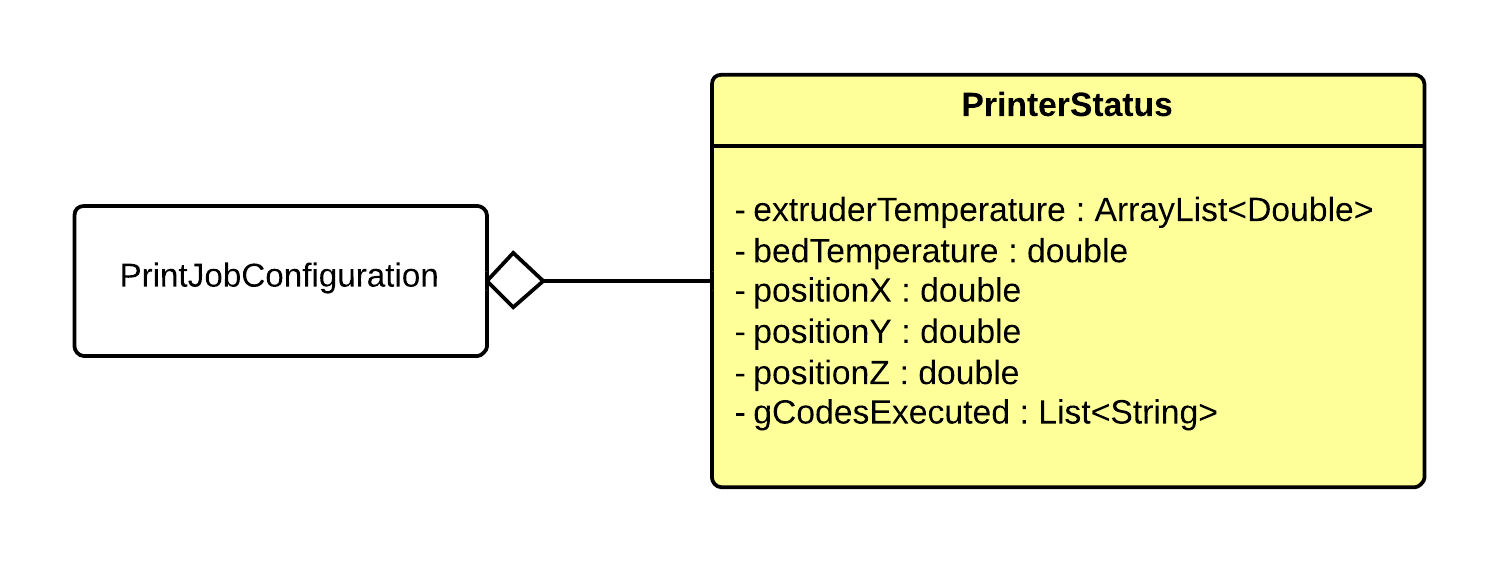


Figure 3‑4 : Printer Status UML

#### Data Elements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Description | Units | Boundaries |
| extruderTemperature | ArrayList<Double> | An ArrayList containing the current temperature of each extruder. | degrees C | Set by the material constraints of the materials in each extruder. |
| bedTemperature | double | The current temperature of the printer bed. | degrees C | Set by the printer configuration constraints. |
| positionX | double | The x position of the first extruder. | mm | Set by the printer configuration contraints. |
| positionY | double | The y position of the first extruder. | mm | Set by the printer configuration contraints. |
| positionZ | double | The z position of the first extruder. | mm | Set by the printer configuration contraints. |
| gCodesExectured | List  <String> | The last 5 GCodes executed. | N/A | N/A |

Table 3‑3 : Printer Status Data Elements

### SubsectionConfiguration Class

#### Aggregation Relationships



Figure 3‑5 : Subsection Configuration UML

#### Data Elements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Description | Units | Boundaries |
| bottomZ | double | The bottom of the subsection with relation to the entire object to be printed. | mm | Greater or equal to 0. |
| topZ | double | The top of the subsection with relation to the entire object to be printed. | mm | Greater than 0 and less than the total height of the object. |
| amfFile | File | The reference to the AMF file of the subsection. This is set during preprocessing. | N/A | N/A |
| gCodeFile | File | The reference to the gCode file of the subsection. This is set during processing. | N/A | N/A |

Table 3‑4: Subsection Configuration Data Elements

### PrintConfiguration Class

#### Aggregation Relationships

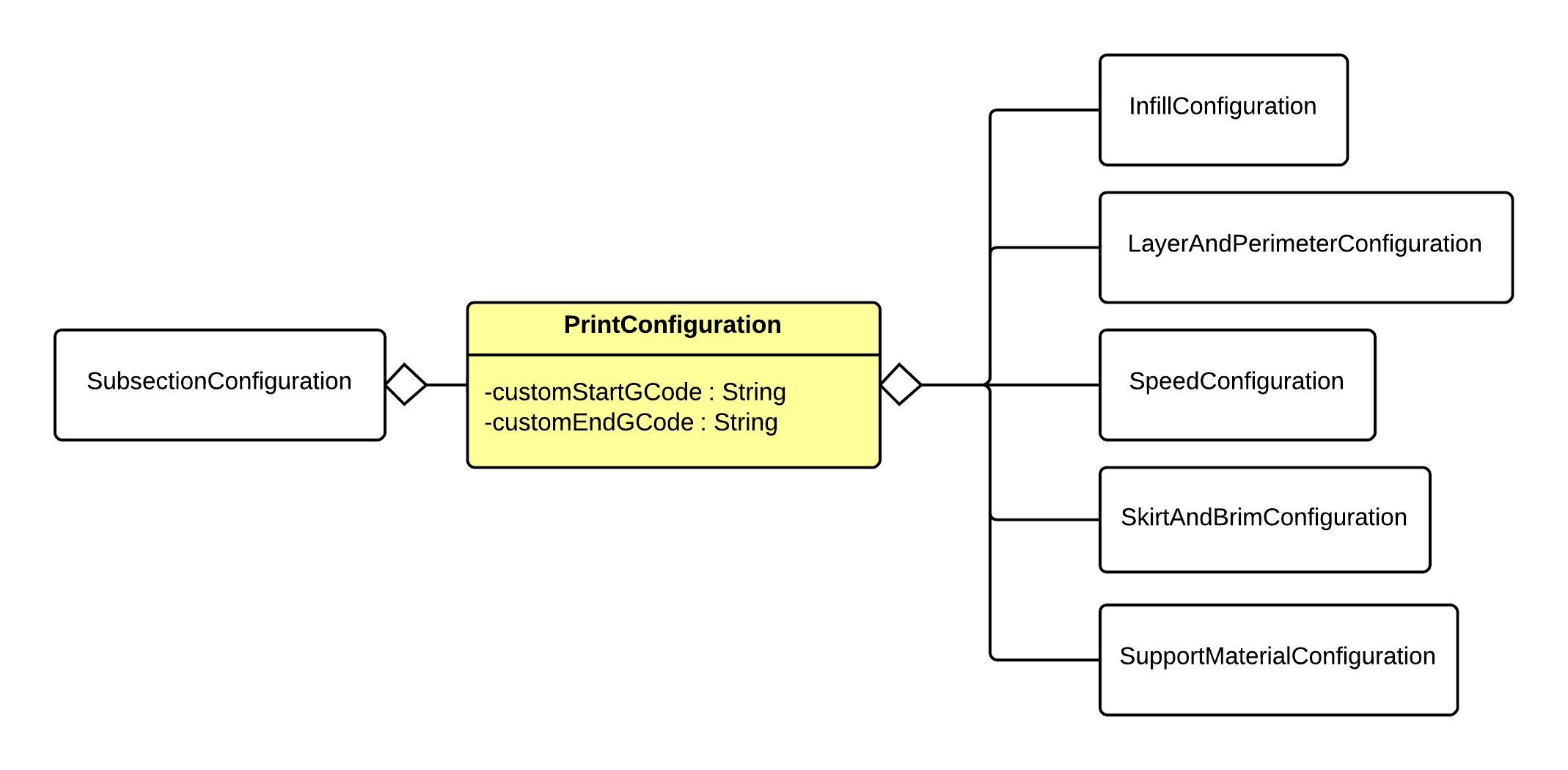


Figure 3‑6 : Print Configuration UML

#### Data Elements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Description | Units | Boundaries |
| customStartGCode | String | The custom G-Code to run when the print starts. | N/A | G-Codes that are understood by the printer. |
| customEndGCode | String | The custom G-Code to run when the print ends. | N/A | G-Codes that are understood by the printer. |

Table 3‑5 : Print Configuration Data Elements

### InfillConfiguration Class

#### Aggregation Relationships



Figure 3‑7 : Infill Configuration UML

#### Data Elements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Description | Units | Boundaries |
| infillDensity | double | The density of infill from 0.0 - 1.0. 0.0 being no infill, 1.0 being a solid infill. | Ratio | 0.0 – 1.0 |
| infillPattern | String | The pattern to use for internal infill. | N/A | Within set of available infill patterns. |
| topBottomInfillPattern | String | The pattern to use for the top and bottom layers' infill. | N/A | Within set of available infill patterns. |
| infillEveryNLayers | int | The ratio of layers to infill layers expresses as an integer >= 1. For example, infillEveryNLayers = 2 results in using infill every other layer, while infillEveryNLayers = 1 results in infill every layer. | N/A | Greater than or equal to 1. |
| onlyInfillWhereNeeded | boolean | When set to true, infill is treated as support material and only extruded where necessary. | N/A | True or false |
| solidInfillEveryNLayers | int | When set to a integer other than 0, a layer of solid infill with be extruded n layers, where the value of solidInfillEveryNLayers is n. | N/A | Greater than or equal to 0. |
| infillAngle | int | Default base angle for fill orientation in degrees from 0 to 359. This is the angle the infill will oriented in relation to the vertical perimeters. | degrees | 0 – 359 |
| solidInfillThresholdArea | int | The threshold for area in square mm for which to force solid infill. | mm2 | Greater or equal to 0. |
| onlyRetractInfillWhenCrossingPerimeters | boolean | When set to true, filament will not be retracted unless crossing a perimeter, resulting in some visible oozing throughout the infill. | N/A | True or false |
| infillBeforePerimeters | boolean | When set to true, infill for each layer will be extruded before the perimeters are extruded. | N/A | True or false |

Table 3‑6 : Infill Configuration Data Elements

### LayerAndPerimeterConfiguration Class

#### Aggregation Relationships



Figure 3‑8 : Layer and Perimeter Configuration UML

#### Data Elements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Description | Units | Boundaries |
| layerHeight | double | The height of each layer in mm. | mm | Greater than 0. |
| firstLayerHeight | double | The height of the first layer of the print in mm. | mm | Greater than 0. |
| perimeters | int | The number of vertical perimeters in the print. Essentially the number of "walls" around the perimeter of the print. | N/A | Greater than or equal to 0. |
| randomizedStartingPoints | boolean | If true, each layer should start from a different vertex to avoid build up on a specific corner. | N/A | True or false |
| generateExtraPerimetersWhenNeeded | boolean | If true, extra perimeters should be added in slopes where more than the specified number of perimeters is needed. | N/A | True or false |
| solidTopLayers | int | The number of solid layers to generate on the top of the print. | Layers | Greater than or equal to 0. |
| solidBottomLayers | int | The number of solid layers to generate on the bottom of the print. | Layers | Greater than or equal to 0. |

Table 3‑7 : Layer and Perimeter Configuration Data Elements

### SpeedConfiguration Class

#### Aggregation Relationships



Figure 3‑9 : Speed Configuration UML

#### Data Elements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Description | Units | Boundaries |
| perimetersSpeed | double | Speed for perimeters | mm/s | Greater than 0. |
| smallPerimetersSpeed | double | Speed for small perimeters | mm/s | Greater than 0. |
| externalPerimetersSpeed | double | Speed for external perimeters | mm/s | Greater than 0. |
| infillSpeed | double | Speed for infill | mm/s | Greater than 0. |
| solidInfillSpeed | double | Speed for solid infill | mm/s | Greater than 0. |
| topSolidInfillSpeed | double | Speed for top solid infill | mm/s | Greater than 0. |
| supportMaterialSpeed | double | Speed for support material | mm/s | Greater than 0. |
| bridgesSpeed | double | Speed for bridges | mm/s | Greater than 0. |
| gapFillSpeed | double | Speed for gap fill | mm/s | Greater than 0. |
| nonPrintMovesSpeed | double | Speed for non prdouble movements | mm/s | Greater than 0. |
| firstLayerSpeed | double | Speed for the first layer | mm/s | Greater than 0. |
| perimetersAcceleration | double | Acceleration for perimeters | mm/s2 | Greater than or equal to 0. |
| infillAcceleration | double | Acceleration for infill | mm/s2 | Greater than or equal to 0. |
| bridgeAcceleration | double | Acceleration for bridges | mm/s2 | Greater than or equal to 0. |
| defaultAcceleration | double | Default Acceleration | mm/s2 | Greater than or equal to 0. |

Table 3‑8 : Speed Configuration Data Elements

### SkirtAndBrimConfiguration Class

#### Aggregation Relationships



Figure 3‑10 : Skirt and Brim Configuration UML

#### Data Elements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Description | Units | Boundaries |
| skirtLoops | int | The number loops of skirt to extrude (0 will extrude no skirt). | Loops | Greater than or equal to 0. |
| skirtDistanceFromObject | double | The distance from the object the skirt will be extruded at in mm >= 0. Setting this to 0 will essentially turn the skirt into brim. | mm | Greater than or equal to 0. |
| skirtHeight | int | The height of the skirt in layers >= 1. | Layers | Greater than or equal to 1. |
| skirtMinimumExtrusionLength | double | The minimum extrusion length of the skirt in mm >= 0. | mm | Greater than or equal to 0. |
| brimWidth | double | The width of the brim in mm (0 will extrude no brim). | mm | Greater than or equal to 0. |

Table 3‑9 : Skirt and Brim Configuration Data Elements

### SupportMaterialConfiguration Class

#### Aggregation Relationships



Figure 3‑11 : Support Material Configuration UML

#### Data Elements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Description | Units | Boundaries |
| generateSupportMaterial | boolean | When set to true, G-code for support material will be generated. | N/A | True or false |
| overhangThreshold | int | The overhang threshold in degrees. Support material will not be for overhangs whose slope angle is above this threshold. | degrees | 0 – 180 |
| enforceSupportForFirstNLayers | int | Forces support material on the first n layers. | Layers | Greater than or equal to 0. |
| raftLayers | int | Number of raft layers to print below the object. | Layers | Greater than or equal to 0. |
| supportMaterialPattern | String | Pattern used to generate support material. | N/A | Within set of available infill patterns. |
| supportPatternSpacing | double | The spacing in mm between support lines. | mm | Greater than or equal to 0. |
| supportPatternAngle | int | The angle the support pattern is extruded at (between 0 and 359). | degrees | 0 – 359 |
| interfaceLayers | int | The number of interface layers to print between the raft and object. | Layers | Greater than or equal to 0. |
| interfacePatternSpacing | double | The spacing in mm between support lines. | mm | Greater than or equal to 0. |

Table 3‑10 : Support Material Configuration Data Elements

### FileConfiguration Class

#### Aggregation Relationships

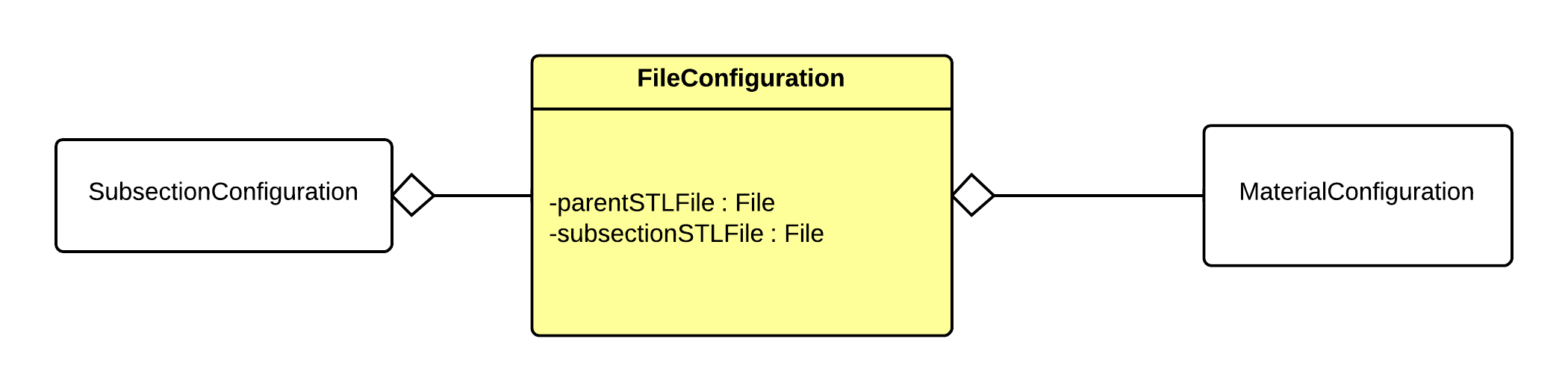


Figure 3‑12 : File Configuration UML

#### Data Elements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Description | Units | Boundaries |
| parentSTLFile | File | The reference to the parent STL that will be subsectioned in the preprocessing layer. | N/A | N/A |
| subsectionSTLFile | File | The reference to the STL file that represents the subsection of the parent STL file. This is set in the preprocessing layer. | N/A | N/A |

Table 3‑11 : File Configuration Data Elements

### MaterialConfiguration Class

#### Aggregation Relationships

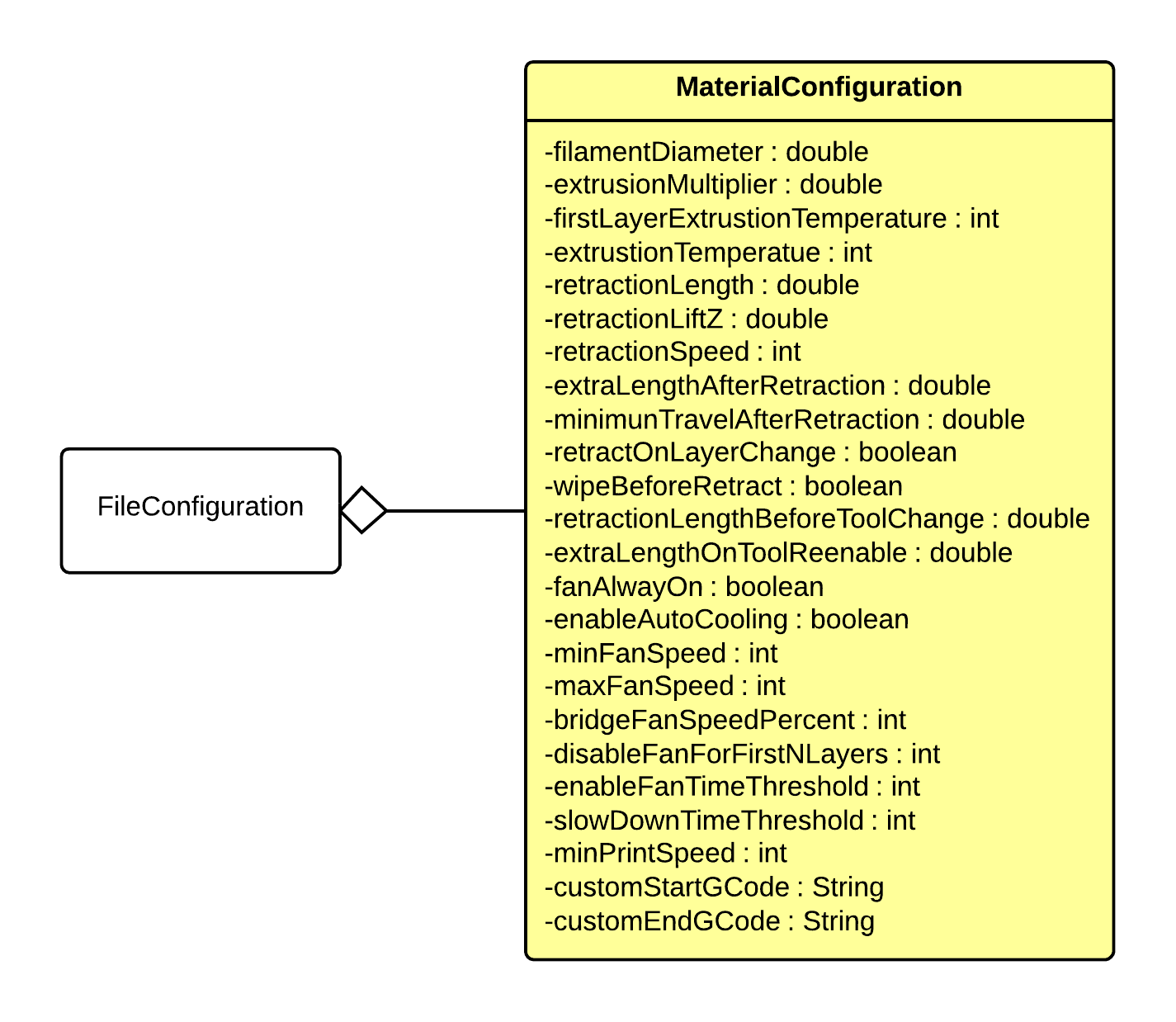


Figure 3‑13 : Material Configuration UML

#### Data Elements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Description | Units | Boundaries |
| filamentDiameter | double | Diameter in mm of the filament. | mm | Greater than 0. |
| extrusionMultiplier | double | Flow rate multiplier. This changes the flow rate proportionally. 0.9 will be 90% flow rate, while 1.1 will be 110% flow rate. | Ratio | Greater than 0.0. |
| firstLayerExtrusionTemperature | int | The temperature (in degrees C) the extruder needs to be to extrude the first layer of this material. | degrees C | Within extruder temperature range. |
| extrusionTemperature | int | The temperature (in degrees C) the extruder needs to be to extrude this material. | degrees C | Within extruder temperature range. |
| retractionLength | double | Length (in mm) to retract during retraction. 0 to disable retraction. | mm | Greater than or equal to 0. |
| retractionLiftZ | double | The (positive) z value to quickly lift the extruder by during a retraction. | mm | Greater than or equal to 0. |
| retractionSpeed | int | The speed (in mm/s) at which to retract the filament during retraction (extruder motor speed). | mm/s | Greater than or equal to 0. |
| extraLengthAfterRetraction | double | The extra length of filament to push out during the first extrude after a retraction. | mm | Greater than or equal to 0. |
| minimumTravelAfterRetraction | double | Retraction is not triggered when travel moves shorter than this distance. | mm | Greater than or equal to 0. |
| retractOnLayerChange | boolean | When set to true, retraction will be triggered on each layer change. | N/A | True or false |
| wipeBeforeRetract | boolean | When set to true, the nozzle will be moved while retracting to reduce blob. | N/A | True or false |
| retractionLengthBeforeToolChange | double | Length (in mm) to retract when the tool is disabled. | mm | Greater than or equal to 0. |
| extraLengthOnToolReenable | double | The extra length of filament to push out during the first extrude after the tool is re-enabled. | mm | Greater than or equal to 0. |
| fanAlwaysOn | boolean | When set to true, fan will always run at at least minimum speed. | N/A | True or false |
| enableAutoCooling | boolean | When set to true, fan speed will automatically be set based on printing time. | N/A | True or false |
| minFanSpeed | int | The minimum fan speed in PWM. | PWM | 0 – maxFanSpeed |
| maxFanSpeed | int | The maximum fan speed in PWM. | PWM | minFanSpeed – 100 |
| bridgeFanSpeedPercent | int | The percentage of default fan speed used for bridges expressed as an int (100 = 100%) | % | 0 – 100 |
| disableFanForFirstNLayers | int | The number of first layers to disable the fan for. | Layers | Greater than or equal to 0. |
| enableFanTimeThreshold | int | If the print time of a layer is below this threshold (in seconds), the fan will be activated. | Seconds | Greater than or equal to 0. |
| slowDownTimeTreshold | int | If the print time of a layer is below this threshold (in seconds), the move speed will be slowed to attempt to get the layer print time up to this threshold. | Seconds | Greater than or equal to 0. |
| minPrintSpeed | int | The move speed will not be scaled down below this speed. | mm/s | Greater than 0. |
| customStartGCode | String | The custom G-Code to run when the material starts printing. | N/A | G-Codes that are understood by the printer. |
| customEndGCode | String | The custom G-Code to run when the material stops printing. | N/A | G-Codes that are understood by the printer. |

Table 3‑12 : Material Configuration Data Elements

### ExtruderConfiguration Class

#### Aggregation Relationships

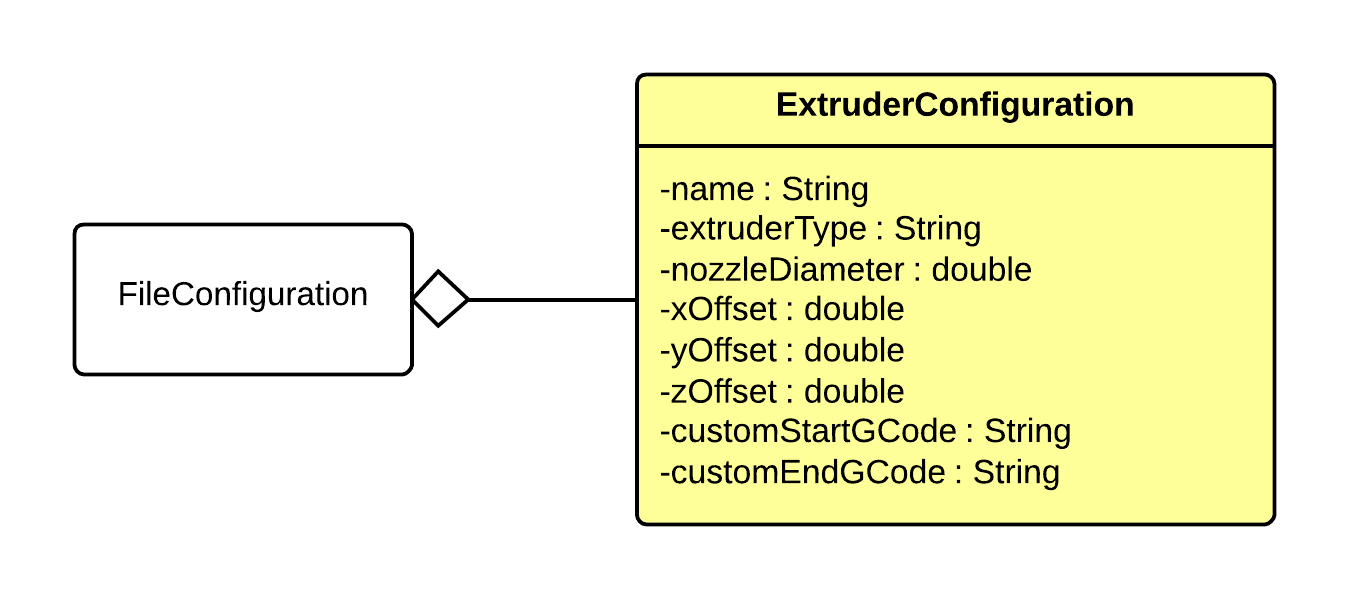


Figure 3‑14 : Extruder Configuration UML

#### Data Elements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Description | Units | Boundaries |
| extruderType | String | The type of extruder. | N/A | Within the set of available extruder types. |
| nozzleDiameter | double | The diameter of the nozzle in mm. | mm | Greater than or equal to 0. |
| xOffset | double | The x offset in respect to the position on the printer. | mm | Greater than or equal to 0. |
| yOffset | double | The y offset in respect to the position on the printer. | mm | Greater than or equal to 0. |
| zOffset | double | The z offset in respect to the position on the printer. | mm | Greater than or equal to 0. |
| customStartGCode | String | The custom G-Code to run when the extruder is activated for extrusion. | N/A | G-Codes that are understood by the printer. |
| customEndGCode | String | The custom G-Code to run when the extruder is deactivated after extrusion. | N/A | G-Codes that are understood by the printer. |

Table 3‑13 : Extruder Configuration Data Elements

# User Interface Layer

The Interface Layer’s purpose is to collect user information and serve it to the processing layers. There are three subsystems providing data to lower levels. The GUI subsystem provides the user the ability to interact with the system and provides the ability to import files, create material profiles, set general configuration, and print specific configuration. The database subsystem provides a persistence framework for the other subsystems in this layer for storage and retrieval of configuration and material information. The controller subsystem decouples the user interface from the program logic. The controller provides the GUI with the information to present and takes in the user actions allowing the initiation of a print run, then packing the collected configuration, object files, and material data in to a print request object for the preprocessing layer.

## GUI Subsystem Modules

The GUI Subsystem is responsible for providing an interface to the user so that he or she can import model files and manage print input configuration information related to materials and printing hardware. The GUI Subsystem is also responsible for providing an interface from which the user can set configuration options for a specific print and initiate a print. During an ongoing print, The GUI Subsystem is responsible for displaying the printer and current print status to the user.

### Import GUI Module

#### Prologue

Import GUI Module is in charge of generating the “Import” menu display that the user interacts with. The menu will provide options on screen such as import and delete.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| importButton | Button Press and string file name | Message to user of success or failure |
| deleteButton | Button Press and string file name | Message to user of success or failure |

Table 4‑1 : Import GUI Module Interfaces

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
| User Input | User |
| STL File | Storage Device (handled by OS) |

Table 4‑2 : Import GUI Module External Data Dependencies

#### Internal Data Descriptors

None

#### Process

importTab(){

Add tab to main window frame named "Model Import";

Add button to importTab.pane named "Import" with function importController.import();

Add list display; Populate it with importController.viewModelDB();

Add button to importTab.pane named "Delete" with function importController.delete(model);

}

### Material Configuration GUI Module

#### Prologue

Material GUI Module is a menu screen where the user can insert the descriptors for new material that is not present in the database. The menu also provides the utility for the user to change the description of any existing material in the database or remove a material from the database.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| newButton | Button Press and string of file name | Display update |
| saveButton | Button Press and string name of material and data to populate the MaterialConfiguration object | Message to user of success or failure |
| deleteButton | Button Press and string of file name | Message to user of success or failure |

Table 4‑3 : Material Configuration GUI Module Interfaces

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
| User Input | User |

Table 4‑4 : Material Configuration GUI Module External Data Dependencies

#### Internal Data Descriptors

None

#### Process

materialTab(){

Add tab to main window frame named "Materials";

empty = new string;

Add button to materialTab.pane named "New" with function populateScreen(empty);

Add list display; Populate it with materialController.loadAvailableMaterials();

Add event to list: when item is clicked, call function populateScreen( materialController.load(material) );

Add button to materialTab.pane named "Save" with function materialController.save(materialStringArray);

Add button to materialTab.pane named "Delete" with function materialController.delete(material);

}

populateScreen(material){

//displays the information on screen into the data input boxes from the string array "material"

}

### Printer Configuration GUI Module

#### Prologue

Printer Configuration GUI Module is a menu that lets the user enter information that will let the system understand the dimensions and capabilities of the 3D printer. The menu also permits the user to save configuration pre-sets to be used again later.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| newButton | Button Press and string of file name | Display update |
| saveButton | String name of file and data to populate the PrinterConfiguration Object | Message to user of success or failure |
| deleteButton | Button press and string of file name | Message to user of success or failure |

Table 4‑5 : Printer Configuration GUI Module Interfaces

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
| User Input | User |

Table 4‑6 : Printer Configuration GUI Module External Data Dependencies

#### Internal Data Descriptors

None

#### Process

printerConfigTab(){

Add tab to main window frame named "Printer Hardware Settings";

empty = new string;

Add button to printerConfigTab.pane named "New" with function populate(empty);

Add drop down list display; Populate it with printerConfigController.loadAvailablePrinters();

Add button to printerConfigTab.pane named "Save" with function printerConfigController.save(printerConfigStringArray);

Add button to printerConfigTab.pane named "Delete" with function printerConfigController.delete(printerConfig);

Add button to printerConfigTab.pane named "Extruder Positions" with function extrudePositionWindow();

}

populate(printerConfig){

//populates the window with the appropriate set of printer configuration info

}

extrudePositionWindow(){

Creates new window frame named "Extruder Positions";

Add text to extrudeposition.frame ("Number of Extruders ");

Add input box that accepts ints, stored as "count";

Add text to extrudeposition.frame ("Extruder Position 0: " + "0");

for(i = 2 ; i < count; i++)

Add text to extrudeposition.frame ("Extruder Position " + i + ": ");

Add input box that accepts ints, stored as "position[i]";

Add button to extrudeposition.frame named "Save" with function printerConfigController.saveExtruderPosition(count, position[]);

}

### Extruder Configuration GUI Module

#### Prologue

Extruder Configuration GUI Module is a menu that lets the user enter and save custom configurations for each extruder they own.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| newButton | Button Press and string name of file | Display update |
| saveButton | Button Press and string name of file and data to populate the ExtruderConfiguration Object | Message to user of success or failure |
| deleteButton | Button Press and string name of file | Message to user of success or failure |

Table 4‑7 : Extruder Configuration GUI Module Interfaces

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
| User Input | User |

Table 4‑8 : Extruder Configuration GUI Module External Data Dependencies

#### Internal Data Descriptors

None

#### Process

extruderConfigTab(){

Add tab to main window frame named "Extruder Configs";

Add button to printConfigTab.pane named "New" with function printerConfigController.new();

Add drop down list display; Populate it with extruderConfigController.loadAvailableExtruders();

Add button to extruderConfigTab.pane named "Save" with function extruderConfigController.save(extruderConfigStringArray);

Add button to extruderConfigTab.pane named "Delete" with function extruderConfigController.delete(extruderConfig);

}

### Print Configuration GUI Module

#### Prologue

Print Configuration GUI Module is a menu that lets the user enter and save a set information that will let the system understand how the user wants a subsection of the print job to run.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| newButton | Button Press and string name of file | Display update |
| saveButton | Button Press and string name of file and data to populate the PrintConfiguration Object | Message to user of success or failure |
| deleteButton | Button Press and string name of file | Message to user of success or failure |

Table 4‑9 : Print Configuration GUI Module Interfaces

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
| User Input | User |

Table 4‑10 : Print Configuration GUI Module External Data Dependencies

#### Internal Data Descriptors

None

#### Process

printConfigTab(){

Add tab to main window frame named "Print Configs";

Add button to printConfigTab.pane named "New" with function printerConfigController.new();

Add drop down list display; Populate it with printConfigController.loadAvailablePrints();

Add button to printConfigTab.pane named "Save" with function printConfigController.save(printConfigStringArray);

Add button to printConfigTab.pane named "Delete" with function printConfigController.delete(printConfig);

}

### Print Job GUI Module

#### Prologue

The Print Job GUI Module is a menu that allows the user to set up a print job by dividing the print job into subsections. The user selects models from the imported STL files to be printed and maps them with appropriate materials listed from the database.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| loadPrintJobConfiguration | Button Press and string name of file | Display update |
| newSubsection | Button Press | Display update |
| savePrintJobConfiguration | Button Press and string name of file and data to populate the PrintJobConfiguration object | Message to user of success or failure |
| deleteSubsection | Button Press and subsection Index | Display update |

Table 4‑11 : Print Job GUI Module Interfaces

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
| User Input | User |

Table 4‑12 : Print Job GUI Module External Data Dependencies

#### Internal Data Descriptors

None

#### Process

printJobTab(){

Add tab to main window frame named "Print Job";

Add inner window pane with scroll bar;

Add button to inner window pane named "New Subsection" with function newSubsection;

Add button named "Start Print" that calls printJobController.startPrint()

}

newSubsection(){

Add subsection.frame inside inner window pane;

Add text to subsection.frame ("Subsection\n from " + input box + "to " + input box );

Add text to subsection.frame ("Print Config: ");

Add drop down box to subsection.frame. Drop down's contents = Print Config from database;

Add grid like input box with scroll bar;

Add button to subsection.frame; button.action() = closes subsection.frame;

}

### Status GUI Module

#### Prologue

Status GUI Module is a window that displays the current status of the print job while the print job is ongoing. The window will also include a button that allows the user to pause and resume the printing process or cancel the print job.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| pauseResumeButton | Button Press | Display Update |
| cancelButton | Button Press | Message to user of success or failure |

Table 4‑13 : Status GUI Module Interfaces

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
| User Input | User |

Table 4‑14 : Status GUI Module External Data Dependencies

#### Internal Data Descriptors

None

#### Process

statusGUI(){

New window frame named "Printing Status";

Add leftStatus.pane to statusGUI.frame;

Add text to leftStatus.pane ("Temperature: \nx: \ny: \nz: ");

Add rightStatus.pane to statusGUI.frame;

For each update from statusController

Clear text in rightStatus.pane;

Coordinates[] = StatusController.updateStatus();

Add text to rightStatus.pane (Coordinates[0] + "\n" + Coordinates[1] + "\n" + Coordinates[2] + "\n" + Coordinates[3];

Add button to status.frame named “Pause” with the function statusController.pause();

Add event to “Pause button;

Event = changes name of “Pause” to resume and function to statusController.resume();

Add button to status.frame named “Cancel” with the function statusController.cancel();

}

## Controller Subsystem Modules

The Controller Subsystem is responsible for carrying out the functions that write or delete files when called by the GUI. When the print job beings, the Controller Subsystem is responsible for collecting all the material and printer configuration information required for the print, bundling that information with the object file information received from the Import Subsystem, and sending that bundle to the Pre Processing Layer to begin the printing process. The Controller Subsystem is also responsible for telling the Printer State Controller to pause, resume, or stop a print job based on user input.

### Import Controller

#### Prologue

Import Controller Module carries out the functions of importing or removing STL files from the system when called by the Import GUI Module.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| importSTL | The file name of the STL file | Boolean success state |
| delete | The file name of the STL file | Boolean success state |

Table 4‑15 : Import Controller Module Interfaces

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
| STL File | Storage Device (handled by OS) |

Table 4‑16 : Import Controller Module External Data Dependencies

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| String File Name | Import GUI Module |

Table 4‑17 : Import Controller Module Internal Data Dependencies

#### Process

import(model){

Call function framework.save(model);

}

delete(model){

Call function framework.delete(model);

}

### Material Configuration Controller

#### Prologue

Material Controller Module carries out the functions related to storing and managing material configuration. This includes storing the user input in a MaterialConfiguration object, modifying a MaterialConfiguration object when new user input is given, or deleting a stored MaterialConfiguration object upon request.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| saveMaterialConfiguration | MaterialConfiguration Object and string file name | Boolean success state |
| loadMaterialConfiguration | String name of file | MaterialConfiguration Object |
| deleteMaterialConfiguration | String name of file | Boolean success state |
| loadAvailableMaterials | None | ArrayList<String> of file names |

Table 4‑18 : Material Controller Module Interfaces

#### External Data Dependencies

None

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| Material Data (string array) | Material Configuration GUI |
| Material ArrayList<String> of names | PersistanceFramework |
| String file name to operate on | Material Configuration GUI |

Table 4‑19 : Material Controller Module Internal Data Descriptors

#### Process

load(material){

call function framework.load(material);

}

save(material){

call function framework.save(material);

}

delete(material){

call function framework.delete(material);

}

loadAvailableMaterials(){

call function framework.loadAvailableMaterials ();

}

### Printer Configuration Controller

#### Prologue

The Printer Configuration Controller Module carries out the functions that allow the user to store and retrieve information that the specify the capabilities of the 3D printer. These functions include storing the user input as a PrinterConfiguration object, modifying a PrinterConfiguration that describes the printer hardware, or deleting a PrinterConfiguration upon request.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| savePrinterConfiguration | PrinterConfiguration Object and string file name | Boolean success state |
| loadPrinterConfiguration | String name of file | PrinterConfiguration Object |
| deletePrinterConfiguration | String name of file | Boolean success state |
| loadAvailablePrinters | None | ArrayList<String> of file names |
| getSerialPortEnumeration | None | ArrayList<String> of port discriptors |

Table 4‑20 : Printer Controller Module Interfaces

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
| Serial Port Enumeration | OS through Javax.comms |

Table 4‑21 : Printer Controller Module External Data Dependencies

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| Printer ArrayList<String> of names | PersistanceFramework |
| Printer Configuration String Array | Printer Configuration GUI |
| String File name to operate on | Printer Configuration GUI |

Table 4‑22 : Printer Controller Module Internal Data Descriptors

#### Process

loadAvailablePrinters(){

call framework.loadAvailablePrinters();

}

Save(printerConfigStringArray){

call framework.save(printerConfigStringArray);

}

delete(printerConfig){

call framework.delete(printerConfig);

}

saveExtruderPosition(count, position[]){

call framework.saveExtruderPosition(count, position[]);

}

### Extruder Configuration Controller

#### Prologue

The Extruder Configuration Controller Module carries out the functions that store or retrieve an ExtruderConfiguration object. These functions include storing the user input as an ExtruderConfiguration object, modifying an ExtruderConfiguration object that describes the print configuration, or deleting an ExtruderConfiguration object upon request.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| saveExtruderConfiguration | ExtruderConfiguration Object and string file name | Boolean success state |
| loadExtrudeConfiguration | String name of file | ExtruderConfiguration Object |
| deleteExtruderConfiguration | String name of file | Boolean success state |
| loadAvailableExtruders | None | ArrayList<String> of file names |

Table 4‑23 : Extruder Configuration Controller Interfaces

#### External Data Dependencies

None

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| ExtruderConfig (String Array) | ExtruderConfigurationGui |
| String File name to operate on | ExtruderConfigurationGui |
| Print ArrayList<String> of names | PersistanceFramework |

Table 4‑24 : Extruder Configuration Controller Internal Data Descriptors

#### Process

new(){

}

loadAvailableExtruders(){

call function framework.loadAvailableExtruders();

}

save(ExtruderConfig){

call function framework.save(ExtruderConfig);

}

delete(ExtruderConfig){

call function framework.delete(ExtruderConfig);

}

### Print Configuration Controller

#### Prologue

The Print Configuration Controller Module carries out the functions that allow the user store or retrieve a PrintConfiguration object. These functions include storing the user input as a PrintConfiguration object, modifying a PrintConfiguration object that describes the print configuration, or deleting PrintConfiguration object upon request.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| savePrintConfiguration | PrintConfiguration Object and string file name | Boolean success state |
| loadPrintConfiguration | String name of file | PrintConfiguration Object |
| deletePrintConfiguration | String name of file | Boolean success state |
| loadAvailablePrints | None | ArrayList<String> of file names |

Table 4‑25 : Print Configuration Controller Interfaces

#### External Data Dependencies

None

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| PrintConfig (String Array) | PrintConfigurationGui |
| String File name to operate on | PrintConfigurationGui |
| Print ArrayList<String> of names | PersistanceFramework |

Table 4‑26 : Print Configuration Controller Internal Data Descriptors

#### Process

new(){

}

loadAvailablePrints(){

call function framework.loadAvailablePrints();

}

save(PrintConfigStringArray){

call function framework.save(PrintConfigStringArray);

}

delete(PrintConfig){

call function framework.delete(PrintConfig);

}

### Print Job Controller

#### Prologue

The Print Job Controller Module plays an integral role in preparing the object file for the other layers to interpret. The Print Job Controller Module receives the information gathered by the Print Job GUI Module, which includes the STL files of the models, the size of the subsections, the print configuration for each subsection, the relationship mapping chosen between the materials and each model, and the printer hardware configuration,. The Print Job Controller Module then compiles this information into an object that the other layer’s modules will reference throughout the processing sequence. The last step, the print job controller initiates the preprocessor.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| loadAvailablePrintJobs | None | ArrayList<String> of file names |
| loadAvailablePrinters | None | ArrayList<String> of file names |
| loadAvailablePrints | None | ArrayList<String> of file names |
| loadAvailableExtruders | None | ArrayList<String> of file names |
| loadAvailableMaterials | None | ArrayList<String> of file names |
| loadPrintJobConfiguration | String file name | PrintJobConfiguration object |
| savePrintJobConfiguration | String file name and data objects to populate the PrintJobConfiguration object | Boolean success state |
| startPrint | PrintJobConfiguration Object | Opened StatusGUI |

Table 4‑27 : Print Job Controller Interfaces

#### External Data Dependencies

None

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| PrintJob (Object Array) | PrintJobGui |
| String File name to operate on | PrintJobGui |
| Print ArrayList<String> of names | PersistanceFramework |
| PrintConfiguration Objects | PersistanceFramework |
| PrinterConfiguration Objects | PersistanceFramework |
| MaterialConfiguration Objects | PersistanceFramework |
| ExtruderConfiguration Objects | PersistanceFramework |

Table 4‑28 : Print Job Controller Internal Data Descriptors

#### Process

startPrint(){

Store all the following information into the configuration file;

* + - * Model file(s)
      * Array of the subsection sizes (int)
      * Print Configuration for each subsection (xml)
      * Printer Configuration for the hardware (xml)
      * All related material info (xml)
      * Array map of material to models (int)

call the function StatusGUI.statusGUI();

}

### Status Controller

#### Prologue

The Status Controller Module listens for feedback from the Communication Layer and passes the values it receives to the Status GUI module to be displayed. These values include the temperature of the nozzle (described as a float or double type) and the nozzle’s current position (described as a vec3, which is 3 float 32’s). The Status Controller Module also passes a pause or resume command to the Printer Controller Layer upon request.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| updateStatus | printerStatus object | Data value updates |
| pauseResumeJob | CurrentStateFlag | Boolean success state |
| cancelPrintJob | None | Boolean success state |

Table 4‑29 : Status Controller Interfaces

#### External Data Dependencies

***None***

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| printerStatus object | StateControllerModule |
| Button Press event object | StatusGUI |

Table 4‑30 : Status Controller Internal Data Desctiptors

#### Process

updateStatus(printerStatusObject){

new int Coordinates[];

Coordinates[0] = temperature;

Coordinates[1] = x;

Coordinates[2] = y;

Coordinates[3] = z;

Return Coordinates[];

}

pause(){

//Call function from printer controller to pause

If successful, success = 1;

Return success;

}

resume(){

//Call function from printer controller to resume

If successful, success = 1;

Return success;

}

cancel(){

//call function from printer controller to cancel

If successful, success = 1;

Return success;

}

## Database Subsystem Modules

The Database Interface subsystem is responsible for providing an abstract interface between the database of the system and any subsystems that need to store or retrieve information from the database. As such, the Database Interface Subsystem is responsible for exposing all the methods necessary for the other subsystems to communicate with the database in an abstract manner.

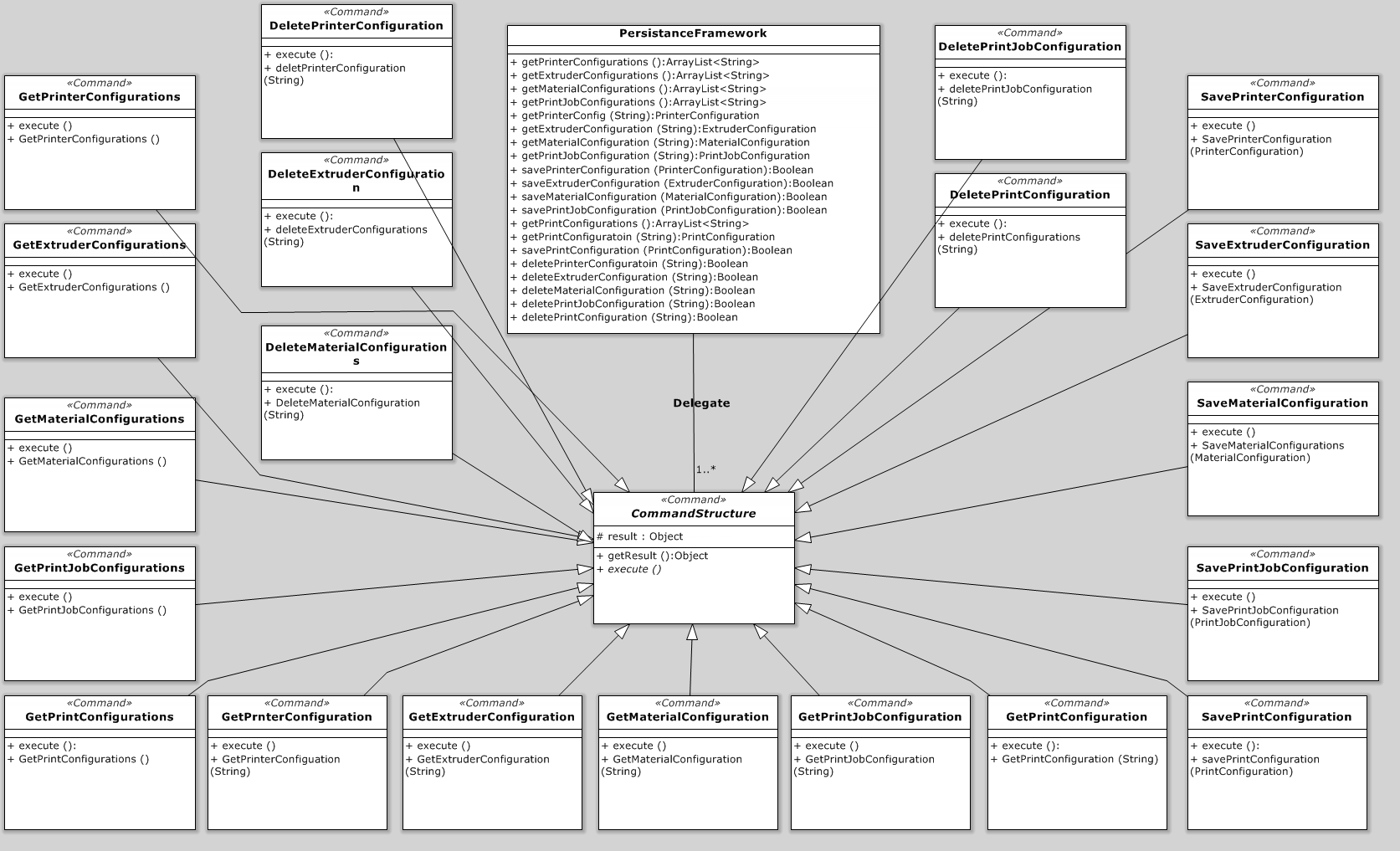


Figure 4‑1: Database Subsystem UML

### Persistence Framework

#### Prologue

The Persistence Framework Module is responsible for all storage and retrieval of configuration objects. The purpose of this module is to provide a single point of contact for the application to manage stored data. The Persistence Framework creates instances of the commands from the command structure to create configuration objects and return them to the controller that requests the object. It also receives objects to store which will then be passed to the command structure to store in the correct format and location.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| getPrinterConfigurations | None | ArrayList<String> of names |
| getExtruderConfigurations | None | ArrayList<String> of names |
| getMaterialConfigurations | None | ArrayList<String> of names |
| getPrintJobConfigurations | None | ArrayList<String> of names |
| getPrintConfigurations | None | ArrayList<String> of names |
| getPrinterConfig | String name of the file | PrinterConfiguration Object |
| getExtruderConfig | String name of the file | ExtruderConfiguration Object |
| getMaterialConfig | String name of the file | MaterialConfiguration Object |
| getPrintJobConfiguration | String name of the file | PrintJobConfiguration Object |
| getPrintConfiguration | String name of the file | PrintConfiguration Object |
| savePrinterConfiguration | PrinterConfiguration Object | Boolean success state |
| saveExtruderConfiguration | ExtruderConfiguration Object | Boolean success state |
| saveMaterialConfiguration | MaterialConfiguration Object | Boolean success state |
| savePrintJobConfiguration | PrintJobConfiguration Object | Boolean success state |
| savePrintConfiguration | PrintConfiguration Object | Boolean success state |
| deletePrinterConfiguration | String name of the file | Boolean success state |
| deleteExtruderConfiguration | String name of the file | Boolean success state |
| deleteMaterialConfiguration | String name of the file | Boolean success state |
| deletePrintJobConfiguration | String name of the file | Boolean success state |
| deletePrintConfiguration | String name of the file | Boolean success state |

Table 4‑31 : Persistence Framework Interfaces

#### External Data Dependencies

***None***

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| PrinterConfiguration Object | Printer Hardware Configuration Controller |
| ExtruderConfiguration Object | Extruder Controller |
| MaterialConfiguration Object | Material Controller |
| PrintJobConfiguration Object | Print Job Controller |
| PrintConfiguration Object | Print Controller |

Table 4‑32 : Persistence Framework Internal Data Descriptors

#### Process

getPrinterConfigurations(){

Create GetPrinterConfigurations command

Execute GetPrinterConfigurations command

Return result from GetPrinterConfigurations command

}

savePrinterConfiguration(){

Create SavePrinterConfiguration Command

Execute SavePrinterConfiguration Command

Check result of SavePrinterConfiguration Command

If true continue

Else throw error

}

### Command Structure

#### Prologue

The Command Structure module is a collection of very small classes that each does one simple operation. All of the operations deal with data stored on the hard drive database. This database is a structure of folders in the running directory of the program that contain xml file representations of the configuration objects used by the system. There are two kinds of get commands. First, the Get…Configurations commands read the contents of a folder and return the file names in the different configuration folders for the User Interface layer to create selection boxes. Second, the Get…Configuration commands take a string file name and return the correct object based on the xml file selected. The Save commands all convert the configurations to xml and store them on the hard drive in the correct folder locations. The Delete commands take the file name and delete the file from the folder structure.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| getPrinterConfigurations | None | ArrayList<String> of names |
| getExtruderConfigurations | None | ArrayList<String> of names |
| getMaterialConfigurations | None | ArrayList<String> of names |
| getPrintJobConfigurations | None | ArrayList<String> of names |
| getPrintConfigurations | None | ArrayList<String> of names |
| getPrinterConfig | String name of the file | PrinterConfiguration Object |
| getExtruderConfig | String name of the file | ExtruderConfiguration Object |
| getMaterialConfig | String name of the file | MaterialConfiguration Object |
| getPrintJobConfiguration | String name of the file | PrintJobConfiguration Object |
| getPrintConfiguration | String name of the file | PrintConfiguration Object |
| savePrinterConfiguration | PrinterConfiguration Object | Boolean success state |
| saveExtruderConfiguration | ExtruderConfiguration Object | Boolean success state |
| saveMaterialConfiguration | MaterialConfiguration Object | Boolean success state |
| savePrintJobConfiguration | PrintJobConfiguration Object | Boolean success state |
| savePrintConfiguration | PrintConfiguration Object | Boolean success state |
| deletePrinterConfiguration | String name of the file | Boolean success state |
| deleteExtruderConfiguration | String name of the file | Boolean success state |
| deleteMaterialConfiguration | String name of the file | Boolean success state |
| deletePrintJobConfiguration | String name of the file | Boolean success state |
| deletePrintConfiguration | String name of the file | Boolean success state |

Table 4‑33 : Command Structure Interfaces

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
| ArrayList<String> of file name | Storage Device (handled by OS) |
| XML files | Storage Device (handled by OS) |

Table 4‑34 : Command Structure External Data Dependencies

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| PrinterConfiguration Object | Persistence Framework |
| ExtruderConfiguration Object | Persistence Framework |
| MaterialConfiguration Object | Persistence Framework |
| PrintJobConfiguration Object | Persistence Framework |
| PrintConfiguration Object | Persistence Framework |

Table 4‑35 : Command Structure Internal Data Descriptors

#### Process

Get Command

execute(){

Open file for reading

Create new Object

Use Jaxb to load object with data from xml

}

Save Command

execute(){

Open file for writing

Use Jaxb to write the xml file to the correct location

}

# Preprocessing Layer



Figure 5‑1: Preprocessing Layer UML

The Preprocessing Layer provides an abstract interface between the User Interface and the Processing Layer. This layer‘s purpose is to translate and repackage the print request object in to the format that the processing layer needs. The Preprocessing Layer starts by receiving all of the necessary configuration and object data from the User Interface’s Controller Subsystem then converts it into a unified format that the Processing Layer understands. In the current iteration of this project, the Preprocessing Layer only has one subsystem, the Normalization Subsystem and will be converting STL files it receives from the object sent to it into an AMF file. The Preprocessing Layer then packages the configuration, object definition, and material data into the correct format for the Processing Layer.

## Normalization Subsystem Modules



Figure 5‑2 : Normalization Subsystem UML

The Normalization Subsystem is responsible for receiving the bundle containing the object data and printer configurations from the User Interface’s Controller Subsystem and converting the object data into a format the Processing Layer can understand. After the object file(s) is converted, this subsystem modifies the bundle of the object file information and the printer configuration for the Processing Layer so it can calculate a printing path.

### Object Subsection Module

#### Prologue

The Object Subsection Module splits objects into multiple subsections about the z axis and produces a new STL file to represent each subsection.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| createSubsections | Print Job Configuration Object | Modified Print Job Configuration Object |

Table 5‑1 : Object Subsection Module Interfaces

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
| Subsectioned STL files | OpenScad Executable |

Table 5‑2 : Object Subsection Module External Data Dependencies

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| PrintJobConfiguration object | PrintJobController |

Table 5‑3 : Object Subsection Module Internal Data Descriptors

Data Required from Print Job Configuration

* Parent STL Files in each subsection
* Bottom Z for each subsection
* Top Z for each subsection

Data Modified in Print Job Configuration

* Subsection STL Files for each subsection

#### Process

subsectionConfigurationList = printJobConfiguration.getSubsetConfigurationList()

for each subsectionConfiguration in subsectionConfigurationList:

stlFile = subsectionConfiguration.getParentStlFile()

bottomZ = subsectionConfiguration.getBottomZ()

topZ = subsectionConfiguration.getTopZ()

subsectionStlFile = createSubsection(stlFile, bottomZ, topZ)

subsectionConfiguration.setStlFile(subsectionStlFile)

### File Translation Module

#### Prologue

The File Translation Module translates multiple STL files, each describing a part of the object that is a different material, into a single AMF file. The produced AMF file describes the object as a whole while also containing information mapping parts of the objects to the material that the part consists of.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| translateFiles | Print Job Configuration Object | Modified Print Job Configuration Object |

Table 5‑4 : File Translation Module Interfaces

#### External Data Dependencies

None

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| PrintJobConfiguration object | PrintJobController |

Table 5‑5 : File Translation Module Internal Data Descriptors

Data Required from Print Job Configuration

* Subsection STL Files for each subsection
* Materials of each STL File for each subsection

Data Modified in Print Job Configuration

* AMF File for each subsection

#### Process

create new materialFile

create new vertexFile

sort file configuration list by extruder number

for each file configuration:

write material definitions to materialFile

for each file configuration:

create new volumeFile

add material attribute to volumeFile

stlFile = get subset stl file from file configuration

for each face in stlFile:

for each vertex in face:

write vertex to amf file

write face to volume file

amfFile = concatenate(materialFile, vertexFile, volumeFile)

store AMF file in SubsetConfiguration

# Processing Layer



Figure 6‑1 : Processing Layer UML

The Processing Layer takes the formatted package and translates it into G-codes. The slicing subsystem is the only subsystem in this layer. The slicing subsystem needs to be able to be replaced based on the growing needs of future development. The Processing Layer is designed to be expandable. The initial implementation will use Slic3r as its engine, but the normalized data from the Preprocessing layer will not need to change if the slicing implementation changes. This allows the slicing engine to be easily replaced with a different implementation and provides the future possibility of allowing the user to select which slicing engine they wish to use in any given print. Once the Processing Layer processes the normalized data from the Preprocessing Layer, the Processing Layer modifies the PringJobConfiguration object to reference the G-Codes produced and printer configuration information for the Post-Processing Layer.

## Slicing Subsystem Modules



Figure 6‑2 : Slicing Subsystem UML

The responsibility of the Slicing Subsystem is to read the object file and divide the geometry of the object into appropriate layers. Then for each material object of each layer, the subsystem will draw out a printing path for the head to follow. Once the object has been drawn out, the printing instructions will be converted to G-Codes and passed to the Post-Processing Layer for final changes to the instructions.

### Slicing Engine Wrapper

#### Prologue

The Slicing Engine Wrapper is responsible for wrapping an abstract API around the slicing engine of the system. The wrapper takes a normalized Print Job Configuration Object, writes the properties from that object to a configuration file readable by the slicing engine, then uses the configuration file as a parameter to generate G-Codes for each subsection found in the Print Job Configuration Object. The G-Code for each subsection is then placed into each respective Subsection Configuration object in the Print Job Configuration object so that it can later be accessed by the Post Processing Layer.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| GenerateGCode | Print Job Configuration Object | Modified Print Job Configuration Object |

Table 6‑1 : Slicing Engine Wrapper Interfaces

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
| G-Code files | Slic3r Slicing Engine |

Table 6‑2 : Slicing Engine Wrapper External Data Descriptors

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| PrintJobConfiguration object | PrintJobController |

Table 6‑3 : Slicing Engine Wrapper Internal Data Descriptors

Required Data from Print Job Configuration

* For each subsection, all data elements from the following classes:
  + InfillConfiguration
  + LayerAndPerimeterConfiguration
  + SpeedConfiguration
  + SkirtAndBrimConfiguration
  + SupportMaterialConfiguration
* For each file in each subsection, all data elements from the following classes:
  + MaterialConfiguration
  + ExtruderConfiguration
* The following data elements from the PrinterConfiguration class:
  + bedX
  + bedY
  + printCenterX
  + printCenterY
  + zOffset
  + gCodeFlavor
  + useRelativeEDistances
  + vibrationLimit

Data Modified in Print Job Configuration

* G-Code files for each subsection

#### Process

configFile = new File().openForWriting()

printerConfiguration = PrintJobConfiguration.getPrinterConfiguration()

for each property in printerConfiguration:

write property to configFile

subsectionConfigurationList = printJobConfiguration.getSubsetConfigurationList()

zOffset = 0

for each subsectionConfiguration in subset:

write zOffset to configFile

printConfiguration = subsetConfiguration.getPrintConfiguration()

for each property in printConfiguration:

write property to configFile

amfFile = subsetConfiguration.getAMF()

gCode = run slicing engine with amfFile and configFile as parameters

subsectionConfiguration.setGCode(gCode)

zOffset += subsetConfiguration.getEndZ()

# Post processing Layer



Figure 7‑1: Post Processing Layer UML

The Post Processing Layer receives the G-Codes from the Processing Layer and modifies the instructions to match special considerations the printer may need. G-Codes are somewhat standard, but some printers have instructions that are specific to that printer only. It is the job of the Post Processing Layer to correct the G-Code received from the Processing Layer to accommodate for those special instructions. Once finished, the Post Processing Layer will output G-Codes and the configuration to the Print Controller Layer. Similar to the Preprocessing and Processing Layers, the Post-Processing layer is designed to be easily expanded in future iterations. Initially, this layer will only support the printer supplied by the Mechanical Engineering Team.

## G-Code Preparation Subsystem Modules



Figure 7‑2 : G-Code Preparation Subsystem UML

The responsibility of the G-Code Preparation Subsystem is to modify the G-Code produced by the Processing Layer to be 100% compatible with the printer being used. This subsystem therefore must accommodate for any special instructions that are unique to that printer.

### Parser Module

#### Prologue

The Parser Module is responsible for parsing G-Code data it receives and modifying G-Codes to ensure that they are compliant with the G-Code standard required by the printing hardware and removing any unnecessary G-Codes.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| parseAndModifyGCode | Print Job Configuration Object | Modified Print Job Configuration Object |

Table 7‑1 : Parser Module Interfaces

#### External Data Dependencies

None

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| PrintJobConfiguration object | PrintJobController |

Table 7‑2 : Parser Module Internal Data Descriptors

Required Data from Print Job Configuration

* G-Code files for each subsection
* Printer G-Code flavor
* Custom print start/end G-Code

Data Modified in Print Job Configuration

* G-Code files for each subsection

#### Process

tempFile = new File for writing

for each line in gCodeFile

tokens = line.tokenize()

for each token in tokens:

if(isValidToken(token)):

tempLine += token

else

continue

write tempLine to tempFile

### Unification Module

#### Prologue

The Unification Module is responsible for combining separated G-Codes into a single string of G-Codes that can be streamed to the printer. When parts of an object are sliced separately, the Unification Module must be used to combine the separated G-Codes together.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| unifyGCode | Print Job Configuration Object | Modified Print Job Configuration Object |

Table 7‑3 : Unification Module Interfaces

#### External Data Dependencies

None

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| PrintJobConfiguration object | PrintJobController |

Table 7‑4 : Unification Module Internal Data Descriptors

Required Data from Print Job Configuration

* G-Code files for each subsection

Data Modified in Print Job Configuration

* Finalized Print Job G-Code file

#### Process

subsetConfigurationList = printJobConfiguration.getSubsetConfigurationList()

unifiedGCodeFile = new File().openForWriting()

for each subsectionConfiguration in subsectionConfigurationList:

gCode = subsectionConfiguration.getGCode()

write gCode to unifiedGCodeFile

printJobConfiguration.setFinalizedGCode(unifiedGCodeFile)

# Printer control layer

The printer control layer receives inbound G-Codes from the post processing layer. The printer control layer will then assemble the data received in preparation to be sent to the communications layer. The G-Codes may be modified contingent on data received from the printer feedback and user interface layers. If either the printer feedback or user interface layers indicate that the print must be stopped then the printer control layer must insert halt commands into the G-Code command stream such that the machine terminates the print in a timely manner.

## Printer State Controller Subsystem Modules

The purpose of the Printer State Controller Modules are to receive incoming G-Codes from the Post Processing Layer, insert safety G-Codes if necessary based on input from the User Interface Layer and the Printer Feedback Layer, and buffer the G-Codes into an appropriately sized buffer to be sent to the Communications Layer.

### Printer State Control Module

#### Prologue

The printer state control module will receive a PrintJobConfiguration object containing all G-Codes necessary to print the object(s). The printer state control module will then begin to buffer the data such that it can be presented to the communications layer to be transferred to the printer hardware. During this process, the printer state control module will consume operating information from the printer feedback layer and user interface layer to conditionally insert safety G-Codes into the buffer. These safety G-Codes may include halt commands, return to home position, reduce extruder temperature, or other commands necessary to keep the print head within operating parameters.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| runPrintJob | Print Job Configuration Object | None |
| pauseResumePrintJob | Change flag status | Boolean success state |
| cancelPrintJob | None | Boolean success state |
| updateStatus | PrinterStatus Obect | None |

Table 8‑1 : Printer State Control Module Interfaces

#### External Data Dependencies

None

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| PrintJobConfiguration object | PrintJobController |
| PrinterStatus object | DispatchModule |

Table 8‑2 : Printer State Control Module Internal Data Descriptors

#### Process

// Get finalized G-Code buffer //

finalizedGcodeBuffer = printJobConfiguration.GetFinalizedGcode()

// Get the current Printer Feedback status //

printerFeedbackStatus = g\_pDispatch->GetCurrentStatus()

// Get outbound buffer ready to go //

outboundBuffer = ArrayList<String>

if(printJobConfiguration.ACK)

if(finalizedGcodeBuffer.length >= 1)

{

if(OutOfBounds(printerFeedbackStatus))

outboundBuffer.append(STOP)

else

{

// Add a single G-Code to the buffer //

outboundBuffer.append(finalizedGcodeBuffer.head, 1)

finalizedGcodeBuffer.remove(last)

}

}

else

while(finalizedGcodeBuffer.length >= 1)

if(!OutOfBounds(printerFeedbackStatus))

{

if(finalizedGcodeBuffer.length < 5)

{

// If the length is less than 5 deep, just append the rest //

outboundBuffer.append(finalizedGcodeBuffer.head, finalizedGcodeBuffer.length)

finalizedGcodeBuffer.remove(length)

}

else

{

outboundBuffer.append(finalizedGcodeBuffer.head, 5)

finalizedGcodeBuffer.remove(last 5)

}

}

else

outboundBuffer.append(STOP)

// Wait for buffer to become available for writing //

while(g\_pSerialization->IsBufferFull()) {}

// Buffer is ready: lock, copy, unlock fast

g\_pSerialization->LockBuffer()

g\_pSerialization->CopyBuffer(outboundBuffer)

g\_pSerialization->UnlockBuffer()

# Printer Feedback Layer

The printer feedback layer is responsible for receiving buffered input from the printer hardware, interpreting the data, then formatting and dispatching the information to the print controller layer and user interface layers. The information received from the printer will mainly consist of printer state such as extruder temperature, arm position, and other operating parameters. The printer feedback layer will transform the data received from the printer into useful and readable data that can be passed to the user interface and printer controller layers.

## State Monitoring Subsystem Modules

The State Monitoring Modules receive buffered input from the printer hardware. The input received will be in a format that is unreadable to the other layers. These modules interpret this information, such as temperature and arm position, and dispatch them to the User Interface Layer and the Printer Control Layer.

### Dispatch Module

#### Prologue

The dispatch module will poll the receive buffer continually at a specified time interval. The receive buffer will be filled by the printer hardware at a particular rate specified by the operator. The information contained in the buffer will be the data required to populate a PrinterStatus object. The incoming data will be packaged by the dispatch module into a PrinterStatus object that can be read by user interface layers and the printer control layers.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| startPolling | PrinterStatus Object | PrinterStatus object updates |

Table 9‑1 : Dispatch Module Interfaces

#### External Data Dependencies

None

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| PrinterStatus object | Deserialization Module |

Table 9‑2 : Dispatch Module Internal Data Descriptors

#### Process

// Wait until buffer is full //

while(!g\_pDeserialization->IsBufferFull()){}

// Buffer is full //

buffer = new DeSerializedClass

g\_pDeserialization->LockBuffer()

g\_pDeserialization->CopyBuffer(buffer)

g\_pDeserialization->UnlockBuffer()

PrinterStatusObject\* ps = GetPrinterStatusObject()

// Enforce thread safety //

ps->Lock()

ps->Populate(buffer)

ps->Unlock()

# Communications Layer

The purpose of the communications layer is to serve as an interface between the client software (The 3-D Printer Fabrication System) and the printer firmware itself. The communications layer shall be able to implement most RS-232 compliant serial communications medium such that data can be transferred from the client software to the printer firmware reliably. The communications layer serves both the printer feedback and printer control layers. The printer control layer represents outbound communication (from client software to printer) while the printer feedback layer represents inbound communication (from printer to client software).

## Communications Subsystem Modules

The purposes of the communication subsystem modules are to initialize a two way communication with the printer hardware itself, serialize outbound data from the printer control layer, and de-serialize data inbound from the printer hardware. Serialization in this sense means transforming a large data structure or buffer into a serial stream such that it can be transmitted via a serial protocol. De-serialization implies the reverse method where a serial stream is assembled into a data structure and given a context that is meaningful to the upper layers.

### RX/TX Module

#### Prologue

The Receive/Transmit (RX/TX) module is responsible for first establishing a connection to the printer hardware. All information necessary to establish this connection such as port, baud rate, ACK, etc., are passed to the RX/TX module from PrintJobConfiguration object. Once a connection is established, the RX/TX module facilitates a receive ring buffer for data coming from the printer. The module will facilitate methods that allow the dispatch module in the printer feedback layer to query the buffer and consume the data so that it can be formatted and sent to the appropriate modules.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| transmit | Serialized Data | ACK details |
| receive | Buffer | Serialized Data |
| initializeConnection | PrintJobConfiguration object | Boolean success state |

Table 10‑1 : Rx/Tx Module Interfaces

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
| Buffered Serial Data | Printer |

Table 10‑2 : Rx/Tx External Module Data Dependencies

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| Serialized G-Code Buffer | Serialization Module |
| PrintJobConfiguration object | PrintJobController Module |

Table 10‑3 : Rx/Tx Module Internal Data Descriptors

#### Process

// Initialize connection to printer if not initialized //

if(!connectionInitialized)

initializeConnection(PrintJobConfiguration)

// Wait until buffer is ready to be written to //

outboundHandle = CreateThread(outboundCallback)

OUTBOUND CALLBACK

while(!comPort->IsBufferFull()) {}

// Printer buffer is ready //

// Locks and unlocks automatically...enforced by driver //

comPort->WriteToBuffer(m\_outboundGcodes)

receiveHandle = CreateThread(receiveCallback)

RECEIVE CALLBACK

// Leaves port open for reading from printer //

comPort->ReadFromBuffer(m\_printerFeedback)

### Serialization Module

#### Prologue

The purpose of the serialization module is to receive the individual G-Codes that will carry out the print run and serialize them in preparation to be sent to the RX/TX module. Serialization is the process of breaking down a large data structure into a serial stream to be transmitted and reconstructed. One of the facilities provided by the RX/TX module is a ring buffer mechanism. The buffer is 5 G-Codes deep, and 128 bytes wide. If the user chooses to enforce “ACK”, then the buffer becomes 1 G-Code deep. Therefore, it is the serialization module’s responsibility to break the G-Codes down to byte streams according to the buffer configuration and pack the buffer.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| serializeData | G-Code | None |

Table 10‑4 : Serialization Module Interfaces

#### External Data Dependencies

None

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| G-Code Object | Printer State Controller |

Table 10‑5 : Serialization Module Internal Data Descriptors

#### Process

// Continually push G-Codes to RX/TX

while(Outbound GCodeBuffer is full)

{

OutboundGcodeBuffer.Serialize()

// Wait for RX/TX module buffer to be ready to write //

while(g\_pRXTX->IsBufferFull()) {}

g\_pRXTX->LockBuffer()

g\_pRXTX->CopyBuffer(OutboundGCodeBuffer)

g\_pRXTX->UnlockBuffer()

}

### De-Serialization Module

#### Prologue

The purpose of the de-serialization module is to do the reverse of the serialization module for data returning from the printer hardware. The printer will send information regarding its current operating state back to the printer feedback layer via the de-serialization module. Since this data is going to be a byte stream, it must be re-assembled into a data structure that is recognizable and readable by the other modules. This re-assembly is what the de-serialization module is tasked with.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| deserializeData | Serialized Data | ArrayList<Object> |

Table 10‑6 : De-Serialization Module Interfaces

#### External Data Dependencies

None

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| Printer Status Buffer | RxTx Module |

Table 10‑7 : De-Serialization Module Internal Data Descriptors

#### Process

// Wait for data in RX TX buffer //

while(!g\_pRXTX->IsBufferFull()) {}

// Buffer is ready //

g\_pRXTX->LockBuffer()

g\_pRXTX->CopyBuffer(PrinterStatus)

g\_pRXTX->UnlockBuffer()

// DeSerialize Printer Status Data

PrinterStatus.Deserialize(SerializationClass)

# Quality Assurance

This section details the plan the team will follow to ensure quality.

## Unit Testing

### User Interface Layer

|  |  |  |
| --- | --- | --- |
| Subsystem | Module | Test |
| GUI | Import GUI | Given an STL file, the Import GUI Module will verify that the STL file is imported when the import button is pressed and deleted from the system when the delete button is pressed. For a file of an incorrect type, the Import GUI Module should notify the user that the file is not accepted when the import button is pressed. |
| Material GUI | Given a set of inputs that the user types in, the Material GUI Module will verify that each input is of the correct type and will not pass the input to the Material Controller if the input type does not match the type required. (eg. The user types in a string when it needs an int) |
| Printer Config GUI | Given a set of inputs that the user types in, the Printer Config GUI Module will verify that each input is of the correct type and will not pass the input to the Printer Config Controller if the input type does not match the type required. (eg. The user types in a string when it needs an int) |
| Extruder Config GUI | Given a set of inputs that the user types in, the Extruder Config GUI Module will verify that each input is of the correct type and will not pass the input to the Extruder Config Controller if the input type does not match the type required. (eg. The user types in a string when it needs an int) |
| Print Config GUI | Given a set of inputs that the user types in, the Print Config GUI Module will verify that each input is of the correct type and will not pass the input to the Print Config Controller if the input type does not match the type required. (eg. The user types in a string when it needs an int) |
| Print Job GUI | Given a set of inputs that the user types in, the Print Job GUI Module will verify that each input is of the correct type and will not pass the input to the Print Job Controller if the input type does not match the type required. (eg. The user types in a string when it needs an int) |
| Status GUI | Given a set of printer status updates, the Status GUI Module will display the information correctly in the Status window. When the pause/resume button is pressed, the Status GUI Module will call the appropriate function from the Status Controller correctly and the word “pause” will change to “resume” or vice-versa. |
| Controller | Import Controller | Given an STL file, the Import Controller Module will call the Persistence Framework to import the file into the system and return a Boolean value of 1 for the successful operation. For file of the incorrect type, the Import Controller Module will not call the Persistence Framework and return a Boolean value of 0. |
| Material Controller | Given a set of values passed in by the GUI, the Material Controller Module will verify that value does not break the boundary set for that value. If one or more values do not fit in the boundaries provided, a Boolean value of 0 will return. If all values fall into their appropriate boundaries, the Material Controller will call the Persistence Framework to store the values and a Boolean value of 1 will be returned. |
| Printer Config Controller | Given a set of values passed in by the GUI, the Printer Config Controller Module will verify that value does not break the boundary set for that value. If one or more values do not fit in the boundaries provided, a Boolean value of 0 will return. If all values fall into their appropriate boundaries, the Printer Config Controller will call the Persistence Framework to store the values and a Boolean value of 1 will be returned. |
| Extruder Config Controller | Given a set of values passed in by the GUI, the Extruder Config Controller Module will verify that value does not break the boundary set for that value. If one or more values do not fit in the boundaries provided, a Boolean value of 0 will return. If all values fall into their appropriate boundaries, the Extruder Config Controller will call the Persistence Framework to store the values and a Boolean value of 1 will be returned. |
| Print Config Controller | Given a set of values passed in by the GUI, the Print Config Controller Module will verify that value does not break the boundary set for that value. If one or more values do not fit in the boundaries provided, a Boolean value of 0 will return. If all values fall into their appropriate boundaries, the Print Config Controller will call the Persistence Framework to store the values and a Boolean value of 1 will be returned. |
| Print Job Controller | Given a set of values passed in by the GUI, the Print Job Controller Module will verify that value does not break the boundary set for that value. If one or more values do not fit in the boundaries provided, a Boolean value of 0 will return. If all values fall into their appropriate boundaries, the Print Job Controller will package the information into a new Configuration Object. |
| Status Controller | When given a request to update, the Status Controller will collect the most recent Printer Status and return it in an array format. When given a request to pause, resume, or cancel, the Status Controller will pass the command to the Printer Control Layer. |
| Database | Persistence Framework | Given a String and get command or object and save command the Persistence Framework will retrieve the requested object or store the object based on the call given. |
| Command Structure | Each command will be exercised with an object to save, a type to retrieve, a string to delete, a type to get a list. The saves will require XML inspection to ensure proper storage, retrievals will require object inspection, deletes will be confirmed on the drive, and lists will be checked against the folder contents. |

Table 11‑1 : User Interface Layer Unit Tests

### Preprocessing Layer

|  |  |  |
| --- | --- | --- |
| Subsystem | Module | Test |
| Normalization | Subsection | Given a Print Job Configuration Object containing the bottom z, top z, and Parent STL files for each subsection, the Subsection Module will create new Subsection STL files (sub-sectioned by z = bottom z and z = top z planes) for each Parent STL file and store the reference to the created Subsection STL files back into the Print Job Configuration Object. |
| File Translation | Given the Print Job Configuration Object containing the Subsection STL files for each subsection as well as the materials for each STL file, the File Translation Module will translate these files into a correct AMF File for each subsection. |

Table 11‑2 : Preprocessing Layer Unit Tests

### Processing Layer

|  |  |  |
| --- | --- | --- |
| Subsystem | Module | Test |
| Slicing Engine | Slicing Engine Wrapper | Given a Print Job Configuration Object with all required data elements (as described in Section 6.1.1.4) the Slicing Engine Wrapper will run the Slic3r slicing engine to produce a G-Code file for each subsection and place a reference to each G-Code file into the Print Job Configuration Object. |

Table 11‑3 : Processing L ayer Unit Tests

### Post Processing Layer

|  |  |  |
| --- | --- | --- |
| Subsystem | Module | Test |
| G-Code Preparation | Parser | Given a Print Job Configuration Object with all required data elements (as described in Section 7.1.1.4) the Parser Module will modify or delete any unacceptable G-Codes found any G-Code files. |
| Unification | Given a Print Job Configuration Object with all required data elements (as described in Section 7.1.2.4) the Unification Module will concatenate all Subsection G-Code files into a single Finalized G-Code file and place a reference to the finalized G-Code file into the Print Job Configuration Object. |

Table 11‑4 : Post Processing Layer Unit Tests

### Printer Control Layer

|  |  |  |
| --- | --- | --- |
| Subsystem | Module | Test |
| Printer State Controller | Printer State Control | Given a set of G-Codes, out-of-range test feedback, and pause/resume/cancel commands from the user, the Printer State Control Module will insert appropriate G-Codes to halt the print process into the output G-Code buffer. |

Table 11‑5Printer Control Layer Unit Tests

### Printer Feedback Layer

|  |  |  |
| --- | --- | --- |
| Subsystem | Module | Test |
| State Monitoring | Dispatch Module | Given a test buffer that is filled with de-serialized feedback data, the Dispatch Module shall assemble the data into a PrinterStatus Object and transmit the object to the User Interface and Printer Control Layers. |

Table 11‑6 : Printer Feedback Layer Unit Tests

### Communication Layer

|  |  |  |
| --- | --- | --- |
| Subsystem | Module | Test |
| Communications | RX/TX | Given a PrintJobConfiguration object, will validate that the selected serial port is valid and enumerated. The RX/TX module will establish a connection and verify that it is established by querying that the remote buffer is available. The RX/TX module will then send a buffer of 5 G-Codes to the printer and verify that the printer received them. The RX/TX module will also send 1 G-Code and verify that an appropriate ACK is received. |
| Serialization | Given a set of G-Codes, the Serialization module will serialize the data according to the ISerializable class definition. The Serialization Module will then pack the RX/TX ring buffer with the serialized commands. |
| De-Serialization | Given a stream of serialized feedback, the De-Serialization Module will de-serialize the stream according to the ISerializable class definition. The De-Serialization module will then raise a buffer full flag to indicate to the Dispatch Module that feedback data is ready for consumption. |

Table 11‑7 : Communications Layer Unit Tests

## Component Testing

This section describes the approach that will be taken to perform component testing. Each layer of the system is considered a separate component and will be tested separately.

### User Interface Layer

The User Interface Layer will be tested to ensure it performs correctly as a component. To test this layer user input will be entered into each GUI module and saved then reloaded. This will check for correct creation storage and retrieval. A print Job will then be created. This Print job object will be inspected to ensure all data was collected and compiled correctly.

### Preprocessing Layer

The Preprocessing Layer will be tested to ensure that it performs correctly as a component. To test this layer, a Print Job Configuration object containing all the necessary data for preprocessing will be passed in for preprocessing. After preprocessing takes place, the Print Job Configuration object will be examined to ensure that subsections were created correctly and STL files were combined into AMF files correctly.

### Processing Layer

The Processing Layer will be tested to ensure that it performs correctly as a component. To test this layer, a Print Job Configuration object containing all the necessary data for processing layer will be passed in for processing. After the data is processed, the Print Job Configuration object will be examined to ensure that G-Codes were created for each subsection. These G-Codes will then be examined in Repetier Host using the G-Code visualizer function to ensure that correct paths were created.

### Post Processing Layer

The Processing Layer will be tested to ensure that it performs correctly as a component. To test this layer, a Print Job Configuration object containing all the necessary data for post processing will be passed in for post processing. After post processing takes place, the Print Job Configuration object will be examined to ensure G-Codes have been correctly parsed, modified, and combined into a final G-Code file. These G-Codes will then be examined using Repetier Host and printed using Repetier Host as the communication link to the 3-D printer.

### Printer Control Layer

The Printer Control Layer will be tested to ensure that it performs correctly as a component. To test the Printer Control Layer, a Print Job Configuration object containing the G-Codes necessary to complete a print job will be passed to the Print State Control Module. When given random pause/resume/cancel commands, the resulting G-Codes will be examined to ensure that the appropriate G-Codes have been added to the G-Code buffer to reflect these state changes. The modified G-Codes can be compared against the original G-Codes to ensure that they have been inserted correctly into outbound the G-Code buffer.

### Printer Feedback Layer

The Printer Feedback Layer will be tested to ensure that it performs correctly as a component. To test the Printer Feedback Layer, de-serialized printer feedback will be provided to the Dispatch Module. The feedback will consist of out-of-bounds operating parameters and within bounds operating parameters to ensure that the Printer Control Layer and User Interface Layer are notified appropriately of the status. We shall also verify that given de-serialized printer feedback that the layer properly assembles and dispatches a PrinterStatus object loaded with values that are in range and of the proper type.

### Communications Layer

The Communications Layer will be tested to ensure that it performs correctly as a component. To test the Communications Layer, printer connection parameters (defined in Print Job Configuration object) will be passed to the RX/TX module. We will verify that the layer has made a connection and that it recognizes the printer device and its ring buffer. We will send simple test G-Codes to verify that the printer firmware has received the data and can execute the instructions. We will also verify that the De-Serialization module correctly acknowledges and allows its buffer to be filled by the printer firmware with printer feedback information such as extruder temperature and arm position.

## Integration Testing

Integration testing will be performed by creating a complete print job and initiating the print process. The testing will consist of supplying print parameters to the User Interface Layer and starting the print job. The data supplied will then be used by the other layers to create G-Codes and send G-Codes to the printer for printing. The final result will be that the printer produces the correct object.

## System Verification Testing

These tests will be used for verification testing of the 3-D Printer Fabrication System project. The following are critical points in the functional success of the 3-D Printer Fabrication System project.

### Verify that the system reads STL files

**9.1.1 Requirement(s) addressed:** 3.1 – STL File Input

**9.1.2 Verification Procedure:** The user will be able to see the file has been accepted and the file name will be displayed in the GUI.

### Verify the database interface

**9.2.1 Requirement(s) addressed:** 3.2 – Graphical User Interface, 3.13 – Database Interface, 3.14 – Store and Load Material Records, 8.1 – Material Database

**9.2.2 Verification Procedure:** The user will load the GUI and click on the view/edit database button. The user then will see and be able to edit stored values. Upon changing values the user will return to the main menu. Clicking on the view/edit button again will display the edited values.

### Verify the system prints a model

**9.3.1 Requirement(s) addressed:** 3.1 – STL File Input, 3.3 – Generate Machine Instructions, 3.4 – Issue Machine Instructions, 3.6 – Monitor Temperature, 3.7 – Monitor Position, 3.8 – Identify Material Constraints, 3.9 – Identify Materials, 3.10 – Identify Shapes, 3.11 – Determine Shape of Support Material Structure, 3.12 – Create Printing Path, 3.14 – Store and Load Material Records, 3.15 – Slice Geometry into Thickness Levels, 3.16 – Monitor Flow Sensors, 4.2 – Host Software to Printer Connection

**9.3.2 Verification Procedure:** The user will load an STL file and click print. The system will then print the correct shape and material.

### Verify the system stops printing of out of operational range

**9.4.1 Requirement(s) addressed:** 6.1 – Temperature cutoff threshold, 3.16 – Monitor Flow Sensors

**9.4.2 Verification Procedure:** During a print run a fan will be pointed at the head reducing its temperature to below specified material requirements and the printer will stop printing until the temperature is raised to the correct level.

# Requirements Traceability Matrix

The tables below detail the mapping of requirements to individual modules in the system. The matrix also shows the size and complexity of the different modules. If there The User Interface layer is very complex and provides the features needed for many of the requirements. The matrices also demonstrate that all requirements are being handled by at least one module.

Coupling between modules has been greatly limited. There is a major pitfall in the overall design in that the data objects enforce a high degree of coupling between the GUI and controller modules. These data elements also tie directly to the processing layers so that if a new slicing engine uses different data points the data objects, GUIs, and controllers would need to be modified to accommodate the change. The concept of modularity and configurability are opposed in this setup. Considering the complexity of the data objects there should not be much need for a new slicing engine to require more information.

Each module is designed to be independent of the rest of the system. One side effect of this is that all the data objects are passed from one module to the next. If there is a change in the data needs of a module only the GUI/Controller and the module that needs the new information will need to change. The complex data object provides independence in that each module only knows what it needs and does not care what extra information is stored in the objects.

The processing modules are the most critical to the requirements of the system. Each of the processing modules does a transformation of the object representation. These transformations are the primary need of the sponsor. The next most significant need is the GUI allowing interaction with the database while providing an interface to the complex processing modules.

## User Interface Layer

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Number | Title | Interface Layer | Import GUI | Printer Configuration GUI | Material GUI | Print Configuration GUI | Print Job GUI | Status GUI | Extruder Configuration GUI | Import Controller | Printer Configuration Controller | Material Controller | Print Configuration Controller | Print Job Controller | Extruder Configuration Controller | Status Controller | Persistence Framework | Command Structure |
| 3.1 | STL File Input | X | X |  |  |  |  |  |  | X |  |  |  |  |  |  |  |  |
| 3.2 | Graphical User Interface (GUI) | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 3.3 | Generate Machine Instructions |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3.4 | Issue Machine Instructions |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3.5 | Monitor Temperature | X |  |  |  |  |  | X |  |  |  |  |  |  |  | X |  |  |
| 3.6 | Monitor Position |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3.7 | Adhere to Material Constraints | X |  |  | X | X | X |  | X |  | X | X | X |  |  |  |  |  |
| 3.8 | Identify Materials | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3.9 | Identify Shapes |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3.1 | Determine Shape of Material Support Structure |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3.11 | Create Printing Path |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3.12 | Database Interface | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X |
| 3.13 | Load & Store Materials | X |  |  | X |  |  |  |  |  |  | X |  |  |  |  | X | X |
| 3.14 | Slice Geometry into Thickness Levels |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3.15 | Monitor Flow Sensors |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3.17 | Allow for UV Head Polymerization | X |  |  | X |  |  |  |  |  |  | X |  |  |  |  |  |  |
| 8.1 | Material Database | X |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X | X |
| 8.2 | Abstract Hardware Interface | X |  | X |  |  | X |  |  |  | X |  |  | X |  |  |  |  |
| 8.3 | Modular and Scalable Design | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

Table 12‑1 : User Interface Layer Requirement Traceability Matrix

## Processing Layers

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number | Title | Preprocessing Layer Object Subsection | Preprocessing Layer Object File Translation | Processing Layer Slicing Engine Wrapper | Post Processing Layer Parser | Post Processing Layer Unification |
| 3.1 | STL File Input |  |  |  |  |  |
| 3.2 | Graphical User Interface (GUI) |  |  |  |  |  |
| 3.3 | Generate Machine Instructions |  |  | X | X | X |
| 3.4 | Issue Machine Instructions |  |  |  |  |  |
| 3.5 | Monitor Temperature |  |  |  |  |  |
| 3.6 | Monitor Position |  |  |  |  |  |
| 3.7 | Adhere to Material Constraints |  |  | X | X | X |
| 3.8 | Identify Materials | X | X | X | X | X |
| 3.9 | Identify Shapes |  |  | X |  |  |
| 3.1 | Determine Shape of Material Support Structure |  |  | X |  |  |
| 3.11 | Create Printing Path |  |  | X | X | X |
| 3.12 | Database Interface |  |  |  |  |  |
| 3.13 | Load & Store Materials |  |  |  |  |  |
| 3.14 | Slice Geometry into Thickness Levels |  |  | X |  |  |
| 3.15 | Monitor Flow Sensors |  |  |  |  |  |
| 3.17 | Allow for UV Head Polymerization |  |  | X | X | X |
| 8.1 | Material Database |  |  |  |  |  |
| 8.2 | Abstract Hardware Interface |  |  |  |  |  |
| 8.3 | Modular and Scalable Design | X | X | X | X | X |

Table 12‑2 : Processing Layers Requirement Traceability Matrix

## Printer State, Communications, and Printer Feedback Layers

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Number | Title | Printer Control Layer Printer State Controller | Communications Serialization | Communications RxTx | Communications Deserialization | Printer | Printer Feedback Layer  Dispatch |
| 3.1 | STL File Input |  |  |  |  |  |  |
| 3.2 | Graphical User Interface (GUI) |  |  |  |  |  |  |
| 3.3 | Generate Machine Instructions | X |  |  |  |  |  |
| 3.4 | Issue Machine Instructions |  | X | X | X |  |  |
| 3.5 | Monitor Temperature | X | X | X | X | X | X |
| 3.6 | Monitor Position | X | X | X | X | X | X |
| 3.7 | Adhere to Material Constraints | X | X | X | X | X | X |
| 3.8 | Identify Materials |  |  |  |  |  |  |
| 3.9 | Identify Shapes |  |  |  |  |  |  |
| 3.1 | Determine Shape of Material Support Structure |  |  |  |  |  |  |
| 3.11 | Create Printing Path |  |  |  |  |  |  |
| 3.12 | Database Interface |  |  |  |  |  |  |
| 3.13 | Load & Store Materials |  |  |  |  |  |  |
| 3.14 | Slice Geometry into Thickness Levels |  |  |  |  |  |  |
| 3.15 | Monitor Flow Sensors | X |  | X | X | X | X |
| 3.17 | Allow for UV Head Polymerization | X |  |  |  | X |  |
| 8.1 | Material Database |  |  |  |  |  |  |
| 8.2 | Abstract Hardware Interface | X |  |  |  | X | X |
| 8.3 | Modular and Scalable Design | X | X | X | X | X | X |

Table 12‑3 : Printer State, Communications, and Printer Feedback Layers Requirement Traceability Matrix

# Acceptance Plan

This section details the plan to meet the acceptance criteria for the 3-D Printer Fabrication System. Including the final packaging and installation needs, the acceptance testing plan, as well as the criteria the product must meet to be accepted by the customer.

## Package and Installation

The packaging for the 3-D Printer Fabrication System will consist of a USB flash drive containing the executable installer and a user manual. These items will be delivered in a small box for easy transportation.

## Acceptance Testing

Acceptance tests will be performed to ensure all criteria are meet to a reasonable standard. These tests will be fully detailed in the test plan document.

## Acceptance Criteria

The following are the required criteria for the system.

* STL File Input
* Graphical User Interface
* Generate Machine Instructions
* Issue Machine Instructions
* Monitor Temperature
* Monitor Position
* Adhere to Material Constraints
* Identify Materials
* Identify Shapes
* Determine Shape of Material Support Structure
* Create Printing Path
* Database Interface
* Load and Store Materials
* Slice Geometry into Thickness Levels
* Monitor Flow Sensors
* Allow for UV Head Polymerization
* Material Database
* Abstract Hardware Interface
* Modular and Scalable Design

# Appendices

## Slic3r

Slic3r is an open source slicing engine used for 3-D printing. Slic3r allows the user to input printing parameters as well as object files, such as STL or AMF, and produce G-Codes for 3-D printing those objects. Slic3r exposes a command line interface that will be used by the Processing Layer of the 3-D Printer Fabrication System to produce G-Codes.

## OpenSCAD

OpenSCAD is an open source, light weight CAD utility that can be used to create 3D models. The difference between OpenSCAD and most traditional CAD utilities is that OpenSCAD does not focus on a feature-rich graphical user interface, but instead accepts input script files that describe objects. OpenSCAD then builds the objects described by the script files and can export those objects as STL files. OpenSCAD is used in the Preprocessing Layer to create subsections of objects.