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](#_Toc379985840)

[1 Introduction 8](#_Toc379985841)

[1.1 Overview 8](#_Toc379985842)

[1.2 Project Concept 8](#_Toc379985843)

[1.3 Project Scope 8](#_Toc379985844)

[2 Architecture Overview 9](#_Toc379985845)

[2.1 Module Decomposition Tree 11](#_Toc379985846)

[2.2 Module Decomposition Diagram 12](#_Toc379985847)

[2.3 Module Data Flows Table 13](#_Toc379985848)

[2.4 Producer Consumer Table 13](#_Toc379985849)

[2.5 Module Functional Descriptions 13](#_Toc379985850)

[2.5.1 Use Interface Layer Modules 13](#_Toc379985851)

[2.5.2 Preprocessing Layer Modules 13](#_Toc379985852)

[2.5.3 Processing Layer Modules 13](#_Toc379985853)

[2.5.4 Post Processing Layer Modules 13](#_Toc379985854)

[2.5.5 Physical Layer Modules 13](#_Toc379985855)

[2.5.6 Printer Feedback Layer Modules 13](#_Toc379985856)

[2.5.7 Communication Layer Modules 13](#_Toc379985857)

[3 Data Descriptions 14](#_Toc379985858)

[3.1 Print Job Configuration Aggregation Hierarchy 14](#_Toc379985859)

[3.2 PrintJobConfiguration Class 15](#_Toc379985860)

[3.2.1 Aggregation Relationships 15](#_Toc379985861)

[3.2.2 Data Elements 15](#_Toc379985862)

[3.3 PrinterConfiguration Class 15](#_Toc379985863)

[3.3.1 Aggregation Relationships 15](#_Toc379985864)

[3.3.2 Data Elements 15](#_Toc379985865)

[3.4 SubsectionConfiguration Class 16](#_Toc379985866)

[3.4.1 Aggregation Relationships 16](#_Toc379985867)

[3.4.2 Data Elements 16](#_Toc379985868)

[3.5 PrintConfiguration Class 17](#_Toc379985869)

[3.5.1 Aggregation Relationships 17](#_Toc379985870)

[3.5.2 Data Elements 17](#_Toc379985871)

[3.6 InfillConfiguration Class 17](#_Toc379985872)

[3.6.1 Aggregation Relationships 17](#_Toc379985873)

[3.6.2 Data Elements 17](#_Toc379985874)

[3.7 LayerAndPerimeterConfiguration Class 18](#_Toc379985875)

[3.7.1 Aggregation Relationships 18](#_Toc379985876)

[3.7.2 Data Elements 18](#_Toc379985877)

[3.8 SpeedConfiguration Class 19](#_Toc379985878)

[3.8.1 Aggregation Relationships 19](#_Toc379985879)

[3.8.2 Data Elements 19](#_Toc379985880)

[3.9 SkirtAndBrimConfiguration Class 20](#_Toc379985881)

[3.9.1 Aggregation Relationships 20](#_Toc379985882)

[3.9.2 Data Elements 20](#_Toc379985883)

[3.10 SupportMaterialConfiguration Class 21](#_Toc379985884)

[3.10.1 Aggregation Relationships 21](#_Toc379985885)

[3.10.2 Data Elements 21](#_Toc379985886)

[3.11 FileConfiguration Class 22](#_Toc379985887)

[3.11.1 Aggregation Relationships 22](#_Toc379985888)

[3.11.2 Data Elements 22](#_Toc379985889)

[3.12 MaterialConfiguration Class 23](#_Toc379985890)

[3.12.1 Aggregation Relationships 23](#_Toc379985891)

[3.12.2 Data Elements 23](#_Toc379985892)

[3.13 ExtruderConfiguration Class 25](#_Toc379985893)

[3.13.1 Aggregation Relationships 25](#_Toc379985894)

[3.13.2 Data Elements 25](#_Toc379985895)

[4 User Interface Layer 26](#_Toc379985896)

[5 Preprocessing Layer 36](#_Toc379985897)

[5.1 Normalization Subsystem Modules 36](#_Toc379985898)

[5.1.1 Object Subsection Module 36](#_Toc379985899)

[5.1.2 File Translation Module 37](#_Toc379985900)

[6 processing Layer 38](#_Toc379985901)

[6.1 Slicing Subsystem Modules 38](#_Toc379985902)

[6.1.1 Slicing Engine Wrapper 38](#_Toc379985903)

[6.1.2 Slicing Engine 39](#_Toc379985904)

[7 Post processing Layer 40](#_Toc379985905)

[7.1 G-Code Preparation Subsystem Modules 40](#_Toc379985906)

[7.1.1 Parser Module 40](#_Toc379985907)

[7.1.2 Unification Module 40](#_Toc379985908)

[8 Physical Layer 42](#_Toc379985909)

[8.1 [Subsystem] Modules 42](#_Toc379985910)

[8.1.1 [Module Name] 42](#_Toc379985911)

[9 Printer Feedback Layer 43](#_Toc379985912)

[9.1 [Subsystem] Modules 43](#_Toc379985913)

[9.1.1 [Module Name] 43](#_Toc379985914)

[10 Communication Layer 44](#_Toc379985915)

[10.1 [Subsystem] Modules 44](#_Toc379985916)

[10.1.1 [Module Name] 44](#_Toc379985917)

[11 Quality Assurance 45](#_Toc379985918)

[11.1 Unit Testing 45](#_Toc379985919)

[11.1.1 [Layer Name] 45](#_Toc379985920)

[11.2 Component Testing 45](#_Toc379985921)

[11.3 Integration Testing 45](#_Toc379985922)

[11.4 System Verification Testing 45](#_Toc379985923)

[11.5 Test Cases 45](#_Toc379985924)

[12 45](#_Toc379985925)

[13 Requirements Traceability Matrix 46](#_Toc379985926)

[14 Acceptance Plan 47](#_Toc379985927)

[14.1 Package and Installation 47](#_Toc379985928)

[14.2 Acceptance Testing 47](#_Toc379985929)

[14.3 Acceptance Criteria 47](#_Toc379985930)

[15 Appendices 48](#_Toc379985931)

Document Revision History

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

List of Figures

Figure # Title Page #

1-1 System Concept ??

2-1 Architecture Diagram ??

2-2 Decomposition Tree ??

2-3 Detailed Design Diagram ??

List of Tables

Figure # Title Page #

2.2 External Inputs and Outputs 11

11.1 Cost Analysis Table 35

11.2 Function Points Analysis 36

11.3 Influence Multipliers 38

11.4 Jones First Order Estimation 39

11.5 Size Estimate – Lines of Code 39

11.6 CoCoMo Estimate 40

11.7 Estimate Comparison 40

# 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

[Description]

[Module Data Flows Table]

## Producer Consumer Table

[Description]

[Producer Consumer Table]

## Module Functional Descriptions

### Use Interface Layer Modules

[Module Functional Descriptions]

### Preprocessing Layer Modules

[Module Functional Descriptions]

### Processing Layer Modules

[Module Functional Descriptions]

### Post Processing Layer Modules

[Module Functional Descriptions]

### Physical Layer Modules

[Module Functional Descriptions]

### Printer Feedback Layer Modules

[Module Functional Descriptions]

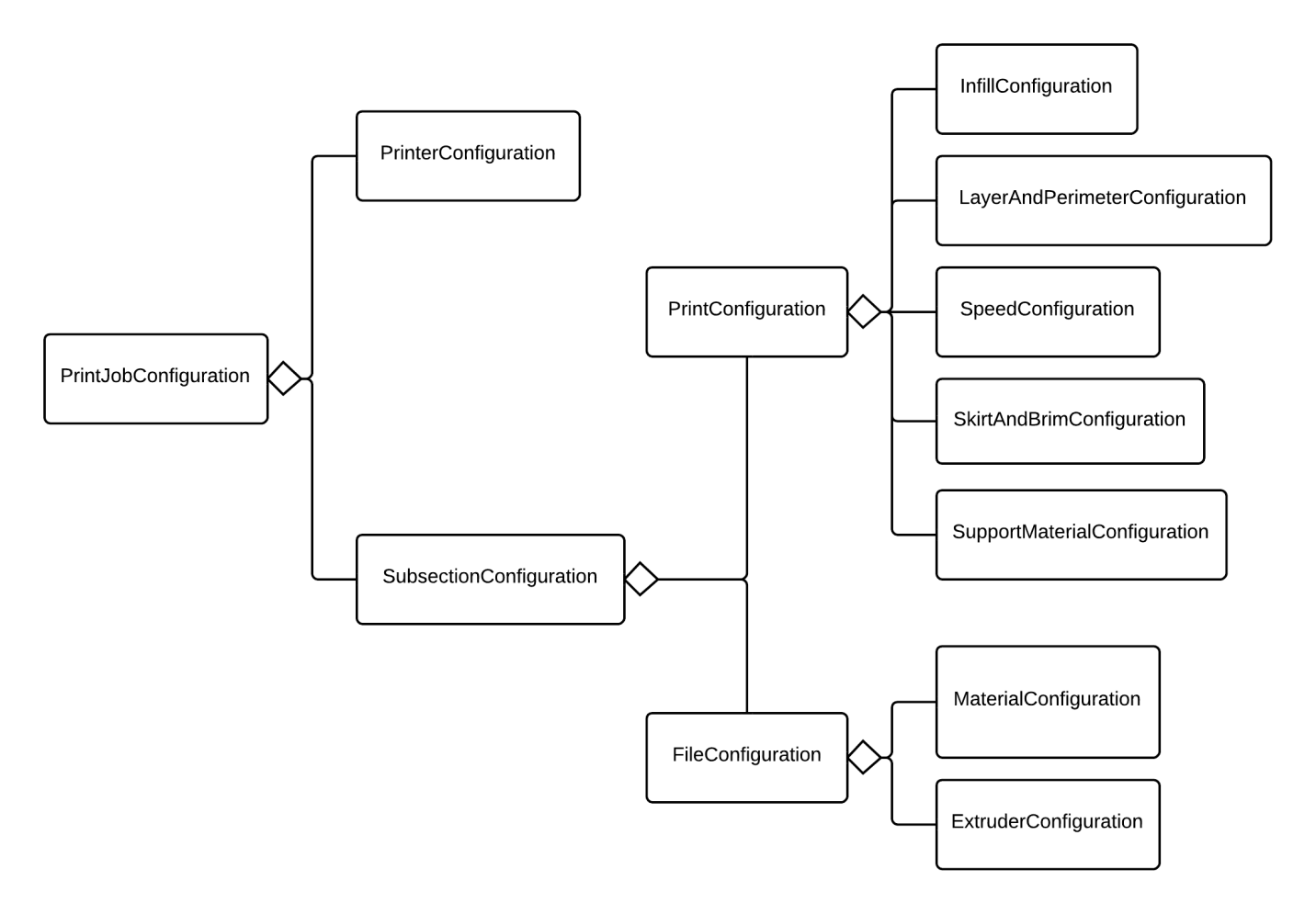
### Communication Layer Modules

[Module Functional Descriptions]

# 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



## PrintJobConfiguration Class

### Aggregation Relationships



### 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 |

## PrinterConfiguration Class

### Aggregation Relationships



### 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. |

## SubsectionConfiguration Class

### Aggregation Relationships



### 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 |

## PrintConfiguration Class

### Aggregation Relationships



### Data Elements

None

## InfillConfiguration Class

### Aggregation Relationships



### 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 |

## LayerAndPerimeterConfiguration Class

### Aggregation Relationships



### 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. |

## SpeedConfiguration Class

### Aggregation Relationships



### 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. |

## SkirtAndBrimConfiguration Class

### Aggregation Relationships



### 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. |

## SupportMaterialConfiguration Class

### Aggregation Relationships



### 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. |

## FileConfiguration Class

### Aggregation Relationships



### 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 |

## MaterialConfiguration Class

### Aggregation Relationships



### 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. |

## ExtruderConfiguration Class

### Aggregation Relationships



### 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 first extruder. | mm | Greater than or equal to 0. |
| yOffset | double | The y offset in respect to the first extruder. | mm | Greater than or equal to 0. |
| zOffset | double | The z offset in respect to the first extruder. | mm | Greater than or equal to 0. |

# 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, GUI Subsystem is responsible for displaying the printer and current print status to the user.

### Import GUI Module

#### Prologue

Import GUI Module is a menu that allows the user to import or remove AMF and STL files into or from the system.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| User input | .stl or .amf file | .stl or .amf file |
|  |  |  |
|  |  |  |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
|  |  |
|  |  |

#### Internal Data Descriptors

[NOT EXACTLY SURE WHAT THIS IS]

#### Process

importTab(){

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

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

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

Add button to importTab.pane named "Save" with function importController.save(model);

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

}

### Material 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 |
| User input | User input strings | Array of strings |
|  |  |  |
|  |  |  |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
|  |  |
|  |  |

#### Internal Data Descriptors

[NOT EXACTLY SURE WHAT THIS IS]

#### Process

materialTab(){

Add tab to main window frame named "Materials";

Add button to materialTab.pane named "New" with function materialController.new();

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

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

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

}

### Printer Hardware Configuration GUI Module

#### Prologue

Printer Hardware 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 sets to be used again later.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| User input | User input strings | Array of strings |
|  |  |  |
|  |  |  |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
|  |  |
|  |  |

#### Internal Data Descriptors

[NOT EXACTLY SURE WHAT THIS IS]

#### Process

printerConfigTab(){

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

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

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

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

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

}

### 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 |
| User input | User input strings | Array of strings |
|  |  |  |
|  |  |  |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
|  |  |
|  |  |

#### Internal Data Descriptors

[NOT EXACTLY SURE WHAT THIS IS]

#### 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.viewPrintConfigDB();

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 AMF or STL files to be printed and maps them with appropriate materials listed from the database.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| subsections | User input strings | Array of strings and pointers to selected files |
|  |  |  |
|  |  |  |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
|  |  |
|  |  |

#### Internal Data Descriptors

[NOT EXACTLY SURE WHAT THIS IS]

#### 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 while the print is ongoing. The window will also include a button that allows the user to pause and resume the printing process.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| Get Status | Temperature (float or double) Current position (vec3) | Temperature (float or double) Current position (vec3) |
| Pause and Resume | Button click | Pause/Resume Command |
|  |  |  |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
|  |  |
|  |  |

#### Internal Data Descriptors

[NOT EXACTLY SURE WHAT THIS IS]

#### 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;

Add text to rightStatus.pane (statusController.getTemp() + "\n" + statusController.getX() + "\n" + statusController.getY + "\n" + statusController.getZ);

}

## 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 AMF and STL files from the system when called by the Import GUI Module.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| Save | .amf or .stl file | .amf or .stl file |
|  |  |  |
|  |  |  |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
|  |  |
|  |  |

#### Internal Data Descriptors

[NOT EXACTLY SURE WHAT THIS IS]

#### Process

add(){

Open windows explorer to select .stl or .amf file;

//Don't know the save/open windowing process in code;

Model = selected .amf or .stl file

Call function framework.save();

}

save(model){

Call function framework.save(model);

}

### Material Controller

#### Prologue

Material Controller Module carries out the functions related to storing and managing material info. This includes storing the user input as an xml file, modifying an xml file when new user input is given, or deleting an xml upon request.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| Save | Array of strings | Xml file |
| Open request | Xml file | Array of strings |
|  |  |  |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
|  |  |
|  |  |

#### Internal Data Descriptors

[NOT EXACTLY SURE WHAT THIS IS]

#### Process

Pseudo Code

### Printer Hardware Configuration Controller

#### Prologue

The Printer Hardware Configuration Controller Module carries out the functions that lets the user store and retrieve information that the system needs to understand the capabilities of the 3D printer. These functions include storing the user input as an XML file, modifying an XML file that describes the printer hardware, or deleting such XML files upon request.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| Save | Array of strings | Xml file |
| Open Request | Xml file | Array of strings |
|  |  |  |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
|  |  |
|  |  |

#### Internal Data Descriptors

[NOT EXACTLY SURE WHAT THIS IS]

#### 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.viewPrintConfigDB();

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

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

}

viewPrinterConfigDB(){

Call function framework.viewPrintConfigDB();

}

save(model){

Call function framework.save(model);

}

### Print Configuration Controller

#### Prologue

The Print Configuration Controller Module carries out the functions that lets the user store or retrieve a set of credentials for a print job. These functions include storing the user input as an XML file, modifying an XML file that describes the print configuration, or deleting such XML files upon request.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| Save | Array of strings | Xml file |
| Open request | Xml file | Array of strings |
|  |  |  |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
|  |  |
|  |  |

#### Internal Data Descriptors

[NOT EXACTLY SURE WHAT THIS IS]

#### Process

Pseudo Code

### 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 or AMF 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 |
| Package | Model file(s) (.stl or .amf) Array of subsection sizes (int) Print Configuration(s) (.xml) Printer Hardware Configuration (.xml) Material info (.xml) Array map of material to models (int) | Print Job Configuration Object |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
|  |  |
|  |  |

#### Internal Data Descriptors

[NOT EXACTLY SURE WHAT THIS IS]

#### Process

Pseudo Code

### 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 Communication Layer upon request.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
| Hardware Listening | Temperature (float or double) Current position (vec3) | Temperature (float or double) Current position (vec3) |
| Pause and Resume | Button click | Pause/Resume Command |
|  |  |  |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
|  |  |
|  |  |

#### Internal Data Descriptors

[NOT EXACTLY SURE WHAT THIS IS]

#### Process

Pseudo Code

## 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.

### Persistence Framework

#### Prologue

[Name/Description/Function]

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
|  |  |  |
|  |  |  |
|  |  |  |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
|  |  |
|  |  |

#### Internal Data Descriptors

[NOT EXACTLY SURE WHAT THIS IS]

#### Process

Pseudo Code

### Command Structure

#### Prologue

[Name/Description/Function]

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
|  |  |  |
|  |  |  |
|  |  |  |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
|  |  |
|  |  |

#### Internal Data Descriptors

[NOT EXACTLY SURE WHAT THIS IS]

#### Process

Pseudo Code

# Preprocessing Layer



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 Print 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



The Normalization Subsystem is responsible for receiving the bundle containing the object data and printer configurations from the User Interface’s Print Subsystem and converting the object data into a format the Processing Layer can understand. After the object file(s) is converted, this subsystem sends a modified bundle of the object file information and the printer configuration to 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 |

#### External Data Dependencies

* OpenSCAD executable

#### 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 |

#### External Data Dependencies

None

#### 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



The Processing Layer takes the formatted package and translates it into G-codes. The slicing engine is the only subsystem in this layer. The slicing engine 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. One the Processing Layer processes the normalized data from the Preprocessing Layer, the Processing Layer passes a packet of G-Codes and printer configuration information to the Post-Processing Layer.

## Slicing Subsystem Modules



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 |

#### External Data Dependencies

* Slic3r Slicing Engine

#### 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



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 physical 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



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 also cleaning up any unnecessary G-Codes.

#### Interfaces

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

#### External Data Dependencies

None

#### 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(isVaidToken(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 |

#### External Data Dependencies

None

#### 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 large data structure 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 |
|  |  |  |
|  |  |  |
|  |  |  |

#### External Data Dependencies

None.

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| G-Code Object | Unification Module |
| Printer Feedback Status | Dispatch Module |

#### Process

[Pseudo-code]

# 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 physical 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 physical 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 primarily be arm position and extruder temperature. Other parameters can be added as necessary. The incoming data will be packaged by the dispatch module into a data structure that can be read by user interface layers and the printer control layers.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
|  |  |  |
|  |  |  |
|  |  |  |

#### External Data Dependencies

None

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| Printer Status Object | Deserialization Module |

#### Process

[Pseudo-code]

# Communication Layer

The purpose of the communication layer is to serve as an interface between the client software (The 3-D Printer Fabrication System) and the printer firmware itself. The communication 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).

## Communication Subsystem Modules

The purpose 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 the user interface layer when the software is first run. 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 |
|  |  |  |
|  |  |  |
|  |  |  |

#### External Data Dependencies

|  |  |
| --- | --- |
| Data | Source |
| Printer Status | Printer Firmware |

#### Internal Data Descriptors

|  |  |
| --- | --- |
| Data | Source |
| Serialized G-Code Buffer | Serialization Module |

#### Process

[Pseudo-code]

### Serialization Module

#### Prologue

The purpose of the serialization module is to receive the entire set of G-Codes that will carry out the print run and serialize them in preparation to be sent to the TX/RX 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 bulk of G-Codes down to byte streams according to the buffer configuration and pack the buffer.

#### Interfaces

|  |  |  |
| --- | --- | --- |
| Interface | Information Required | Information Returned |
|  |  |  |
|  |  |  |
|  |  |  |

#### External Data Dependencies

None.

#### Internal Data Descriptors

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

#### Process

[Pseudo-code]

### 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 |
|  |  |  |
|  |  |  |
|  |  |  |

#### External Data Dependencies

None.

#### Internal Data Descriptors

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

#### Process

[Pseudo-code]

# Quality Assurance

[Description of test plans and procedures]

## Unit Testing

### [Layer Name]

#### [Subsystem Name]

##### [Module Name]

## Component Testing

[NOT SURE WHAT EXACTLY THIS IS YET]

## Integration Testing

[Description of integration testing]

## System Verification Testing

[Description of system verification testing (possibly take from SRS)]

## Test Cases

[Description]

|  |  |
| --- | --- |
| Test Case | Expected Result |
|  |  |
|  |  |

# 

# Requirements Traceability Matrix

[Description]

[Requirements traceability matrix (probably want to break this into layers)]

# Acceptance Plan

[Description/Overview]

## Package and Installation

[Probably use requirements from SRS]

## Acceptance Testing

[Description]

## Acceptance Criteria

[Reiterate acceptance criteria for requirements]

# Appendices