Email to 3D Print

Design Document

Revised 20 October 2014

Sections

---

1 Overview

The goal of this project is to provide a program (written in Python), that acts as a pipeline for converting a valid 3D model, received via an email interface, into G-code instructions that can be interpreted by a MakerBot Replicator2 3D printer. This project aims to limit human interaction in the 3D printing process, by providing powerful tools to do the compilation and printing work, and an easy to use email interface that keeps the user informed about their print job's status. The only human interaction beyond an email submission, will be the removing of the previous build from the plate, and verification that the system is ready for the next build by Collexion members.

2 Specifications/Features to be completed

(in priority order and as time allows)

- Should be able to receive a user submitted email, and verify that it contains an OBJ or STL formatted attachment.
- Should be able to convert the model to STL format
- Should be able to validate the model can be printed
- Should be able to slice the model (convert to G-code)
- Should print or queue the model, only when the printer status is OK
- Printer status should be monitored
- Print job status/build plate status should be logged
- User should receive a notification email if there is a problem with the printer or their job
- Detect if the printer is in an OK state automatically so the next job can be printed

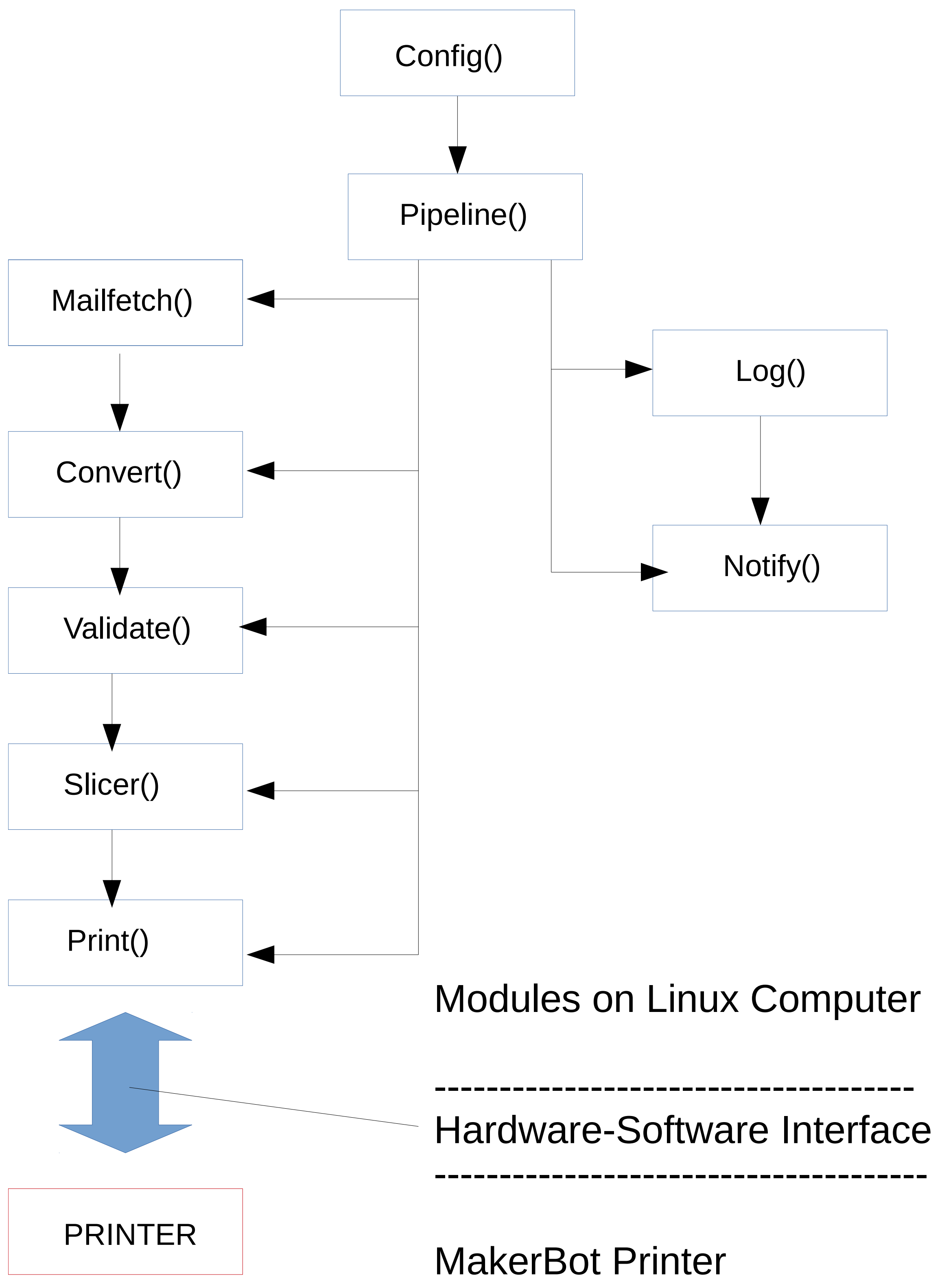
3 Environment

- The application will reside on a workstation/computer provided by the customer
- OS: Linux distribution or virtual machine provided by customer
- Language: Python3
- Various open source projects already exist to perform slicing and to act as a“printer driver”. We anticipate using programs such as Slic3r and ReplicatorG to this end.

4 Module Descriptions and Data Flow

Although the design and anticipated interaction between functions is still quite fluid, this is a good faith, best current estimate of module behavior at this point. A diagram of the module interfaces is first presented, followed by a description of the purpose, inputs, outputs, and design of each module.

Module Interface Diagram



--

Pipeline()

Purpose:

Provide the infrastructure of a PrintJob object to encapsulate data pertinent to each given print request, such as sender, status, file paths, ...

Inputs:

No explicit inputs

Outputs:

A PrintJob object

Design:

In the spirit of object oriented methods, this module is a class with constructor, getters, setters, etc...

Tracks the current pipeline stage (block diagram phase completed)

Contains sender information

Contains system related info such as timing and file paths

--

Config()

Purpose:

Read/write configuration values from/to a configuration file on disk

Inputs:

Read or write flag indicating action to be performed

Outputs:

Read configuration from disk or writes configuration to disk

Design:

If flag is READ, read configuration from disk

IF flag is WRITE, write configuration data to file on disk

--

Mailfetch()

Purpose:

Fetch STL or OBJ attachments sent via email by users

Inputs:

Module configuration parameters

Outputs:

Save attachments to disk

Create a PrintJob object containing attachment and sender email address

Design:

```
Read configuration data from a file on disk
Open a socket to the Google mail server
Login to the e-mail account
Generate a list of unread messages from  the inbox
For each message containing a valid attachment:
    - Capture the e-mail address of the sender
    - Save the attachment to disk
    - Create a PrintJob object to pass to later modules
Logout of e-mail account and close connection socket
After a number of minutes have elapsed, return to step 2
```

--

Convert()

Purpose:

Convert OBJ to STL format model

Inputs:

OBJ file

Outputs:

STL file

Design:

```
If file extension is .stl, do nothing
else if file extension is .obj convert to .stl
else error condition
```

Validate()

Purpose:

Verify that STL model is valid/printable

Inputs:

STL file

Outputs:

Error if file is not valid or printable

Design:

```
Execute a sequence of validation routines
If any return a failure condition log the error and send email to user
```

Slicer()

Purpose:

Produce G-code, which provides low level instructions allowing the target hardware to print the intended object, layer by layer.

Inputs:

An STL format model file

Outputs:

G-code if slicing is successful

Error log and email, otherwise

Design:

```
Call slic3r or replicatorG with appropriate arguments for slicing
If error:
    Log and send email to user
Else:
        Save G-code in file to be sent to hardware for printing
```

Print()

```
Purpose:
    Physically print the requested model
Inputs:
    File containing G-codes
Outputs:
    Hardware prints the requested design or report error is failure occurs
Design:
    Call replicatorG to print the model
    If error:
        Log and send email to user
    Else:
        Model was printed successfully
        Notify user by email that job was successfully completed
        Log job completion and update quota statistics
```

Log()

```
Purpose:
    Write a message tot he error log

Inputs:
    Pipeline stage
    User/sender email address
    Error message

Outputs:
    Write time, user name, stage, and error message to an error log file

Design:
        Open log file for writing
        Write time, user name, stage, and error to log file
        Close log file
```

Notify()

Purpose:

Construct and send an email to the user advising of success or failure

Inputs:

Pipeline stage

Output:

Email to the user informing them of success, or failure and the point of failure

Design:

```
    Construct a properly formed header
    Construct a "form letter" body based on value of the pipeline stage provided
    Send the message to user
```

5 User Screens

This program is designed to operate as a server program running on a local computer connected to the printer hardware. We are not targeting a GUI or non command line execution.

6 User Scenarios (Use Cases)

Case 1 - Normal Case

Here is an example of an anticipated normal user scenario:

1. User submits a .stl or .obj model to be printed to a Gmail account designated for such purpose.

2. Program detects than an unread message has arrived in the INBOX.

3. The attachment is a recognized model file format, so the file is saved to disk, and the user name of the sender is captured.

4. Program verifies that the model file is valid and printable.

5. Model file is converted to G-code.

6. G-code is sent via usb interface and replicatorG software to the Makerbot printer.

7. Model is printed. Job statistics are logged, and user is notified by email that the print job completed successfully.

Case 2 – Print job failed

Here is an example of a failure to print by the hardware:

1. User submits a .stl or .obj model to be printed to a Gmail account designated for such purpose.

2. Program detects than an unread message has arrived in the INBOX.

3. The attachment is a recognized model file format, so the file is saved to disk, and the user name of the sender is captured.

4. Program verifies that the model file is valid and printable.

5. Model file is converted to G-code.

6. G-code is sent via usb interface and replicatorG software to the Makerbot printer.

7. Model did not print.

8. Error is logged and user is notified of the failure and reason for failure.

Case 3 – Model file is invalid

Here is an example of a scenario where the submitted model is invalid or not printable:

1. User submits a .stl or .obj model to be printed to a Gmail account designated for such purpose.

2. Program detects than an unread message has arrived in the INBOX.

3. The attachment is a recognized model file format, so the file is saved to disk, and the user name of the sender is captured.

4. Program examines that the model file is invalid.

5. Error is logged and user is notified of the failure and reason for failure.

Case 4 – Model file is not STL or OBJ

In this example, the program did not receive an STL or OBJ file, due to a mistake or misunderstanding by the target user(s), a spam email that bypassed the Gmail filters, or due to some other unanticipated reason:

1. User submits an unsupported model to be printed to a Gmail account designated for such purpose.

2. Program detects than an unread message has arrived in the INBOX.

3. The attachment is not a supported model file format, so no further action is taken.

7 Design Considerations

- Python was chosen due to text processing and scripting capability, as well as developer familiarity.

- A variety of 3D model formats exist. In the interest or brevity and project focus, we will only accept OBJ and STL submissions, and since STL is the most common format, slicing and low level operations will be performed on STL files.

- Verifying that a file is printable is a complex problem. To this point, our research has not found a straightforward or simple way to do this. Writing that portion of the code from the ground up would probably be beyond the scope of a 2 month project.

- We may not be able to check if platform is clear. Such work may ultimately involve sensors and hardware related issues that are beyond the scope of out expertise.

8 Sizing Estimate

The intermediate COCOMO method has been used to determine the number of PERSON-MONTHS required for this project. The formula is given as:

$$\mathbf{E = a\_i(KLOC)^{(b\_i)} * EAF}$$

Assumptions and Estimations:

This is an “Organic Project”, with little or no preexisting code base, and a small team of developers working on the project. Therefore, a_i = 3.2, b_i = 1.05

KLOC is “kilo lines of code”. This is a very rough, uneducated guess, but the “gut feeling” is that this project is closer to 1000 lines than 2000 or more. Therefore, KLOC = 1.

Next, we calculate the EAF (effort adjustment factor) by rating a number of categories from very low to very high, and providing the appropriate adjustment from a table.

Required software reliability: 1.15

Size of application database: 0.94

Complexity of product: 1.00

Run time constraints: 1.00

Memory constraints: 1.00

Volatility of VM environment: 1.00

Required turnabout time: 1.15

Analyst capability: 1.19

Applications experience: 1.13

Software engineer capability: 1.17

Virtual machine experience: 1.00

Programming language experience: 1.00

Application of SW engineering methods: 1.10

Use of software tools: 1.10

Required development schedule 1.10

EMF is calculated as the product of the above 15 estimates, which is ~2.60

So:

#PERSON-MONTHS = $3.2(1)^{1.05} * 2.60$ = **~ 8.3 PERSON-MONTHS**

9 Documentation specific to our project

None at this time, but may be added in the future if needed.