# WEB-BASED MODEL SLICING FOR 3D PRINTERS

BY

MICHAEL U.B. MEDING
B.S., UNIVERSITY OF MASSACHUSETTS LOWELL (2015)

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF MASSACHUSETTS LOWELL

Author .....
December 1, 2010

Certified by .....
Fred G. Martin
Associate Professor
Thesis Supervisor

Certified by .....
Jeff Brown
Associate Professor
Thesis Reader

Accepted by .....

# Web-Based Model Slicing for 3D Printers

by

Michael U.B. Meding

Abstract of a thesis submitted to the faculty of the
Department of Computer Science
in partial fulfillment of the requirements
for the degree of
Master of Science
University of Massachusetts Lowell
2016

Thesis Supervisor: Fred G. Martin
Title: Associate Professor

Thesis Reader: Jeff Brown
Title: Associate Professor

## Abstract

3D printing currently has a large gap between software and hardware. A hobbyist machine can now be purchased for less than $500 but having good software to drive it is hard to find. Currently the only competent and free slicing software available is Cura and Repetier Host. Currently, Cura has varied support on all platforms, and Repetier Host is intended only for Windows. Neither software have any web support nor will they be likely to have any support in the future as the slicing process requires a computer with a considerable amount of both graphical and computational power. The purpose of this research is to construct a web based slicing software and make it simple for users without any prior knowledge of 3D printing to take full advantage of their printer and as a result will make 3D printing much more approachable for users who are not computer savvy. Additionally, this opens up opportunities for educators in STEM programs to teach students about 3D printing in a simple and practical way.

# Acknowledgments

# Contents

# List of Figures

# List of Tables

# Listings

# Chapter 1

# Introduction

3D printing, over the past few years, has become immensely popular in the home hobbyist space because of its new found availability. A hobbyist can now go and buy a do-it-yourself 3D printer kit for less than $500. Even though the hardware to build a 3D printer is easily available, the software support leaves much to be desired. Most of the big name companies that used to hold all of the patents for 3D printers have retained the software patents but not the hardware patents. This creates a gap in knowledge between building the printer and actually running it. This proposed research is to find out what software already exists in the open source world and then try to expose this on the web in a simple and easy to use manner. This effectively will democratize the world of 3D printing, much in the same way that Google has democratized the way that we search the web. In an article written by Harvard Business review they theorize that this rise in the popularity of 3D printing will spur an industrial revolution as manufacturing becomes more personalized and decentralized. D'Aveni (2015)

To print something on a 3D printer, there is a multi step process that can be daunting to many first time users. The first step in the process is to either create or download a model. Creating a model can be done with any standard 3D CAD software, such as AutoCAD or SolidWorks. Downloading pre-existing models from an online repository can be done from websites such as Thingiverse or YouMagine. Once

a model file is obtained, it is time to slice the model. Slicing is the act of taking this model file and splitting it up into many thin layers that the 3D printer can understand. This process can only be done by a dedicated slicing software which can often be complicated to use and difficult to install. Once the file has been sliced, the resulting file is a G-code file which is simply a set of movement instructions that the printer head must follow. This file is then loaded to an SD card or sent via a print server similar to the way that a normal 2D printer is networked. Once the G-code file has been loaded all that is left is to hit print either manually using the printers interface for the SD card or by hitting print on the network interface for the file that was uploaded.

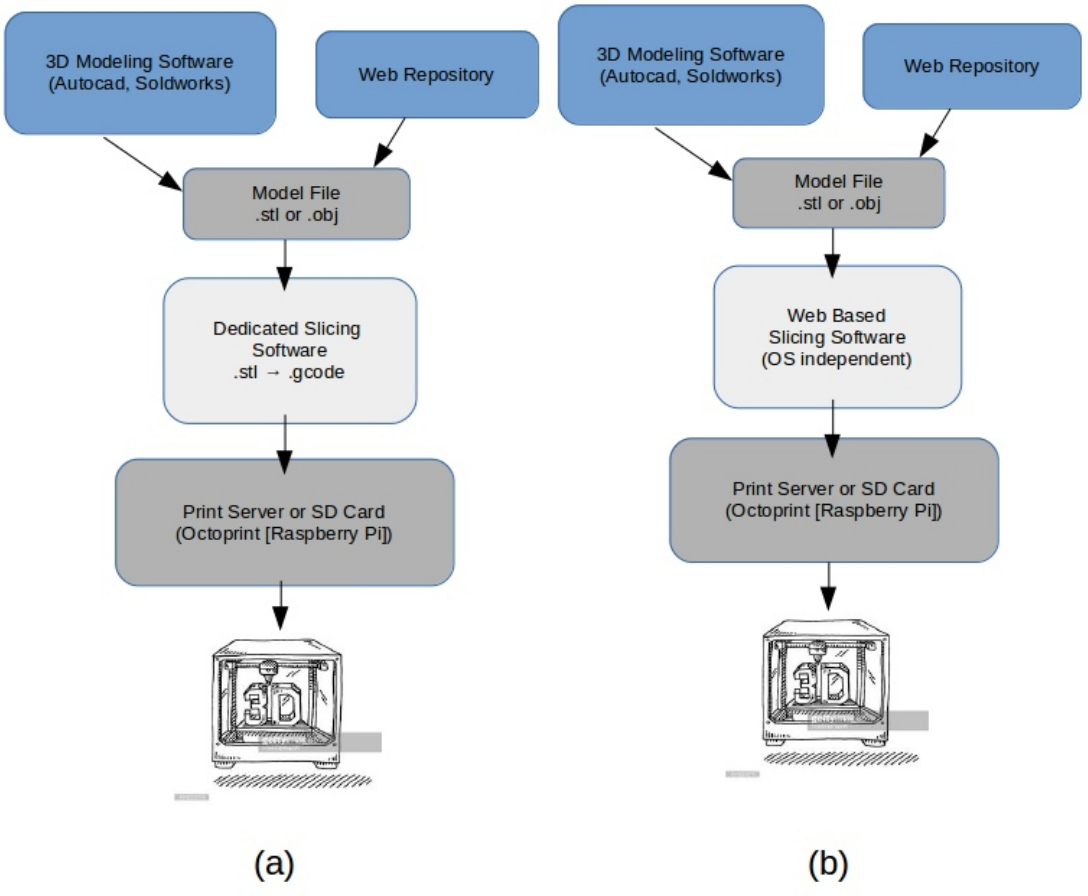


Figure 1-1: (a) High level view of the normal 3D printing process. (b) High level view of proposed new process using WebSlicer.

## 1.1 The RepRap Idea

The Replication Rapid-Prototyper Project (RepRap) is a movement with the goal of providing Open-source, diy 3D printers at low cost. RepRap printers are 3D printers with the additional ability to produce most of the parts necessary to assemble another identical printer. This idea also extends outside of hardware but to the software as well. Much of the software available for RepRap style printers are open source projects put together by the community.

## 1.2 Sudden Growth Of 3D Printing

In an article written by Forbes they did an analysis of the current market trend for 3D printing and were surprised to find that it is becoming one of the fastest growing emerging markets. "According to Wohlers Report 2014, the worldwide 3D printing industry is now expected to grow from $3.07 billion in revenue in 2013 to $12.8 billion by 2018, and exceed $21 billion in worldwide revenue by 2020". Columbus (2015) One reason that 3D printing has been reaching more people is the availability of the RepRap designs. Additionally, to build a 3D printer from a kit only requires basic hand tools and basic electronics skills which means that it has been opened up to a much broader audience.

## 1.3 Purpose of This Research

The purpose of this research is to construct a web based slicer and make it simple for a user, who knows hardly anything about 3D printing, to slice models and run their 3D printer. This will make 3D printing much more approachable for occasional users who are not prepared to spend thousands of dollars on a professional machine and software. Additionally, this opens up opportunities for educators in STEM programs to teach students about 3D printing in a simple and practical way. Under normal circumstances this would not be a feasible project for a year long Masters thesis. However, many of the technologies required to complete this project exist in varying

states of completeness. Putting them together will be the subject of this study.

## 1.4 Research Objectives

This research has several milestones which must be met for this thesis to be considered complete. These milestones are in no particular order. However, several of them must be completed before allowing other milestones to be completed as detailed below.

### 1.4.1 See a model, get a model

Getting a model directly from a repository, such as Thingiverse, would be ideal for this web interface because it would not require any local storage for a model file. A file stored in local storage could be quite large and cause memory problems for the user.

### 1.4.2 Wrap CuraEngine

CuraEngine is an open-source slicing engine designed to take model files in .stl file format and convert them into G-code for 3D printing. For this project, wrapping this code and making it callable from the web would be the heart of this application.

### 1.4.3 Design intuitive web interface

This project requires both an intuitive and easy to way to slice a model file for 3D printing. This would be done using Bootstrap and AngularJS because they are both scalable and flexible for almost any web design. These technologies also allow for small scale and mobile use.

## 1.5 Existing Technology

AstroPrint is an all included cloud operating system for 3D printers. It attempts to encompass the entire package of 3D printing to a dedicated Raspberry Pi, or similar computer system. It is also one of the only forays I have found into a cloud based slicing software. Unfortunately, their software tries to accomplish too many tasks at once and has become somewhat like a Swiss army knife as it is capable of many tasks but can only do a few tasks well. Additionally, their cloud based slicing software, while being reasonably fast, lacks any support for reviewing the sliced model. This review stage is critical for anyone who is printing something that will take more than a few hours to complete.

## 1.6 Thesis Map

- Chapter 2, Existing libraries and why I chose them.
- Chapter 3, Software architecture.
- Chapter 4, Client side in detail.
- Chapter 5, Server side in detail.
- Chapter 6, Discussion about usability testing and further improvements.

# Chapter 2

# Libraries & Existing Code

## 2.1 CuraEngine

CuraEngine is the back end of a larger application called Cura. Cura is an open source 3D print slicer designed by Ultimaker and is part of their software suite for their Ultimaker line of 3D printers. It will be talked about extensively in latter portions of this paper as it is the main portion of the back end of WebSlicer.

The reason for choosing to use CuraEngine as my main slicing engine is that it has the most clear separation between application and interface. Another option was to use Slicer which is much older and has better documentation but is unfortunately written on windows and would require a large amount of extra effort to get working properly for my needs. CuraEngine is also platform agnostic as it is written in C++ and uses one library called protobuf which is its main interface library for the Cura application.

## 2.2 Client Side Libraries

When structuring a website for optimal layout and scalability there are few better options than using the combination of Bootstrap and AngularJS. Bootstrap is a client side CSS library which includes most commonly used CSS options such as

buttons and input fields and styles them all accordingly. AngularJS creates a client side environment to support higher level language constructs and features that are normally reserved for complex server side applications.

### 2.2.1 Bootstrap

Bootstrap is one of the most popular CSS and JS frameworks for developing responsive and mobile projects on the web. Being responsive means that Bootstrap is capable of being dynamically displayed on many different size screens. When the screen size changes Bootstrap is capable of moving and resizing elements on the page without loosing any content or overlapping items. In addition it speeds up the process of developing web based applications as it removes the need to write heavy amounts of CSS to make an application look and preform nicely. Bootstrap also includes many standard features which most web developers use every day further simplifying the process of building a website.

### 2.2.2 AngularJS

AngularJS is a framework which allows for easy development of dynamic web pages. It contains many ways to logically separate your code similar to popular object oriented programming languages like Java or C++. Freeman (2014) states, "The goal of AngularJS is to bring the tools and capabilities that have been available only for server-side development to the web client and, in doing so, make it easier to develop, test, and maintain rich and complex web applications" (p. 48). Therefore, the combination of Bootstrap for the unified look, responsiveness, and mobile support combined with the power of AngularJS as a framework make it the easy choice for a web based application such as WebSlicer.

## 2.3 JavaEE

JavaEE is a layer built on top of JavaSE the standard Java development environment. This additional layer provides many important architectural interfaces such as being able to put code into EJB containers or web containers for deployment of large applications which may have many of such containers. Pilgrim (2013) JavaEE also provides frameworks for working with RESTful web services which streamlines the creation of a useful application API.

There are many server side frameworks which have support for RESTful web services but most lack any kind of real scalability like that provided in JavaEE. Furthermore, JavaEE provides the ability to easily distribute your computing as more resources are needed. As this research may eventually scale to supporting many users it was logical to choose a framework which allowed for this kind of growth. Another reason for choosing JavaEE as a framework is that it provides the full platform cross compatibility which is standard in Java. Allowing for cross platform compatability would make it simple to link many computers that are running on different operating systems together further simplifying the prospect of scaling.

To run a JavaEE based application it must run in a web capable application container. This container must be placed on an application server which exposes it to the web under some context. For WebSlicer the application server WildFly was chosen for this task as it is well known for its reliability.

## 2.4 OctoPrint

OctoPrint is a small server that connects a Raspberry Pi to a 3D printer and serves printer info and files to the printer remotely. The Raspberry Pi serves a small webpage which allows you to easily keep track of the temperature of the printer and the current print progress. This interface allows for a web based connection between your remote g-code files and the local printer. Horvath (2014) OctoPrint also contains its own API

which makes for easy integration into other web applications.

The fact that OctoPrint is so easily integrated into other web based applications made it a logical choice for integration with WebSlicer. Integrating with OctoPrint would allow for one touch printing by allowing me to send g-code files directly to a connected OctoPrint server. The only interaction that is required of the user is to connect their OctoPrint server which is done by simply indicating the address at which the server can be reached at. Additionally, users are required to include an API key to allow for other applications to directly access the servers API.

## 2.5 G-Code Visualizer

When building the g-code visualizer it seemed logical to find one which already existed and worked well to include in my project. I came across an open source project called gCodeViewer originally written by Nils Hitze which would work perfectly for my needs. Hitze (2015) This project was written in pure JavaScript and would be seemingly easy to integrate with WebSlicer.

Unfortunately this could not have been farther from the truth as many of the libraries that were needed to make this code needed to be updated. Additionally, I wished to integrate this code into my existing AngularJS framework which turned out to be much more challenging than anticipated. This meant that what remained from the existing open source project was very small when all was said and done.

# Chapter 3

# Methodology

## 3.1 Research Design

This thesis is a mixture of both research and design implementation. The research portion of this project focused on linking an existing C++ application (CuraEngine) into a larger JavaEE based project. In this research we also had a small group of people run though some beta testing of the application and logged their observations.

## 3.2 Working procedure

As shown in Figure 3-1, the application will have 3 major components that all need to work together in a cycle until the user decides that the output is what they desire.

### 3.2.1 Web Interface

I anticipate that it will include a set of forms for collecting the users settings for their printer and account tracking info so that they may retain certain settings. Additionally, I would like to include a gcode editor which will allow users to edit specific portions of their gcode and see the resulting output in the viewer. This does not include the actual slicing engine which must be driven and accessed independently.

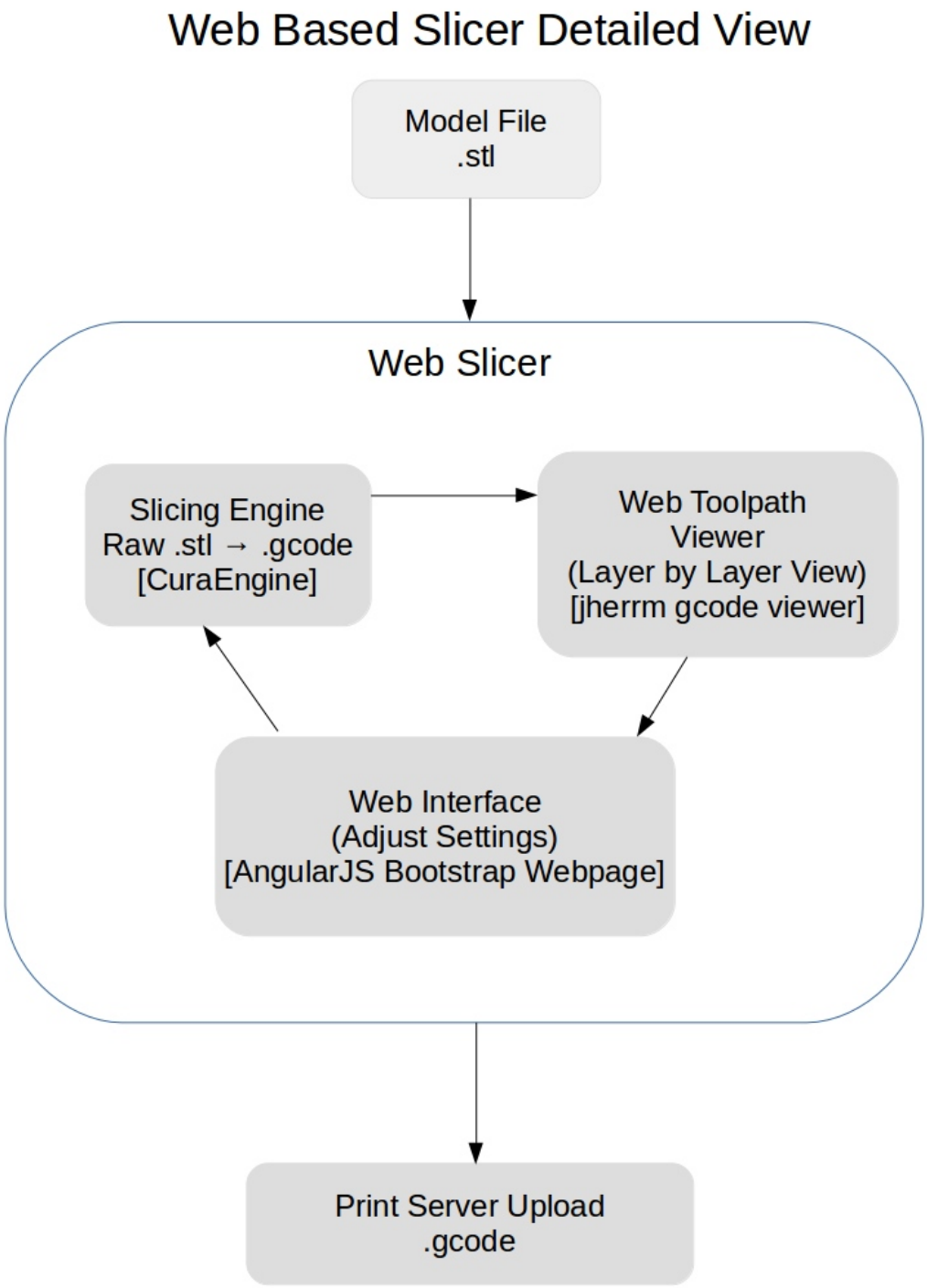


Figure 3-1: High level view of how WebSlicer functions and how users will interact with it

### 3.2.2 Slicing Engine

The slicing engine is at the heart of this project. It will include taking uploaded .stl model files by the user and convert them into raw G-code. The engine that carries out the raw geometry calculations is CuraEngine and is written in pure C++, (it is not web friendly). Thus, this portion of the project will require deploying CuraEngine at a cloud provider and then creating a RESTful API to interface with it.

### 3.2.3 Web Tool Path Viewer

After configuring and generating the G-code representation for a 3D model, there still must be a way to review visually how the slicing engine actually will split up the model.

### 3.2.4 Final Steps

The final step in the process of this application will be creating links to the finished files where the user can either choose to download the completed G-code file. If the user has access to a print server such as OctoPrint, they may opt to copy/paste the link to it where it will be uploaded automatically for printing using the OctoPrint API.

## 3.3 Review and Usability Testing

After running through all steps of the working procedure it was then necessary to test WebSlicer by running a small beta test. This beta test consisted of a small group of users with varying familiarity with 3D printing to test how easy WebSlicer is to use through a series of simple tasks.

# Chapter 4

# Client Side

## 4.1 AngularJS, a bit of background

To understand the structure of WebSlicer a few things must be known about how AngularJS applications are structured and a bit about how AngularJS itself works. Meding (2016), has more detailed explanations of each angular componenet than those that follow should the reader need more detail than that provided.

### 4.1.1 Controllers & Data Binding

### 4.1.2 Factories & Services

### 4.1.3 Directives

## 4.2 WebSlicer AngularJS Structure

The AngularJS structure of WebSlicer is shown in Figure 4-1. Flow through this diagram starts with the app.js node which represents the main controller of the application. This can be thought of as a main function in C++. The main control variables are also inside of app.js which are similar to global variables. Variables in this controller are used to store the current settings, file pointer, and output gcode.

Figure 4-1 also shows that index.html is large hub and as WebSlicer is a single page web application this is the only static HTML file. Index.html has several other functions such as bringing in all libraries and including the custom directives.

## 4.3 Settings Flow

As shown in Figure 4-2, the client side of this applicaiton has a lengthy flow of data. This data flow starts with loading a static JSON (JavaScript Object Notation) file which describes the settings in a pattern as shown in Listing 4.1.

Listing 4.1: A sample from a static settings file in JSON format.

```
{
    "setting": "layer_height",
    "default": 0.1,
    "type": "float",
    "category": "Quality",
    "label": "Layer Height (mm)",
    "description": "Layer height in millimeters. This is
       the most important setting to determine the quality
        of your print. Normal quality prints are 0.1mm,
       high quality is 0.06mm. You can go up to 0.25mm."
}
```

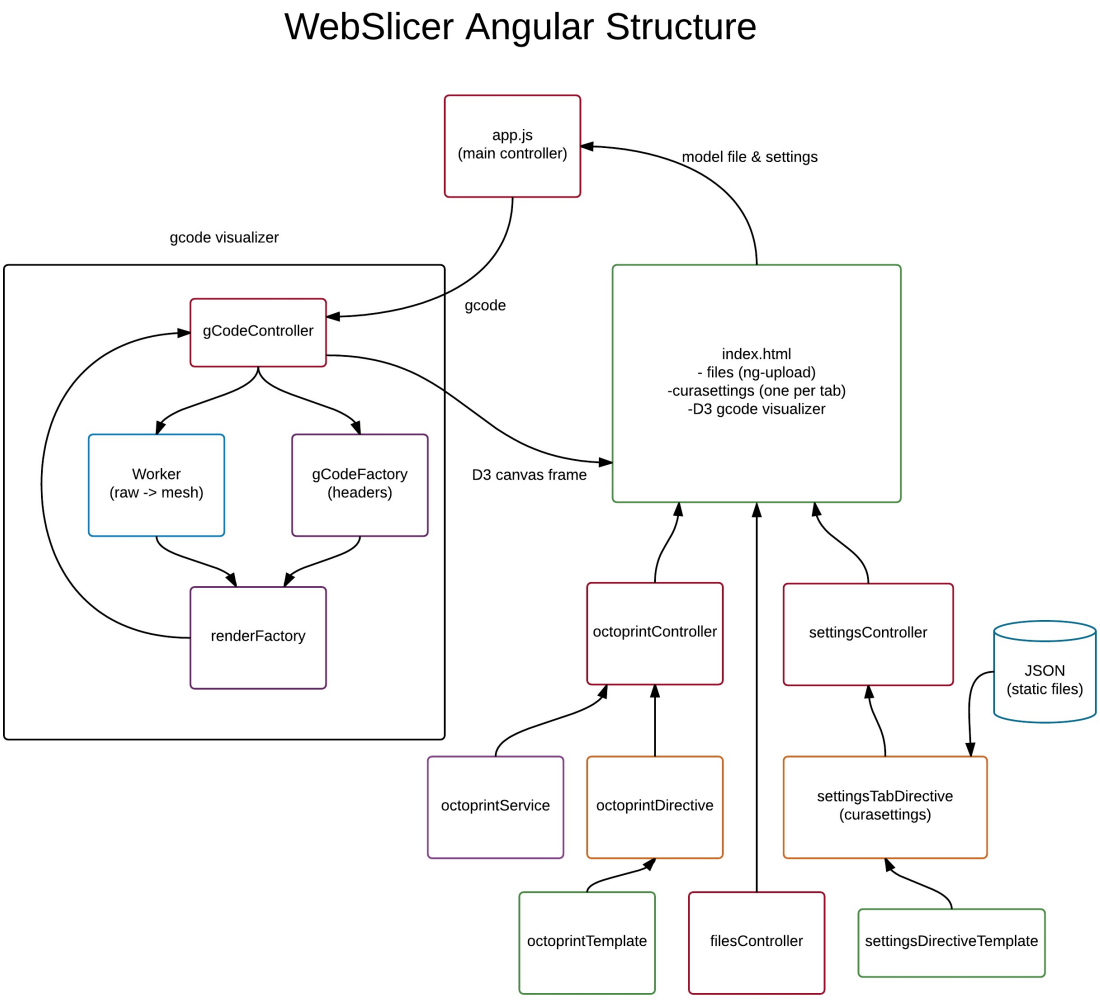


Figure 4-1: Full AngularJS structure breakdown

A directive called curasettings takes this static JSON file and splits it up so that

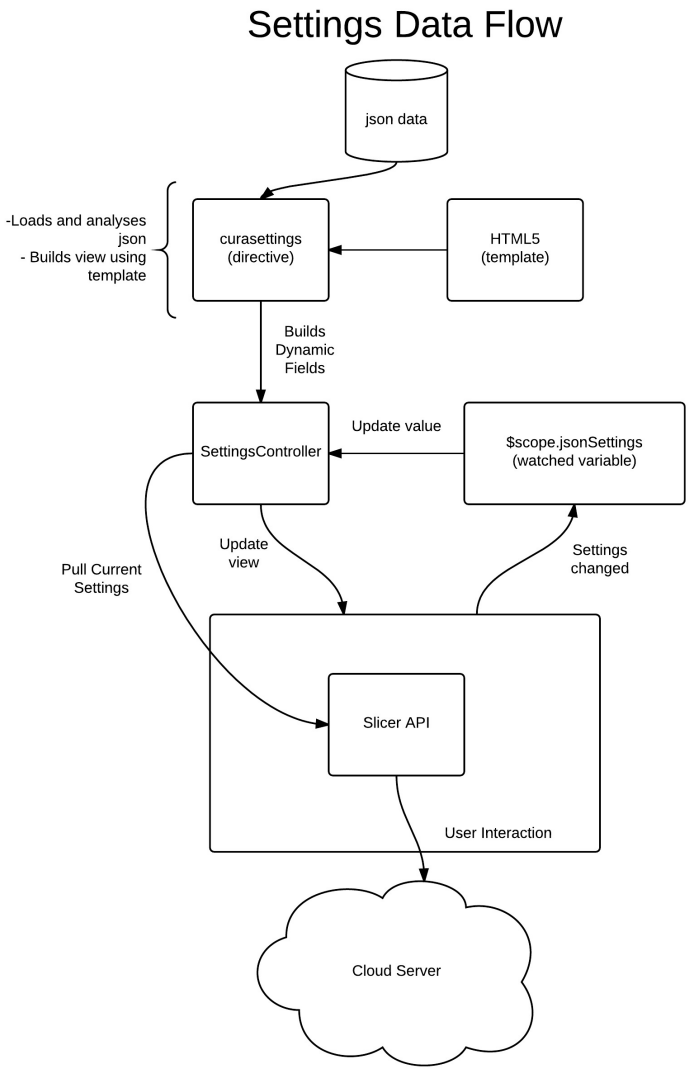


Figure 4-2: The flow of data through the application from beginning to end of client side user interaction

for each setting object a field can exist in the template. AngularJS provides this functionality through the use of an ng-repeat which is written in a similar fashion to that of a for loop in Python.

## 4.4 Key Challenges

### 4.4.1 Visualizer Integration

### 4.4.2 Interpolating Settings

## 4.5 Other Planned Integrations

## 4.6 Issues & Known Bugs

# Chapter 5

# Server Side

## 5.1 JavaEE 7

## 5.2 ProcessBuilder

## 5.3 CuraEngine Integration

## 5.4 REST API

## 5.5 Key Challenges

### 5.5.1 ProcessBuilder Deadlock

## 5.6 Issues & Known Bugs

# Chapter 6

# Discussion

## 6.1 Usability Testing

## 6.2 Data Gathering

## 6.3 Design Updates & Improvements

## 6.4 Future Work

# References

L. Columbus. "2015 Roundup Of 3D Printing Market Forecasts And Estimates." *Forbes*, 2015.

R. D'Aveni. "The 3-D Printing Revolution." *Harvard Business Review*, 2015.

A. Freeman. *Pro AngularJS*. apress, 2014. doi: ISBN-13(electronic):978-1-4302-6449-1.

N. Hitze. "gCodeViewer." January 2015.

J. Horvath. *Mastering 3D Printing*. Apress, 2014. doi: 10.1007/978-1-4842-0025-4_6.

M. Meding. "Why AngularJS is the Future of Web Design.". 2016, UMass Lowell, 2016.

P. A. Pilgrim. *Java EE 7 Developer Handbook*. Packt Publishing, 2013.