EECS 393: Software Enginnering

Software Requirements Specification

for

Android 3D Photocopier App

Authors:
Austin Hacker, Joshua Tang, Callum Grant, Alex Campbell

September 28, 2015

Contents

1 Introduction **2**
1.1 Purpose . . . 2
1.2 Project Scope . . . 3

2 Overall Description **4**
2.1 Product Perspective . . . 4
2.2 User Classes and Characteristics . . . 4
2.3 Operating Environment . . . 5
2.4 Design and Implentation Constraints . . . 5
2.5 User Documentation . . . 5
2.6 Assumptions and Dependencies . . . 5

3 Feature Description and Requirements **6**
3.1 3D Scanning . . . 6
3.2 Mapping Viewer . . . 6
3.3 Create 3D Mesh . . . 6
3.4 Create 3D Mesh from Partial Scans . . . 7
3.5 View 3D Model . . . 7
3.6 Edit 3D Model . . . 7
3.7 Importing and Adding 3D Models . . . 8
3.8 3D Printing . . . 8
3.9 Sharing 3D models . . . 8
3.10 Cloud Processing . . . 9
3.11 Local Processing . . . 9

4 Other Functional Requirements **10**
4.1 Printability Requirements . . . 10
4.2 Server Connectivity Requirements . . . 10
4.3 Storage Requirements . . . 10

5 Non-functional Requirements **10**
5.1 Performance Requirements . . . 10
5.2 Software Usability Requirements . . . 11
5.3 Connectivity Requirements . . . 11
5.4 Security Requirements . . . 11

Revision History

Name	Date	Reason for Change	Version
Austin Hacker	9/28/2015	initial draft	1.0 draft
Austin Hacker	9/28/2015	changes due to inspection reports	1.0 draft 2

1 Introduction

1.1 Purpose

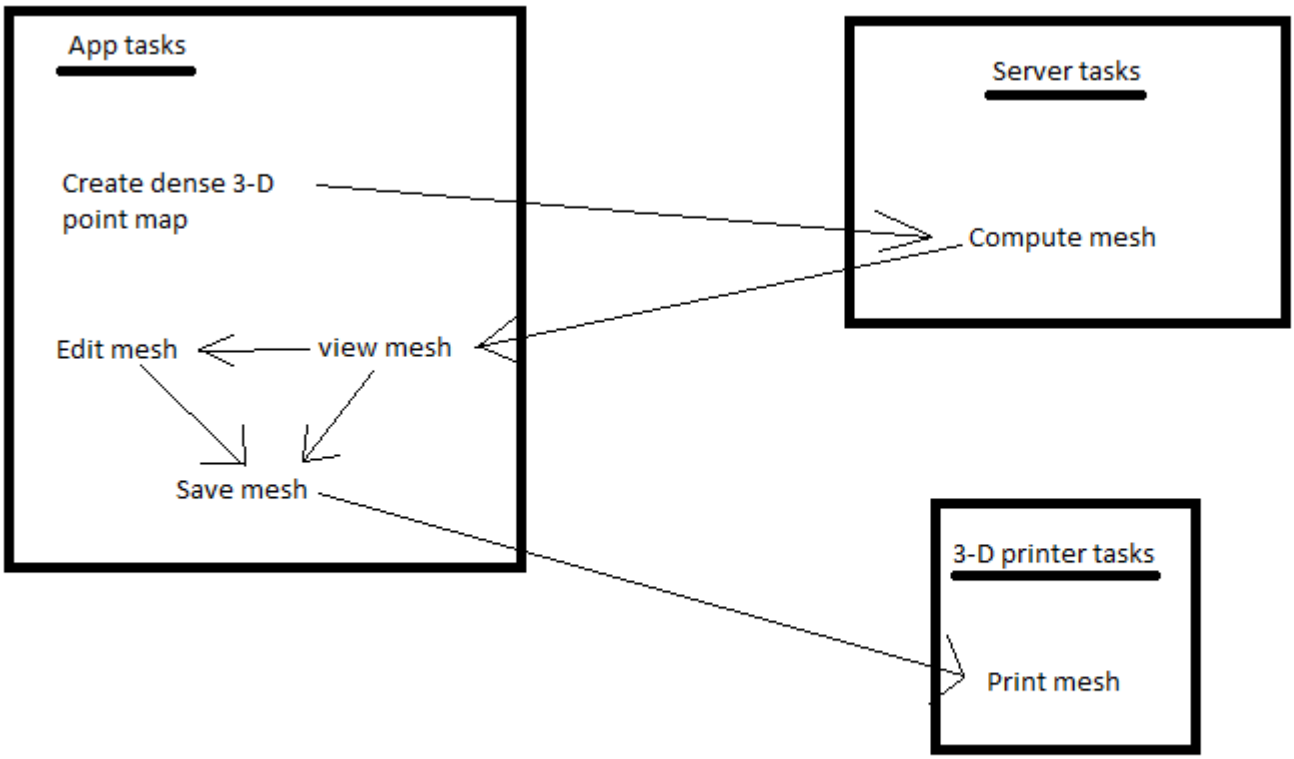
This Software Requirements Specification (SRS) describes the functional and nonfunctional requirements for the initial release of our 3D Scanner App for Android devices. This will be used as an internal guideline to ensure that all features of our product are working as intended. All specifications here are for the initial 1.0 release unless noted otherwise. Initial work will focus on the creation of a server to process a dense point map into a 3D mesh that is printable on a MakerBot 3D printer and an Android app that can remotely connect to the server to retrieve a 3D mesh and display it in a simple 3D viewer. From here, the app will be refined to include a system to create a 3D dense point map to send to the server for processing. This will constitute our first prototype design. Following that, the 3D viewer portion of the app will be redesigned to allow editing of 3D objects and the app will also include a simple system to print the 3D mesh generated on the server in a MakerBot 3D printer. This will mark our 1.0 release.

In order to ensure quick calculations of a 3D mesh from a dense point map, a Robot Operation System (ROS) Fuerte system will be run on top of a Linux Ubuntu 12.04 operating system. The tools provided by ROS will allow for simpler code to meet the harsh time line that is required of this project while the underlying Ubuntu server will handle networking for multiple users and concurrent programming to allow for all users to obtain their meshes in a timely manner rather than certain users being queued indefinitely.

This app will allow for a hole in the mobile 3D scanning marketplace to be filled. Current options for 3D scanning are incredibly slow, taking nearly an hour to compute a 3D design given a series of pictures, or are restrictive in scope, only allowing certain types of designs and aren't truly 3D scanners.

Software design is expected to progress iteratively with features being added only after previously implemented features are stable. In this way, we can physically test certain features that rely on other components in more intuitive ways. The design for this project is inherently linear, as shown in Figure 1, so this will lead to the most efficient design methodology.

Data Flow Diagram



1.2 Project Scope

The 3D Scanner App will allow for a series of features that functionally combined allow Android users to create a 3D model that can be sent to a 3D printer from a simple video of a physical object. In this way, the app allows Android users to replicate designs that they find interesting or significant at various scales.

To accomplish this, the 3D Scanner App will allow users to rotate their Android phone around a physical object and capture visual data that a 3D model is then created from.

Upon completion of the 1.0 release, this project will focus on optimizing the smoothing procedure of the 3D mesh to allow for more accurate and quicker scans. The project scope can also be expanded to include businesses that use

3D scanning and printing by establishing custom servers and support for various 3D printers. Due to the time constraints, this goals are not obtainable in the 1.0 release so the project has focused on a general application for the public.

2 Overall Description

2.1 Product Perspective

The 3D Scanner App provides an easy way for users to obtain a 3D model of a physical object. On top of creating the model after panning the camera around the object, it includes the option to save the model for usage at a later time, or transfer to another device. Currently, no Android app allows for this functionality. The closest competitor requires a series of snapshots to be taken of the object at various angles. If a picture comes out blurry, the app does not make this known until after all snapshots have been taken, at which point the app does not remember which angle the blurry picture was taken at, requiring guesswork by the user. By using a continuous video stream, this app hopes to elevate the burden placed on the user by the competition.

2.2 User Classes and Characteristics

Artist - An artist is a user who is using the app with the intent of scanning in a prototype they have made. This user will likely desire accurate results from the scanner, to match with their prototype model.

Hobbyist - A modeling hobbyist likely is using the app to allow them to make reproductions of models or self-made model parts. As they are used to construction of models and have equipment to help with such, they are less concerned with exact accuracy of models, and appreciate speed of the app.

Tourist - A tourist will likely use the app to create reproductions of interesting objects in their travels. Because they will likely not have anything that can make use of the output model files while traveling, time is less of an issue for them, and ability to save the output files is most important.

Engineer - An engineer could use this app to replicate a specific part of an existing product for a testing prototype. Due to the necessity of an exact replica to be produced to accurately test the component, accuracy is of the utmost importance with speed being a secondary factor. Due to time constraints on this project, this user is not a high priority user for the 1.0 release.

Architect - Architects typically build models of buildings to show to clients. This app would be useful to architects as they can scan their models into a digital copy to send directly to a client or to produce a mesh that they can manipulate on the app or preexisting software to create and print a new model.

Accuracy of design is a high priority for this clientle with portability to other devices and usability in other software being a secondary priority.

2.3 Operating Environment

OE - 1: The 3D Scanner App will run on mobile devices running Android operating systems.

OE - 2: The app shall use an external machine running the Robot Operating System as a remote server for data parsing.

2.4 Design and Implementation Constraints

DI - 1: The code for the app portion shall be small enough in size to fit on an Android device.

DI - 2: The server to parse data will be running Robot Operating System via C++ code.

DI - 3: The initial release must be completed within 3 months.

DI - 4: Streaming video to the server and downloading the file back to the phone may consume large quantities of user data. Design should keep in mind a way to send less to the server and back to the app.

2.5 User Documentation

UD - 1: The app will contain a help page giving an overview of how to produce a 3D model via usage of the app.

UD - 2: The app will contain simple instructions on what the user must do to complete a dense 3D point map for the server while the video is being taken to create an image without holes.

UD - 3: The github page/Android marketplace page for the app will contain a change log so that users can see what additional features or optimizations have occurred following the 1.0 release.

2.6 Assumptions and Dependencies

AS - 1: The user has Internet connection, either via a direct connection to a wireless network, or via their mobile plan.

AS - 2: The user has access to a MakerBot 3D printer.

AS - 3: The user has an Android phone with the hardware equivalent or better than a Samsung Galaxy S6.

AS - 4: The 3D Scanner App is expected to be run in an open environment where the user can easily circle the object being scanned at various angles.

AS - 5: The 3D Scanner App is expected to be run in an environment with low external interference (people walking nearby, changing levels of light, etc.).

3 Feature Description and Requirements

3.1 3D Scanning

3.1.1 Description

A user can scan an object, by walking around it with their phone's camera pointed towards the object. This will build a semi-dense point cloud by mapping out the object.

3.1.2 Priority

This item is a high priority.

3.1.3 Requirements

- When a user is satisfied with the scan, they can stop it to make a mesh.
- The user can cancel a scan at any point during the scan.

3.2 Mapping Viewer

3.2.1 Description

The user can see the scanned object being mapped in real-time.

3.2.2 Priority

This item is a mid priority.

3.2.3 Requirements

- The user receives real-time feedback from the app while scanning.
- This view can replace or complement the output from the camera.

3.3 Create 3D Mesh

3.3.1 Description

Once the user confirms a scan, the result of the scan will be used to create a 3D mesh.

3.3.2 Priority

This item is a high priority.

3.3.3 Requirements

- The server or 3D scanner portion of the app must be able to filter external interference due to passing objects in the background.
- This mesh will be saved as a common 3D object file.

3.4 Create 3D Mesh from Partial Scans

3.4.1 Description

If a scan does not complete a full loop around an object or a hole exists in the semi-dense point cloud, we will give the user the option to automatically try to model what can't be seen. Otherwise, the non scanned surfaces will simply be made flat.

3.4.2 Requirements

- The server must be able to smooth an object into a simple geometry.
- The server can flatten an object.

3.4.3 Priority

This item is a low priority.

3.5 View 3D Model

3.5.1 Description

The mesh that was generated in the scan will appear in as 3D object on the user's phone where it can be rotated and viewed from every possible perspective.

3.5.2 Priority

This item is a high priority.

3.5.3 Requirements

- The 3D viewer supports 360 rotation of a scan.
- The 3D viewer supports zoom.

3.6 Edit 3D Model

3.6.1 Description

The user can change the model that was generated by the scan on the their phone.

3.6.2 Priority

This item is a mid priority.

3.6.3 Requirements

- The 3D editor must be able to perform simple scale adjustments to allow large objects to be printable.
- The user can cut out parts of the 3D model until they are satisfied.

3.7 Importing and Adding 3D Models

3.7.1 Description

Allows user to import preexisting 3D models from the Internet and attach them to the model that they scanned.

3.7.2 Priority

This item is a very low priority. It will not be featured until after the 1.0 release.

3.7.3 Requirements

- The app can find or add compatible files located on the phone.
- The app or server can merge files together.

3.8 3D Printing

3.8.1 Description

The user can send the 3D model to a 3D printer to print a copy of it.

3.8.2 Priority

This item is a high priority.

3.8.3 Requirements

- The app can connect to a MakerBot 3D printer.
- The app can instruct the 3D printer to print a model.

3.9 Sharing 3D models

3.9.1 Description

The user can share the 3D model via email and on Social Media.

3.9.2 Priority

This item is a very low priority. This will not be worked on until after the 1.0 release.

3.9.3 Requirements

- The app can store the file on the phone in such a manner that other applications may easily find it.
- The file is small enough to fit in an email.

3.10 Cloud Processing

3.10.1 Description

The 3D scanning and the creation of the 3D mesh can be done on the cloud, allowing for a greater speed and allowing more devices to be able to use the app.

3.10.2 Priority

This item is a high priority.

3.10.3 Requirements

- The server can manage cloud computing on ROS.
- The server can easily manage many connections, making cloud computing worthwhile.

3.11 Local Processing

3.11.1 Description

Let's the user do the 3D scanning and create the 3D mesh on their device, making the app usable even without data. The is a goal behind the scope of the 1.0 release.

3.11.2 Priority

This item is a very low priority. This will not be worked on until after the 1.0 release.

3.11.3 Requirements

- Code must be optimized to allow all computations without crashing.
- Code must be small enough to fit within an Android app available on the Android Marketplace.

4 Other Functional Requirements

4.1 Printability Requirements

PR - 1: Must be able to establish a persistent connection to a MakerBot 3D printer.

PR - 2: 3D mesh must be in a format compatible with a MakerBot 3D printer.

PR - 3: Mesh must be printable on a MakerBot 3D printer.

PR - 4: The app must notify the user when an object has finished printing.

4.2 Server Connectivity Requirements

SC - 1: The server must be available at all times (dedicated server).

SC - 2: The server must have an automated recovery system to reboot after a crash.

4.3 Storage Requirements

ST - 1: The app must be able to store files received from the server on the phone.

ST - 2: The app must be able to store files received from the server on an SD card.

ST - 3: The app must be able to access stored files of the correct file type from the phone or an SD card.

ST - 4: The app must allow for files created by the server to be used by other applications.

ST - 5: The app must store the 3D mesh in a standard file type for 3D printers.

ST - 6: The 3D editor must be able to "crop" the world space created by the server.

5 Non-functional Requirements

5.1 Performance Requirements

PE - 1: The app must run reliably on a Samsung Galaxy S6 or other Android phone with equivalent or better hardware.

PE - 2: The app must be able to create a dense point map of an item to a degree of accuracy matching competition.

PE - 3: The server must compute a viable mesh for the dense point map in a time frame comparable or better than the competition.

5.2 Software Usability Requirements

SU - 1: The app contains a 3D viewer that can be rotated and zoomed in a way that is intuitive to general users.

SU - 2: The app can store a library of designs created by the user in a manner that all designs can be easily located and identified by the user.

SU - 3: The app must be able to generate a sufficiently dense 3D point map after a single circle is viewed around the object.

5.3 Connectivity Requirements

CO - 1: The app can connect to a 3D printer remotely without hassle.

CO - 2: If the app disconnects from the 3D printer while sending a design to the printer, the app will resend the file upon reconnection.

C0 - 3: If the app/server connection is broken during the processing of a dense point map into a 3D mesh by the server, the server will be able to reconnect to the user and send the 3D mesh to the app upon reconnection.

CO - 4: The app must optimally stream data to the server to prevent overconsumption of user data.

CO - 5: The app must be able to simply interface with an Ubuntu server over a wireless internet connection.

5.4 Security Requirements

SE - 1: The app must respect restrictions on use of a 3D printer (may not be used to bypass payment of use for a public 3D printer or use of a personal printer without consent of the owner).

SE - 2: The server may not store 3D designs or dense point maps after processing data and sending it back to the app user to prevent potential hackers from stealing designs.

SE - 3: The server must be able to identify each user uniquely.

SE - 4: The server may not accept any data that is not a dense point map.

SE - 5: The server will not store any personal information on users.

Individual Work Breakdown

The following people wrote the sections associated with their names.

Austin Hacker : Non-functional Requirements, Data Flow Diagram, Introduction, Other Functional Requirements

Callum Grant : Introduction and Overall Description

Josh Tang : Features Requirements

Alex Campbell : Other Functional Requirements