Automating the Creation of 3D-Printed Water Pipe Fittings
— Final Report —

Group 4: Melinda Chan, Luca Filipi, David Joyce,
Kathryn Shea, Angharad Thomas, Henry Williams
{mc2415, lf1311, dj815, ks815, alt115, hw5115}@doc.ic.ac.uk

Supervisor: Dr. Anandha Gopalan
Course: CO530/533, Imperial College London

13th May, 2016

Contents

1 Introduction

1.1 Problem

In a humanitarian disaster, logistical constraints can make it difficult to quickly source relatively simple but critical and potentially life-saving items. In response to this need, the non-profit organisation Field Ready[1] is working with a number of leading global humanitarian aid agencies (such as World Vision[2] and Oxfam[3]) to introduce on-the-ground 3D printing technologies in communities devastated by natural disasters. Recently, they have been concentrating efforts on providing water fittings to remote areas where makeshift alternatives can lead to water loss or contamination. However, the technology currently used requires a relatively high degree of technical training for local aid workers, preventing such an approach being taken on a wider scale - particularly when immediate and unforeseeable aid is required.

1.2 Solution

In order to address this problem we have created an application for Field Ready called MakeFit, which automates the design stage of the 3D-printing process. MakeFit has been designed so as to allow people with little to no training to easily select and customise basic water pipe connector parts from a simple and intuitive interface, then create the required printer files on their laptop without needing to connect to the internet. The application has been designed in a manner that can easily be built upon and adapted, so that in time it may be used to create a variety of components required by aid workers across the globe.

2 Specification

2.1 Goal-Oriented Requirement Capture

We began the project by clarifying our primary project goals and creating a long list of potential features and functionality for our product, based on a combination of research and early conversations with our client. These features were then discussed and prioritised in a kick-off workshop with Field Ready, based on two dimensions: expected impact and alignment with goals; and difficulty of implementation. The latter encompassed a number of factors, including time taken, skills required, and other resource requirements. That prioritisation was then used to identify and agree with the client a minimum viable product (MVP), as well as a number of 'priority waves' for any additional work, should time permit. These are included in Appendix E.

Mid-way through the project we held a second workshop with Field Ready, in which we demonstrated a fully functional version of the MVP. After the demonstration we held a second brainstorming session to identify any new additional features and re-prioritise further work, now that the client was better able to visualise the product functionality.

2.2 Minimum Viable Product Requirements

The following requirements were identified as being critical for a minimum viable product:

A Generate the STereoLithography (STL[4]) files required to 3D print a single pipe connector type

B Use a parametric approach to designing parts, allowing customisation by the user

C Work as a stand-alone application not requiring internet connection

D Have a simple, intuitive graphical user interface (GUI) requiring little to zero user training

[1] Field Ready: www.fieldready.org
[2] World Vision: Field Ready's prinary partner in Nepal; www.worldvision.org
[3] Oxfam: Another of Field Ready's key collaborators in their water pipe fitting project; www.oxfam.org
[4] STL (STereoLithography) is a file format native to the stereolithography CAD software created by 3D Systems

E Run on the Windows[5] operating system, given its prevalence across aid organisations

2.3 Additional Features and Functionality

Additional attributes and features selected as being beneficial, but not essential, are listed below in thematic order[6]. For more information on priorities, based on the relative importance (impact and alignment with goals) and feasibility estimates made during the second workshop, please refer to Appendix F.

Overall Functionality and Usage

- Amend the architecture to facilitate the easy addition of further parts for printing, including placeholders
- Add other parts for customisation, including O-ring moulds
- Add identifying marks or batch numbers to connectors for accountability and tracking purposes
- Create a cloud-based version in addition to the laptop application

GUI

Parameter Inputs

- Allow input in both inches and millimeters
- Create a dropdown menu on the parameter input page to include common measurements
- Add functionality to forbid the use of parameters outside a pre-designated range, and requiring confirmation of non-standard parameters

Other Functionality

- Allow users to select a subset of components for STL file generation
- Include a warning to users regarding the time taken to create STL files, and a final frame confirming completion of the download process
- Add a help page, including contact information in case of queries or feedback
- Add a quality checklist to assess printed connectors
- Add a means of indicating to users that the program is running while the STL files are being created to avoid them closing the program during execution e.g. a progress bar

Aesthetics

- Replace message alerts with responsive, colour-coded text in the main window (i.e. red text for invalid parameter inputs, and amber text for valid but non-standard inputs)
- Add 3D visualisation of individual components, potentially scaled to reflect the selected diameters

[5]Windows is the most widely used operating system in the world, developed and distributed by Microsoft

[6]Note that some of the original ideas included in Report One were later deemed unnecessary by Field Ready, and have thus been excluded from this list. In the interests of completeness, all are still included in Appendix D

3 Product Design

3.1 Functional Design and Workflow

As per the specification, the guiding principle of our product design was ease of use. In order to achieve this, we aimed to keep the basic flow of control as simple as possible, minimising the number of frames and the links between them. Figure 1 shows our original wireframe outline, comprising just four frames: a menu page to select a part to create, a page on which the user specified the dimensions for the required part, a page to pick a folder to save the resulting files in, and a page to let the user know when the STL file creation process was complete.

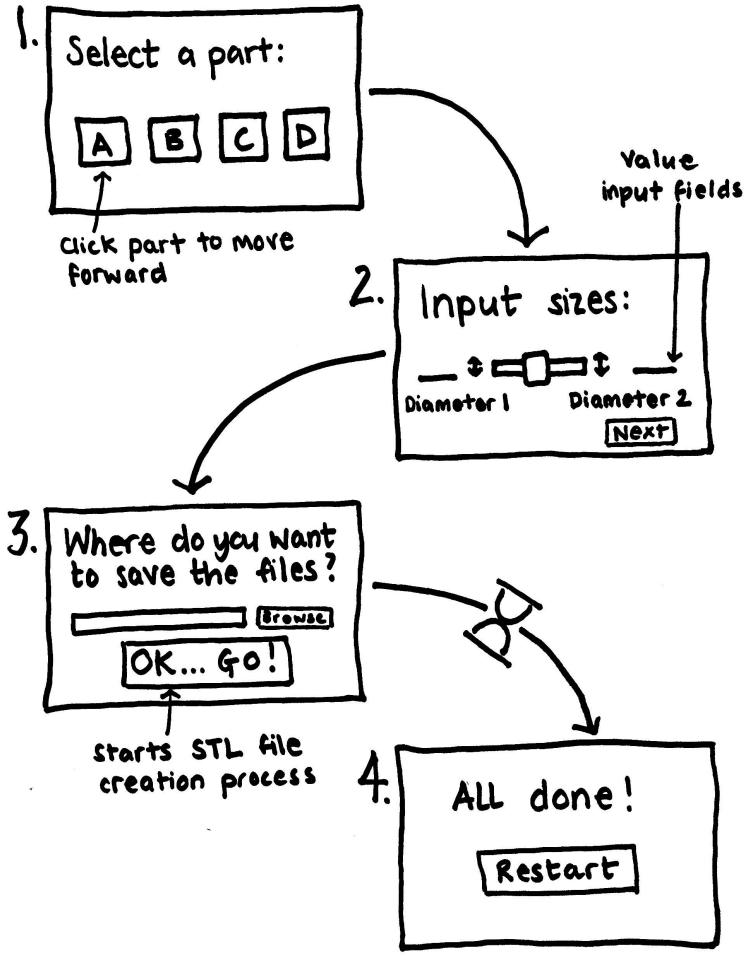


Figure 1: Original Wireframe Outline

The final product is largely based on this work-flow, with some additions and modifications:

1. Added a welcome page
2. Added a help page, which can be reached by clicking a button in the top right of each screen
3. Added a holding page which opens when the file creation process is initiated, to inform the user that the download will take a few minutes
4. Extended the functionality of the frame in which the user selects a folder to download to, to also allow the user to select a subset of components to download

3.2 GUI Design

From an aesthetic perspective, our mandate for the product was two-fold: make it clear and succinct, minimising text and using images where possible and make it attractive to encourage user engagement. We decided to base our colour schemes, fonts and images on Field Ready's website to ensure coordination with their existing products. However, we separated out all formatting definitions into a header file using named variables so that these could be updated easily if required at a later stage. This proved useful when, towards the end of the project, the client updated their branding colour palette.

Figure 3 shows a screenshot of the Main Menu page in the final product, corresponding to the first frame of our original wireframe outline. The first and second buttons, for a straight coupler fitting and an O-ring mould respectively, take the user to the corresponding Parameter Input pages. At present, the other two buttons do not invoke any commands, and are included as placeholders for future fittings as requested by the client.

Figure 3 is a screenshot of the Parameter Input page for the 'straight coupler' fitting, allowing the user to either select standard measurements from the drop-down menus or input a custom value. All inputs can be specified in either inches or millimetres. Depending on the values input, the program may display a warning message underneath the relevant input box. There are three types of messages: warnings that the values input are invalid (e.g. non-numeric characters or blank entries); warnings that the values are outside the printable range; and warnings that the values are printable but outside the range expected. The user is not allowed to proceed to the next screen if either of the first two warnings are being displayed.

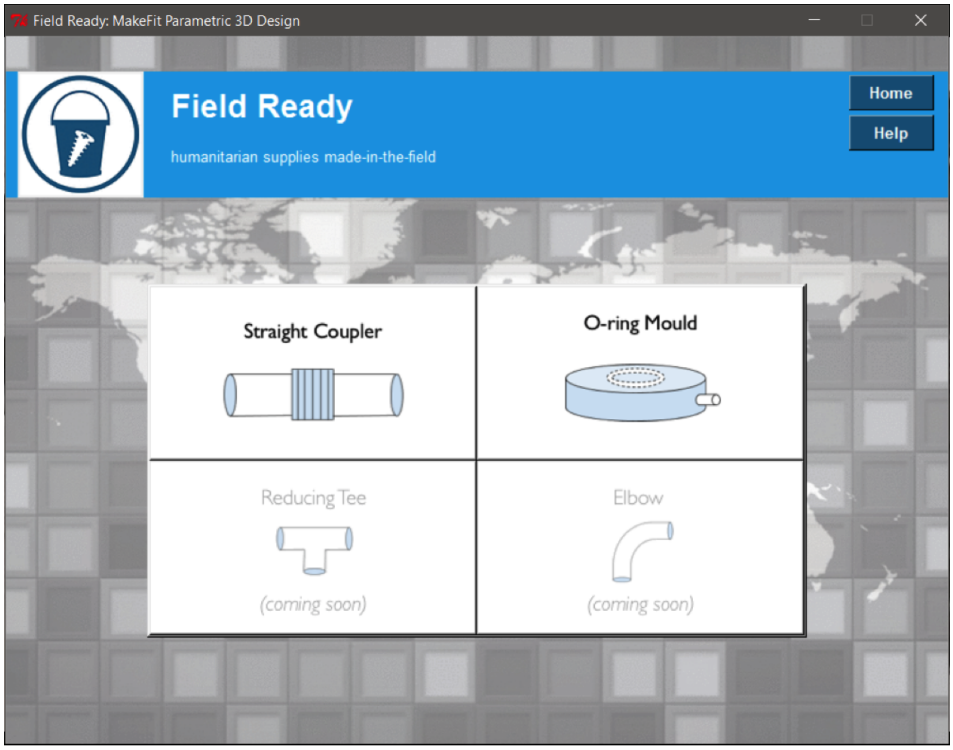


Figure 2: Main Menu Page

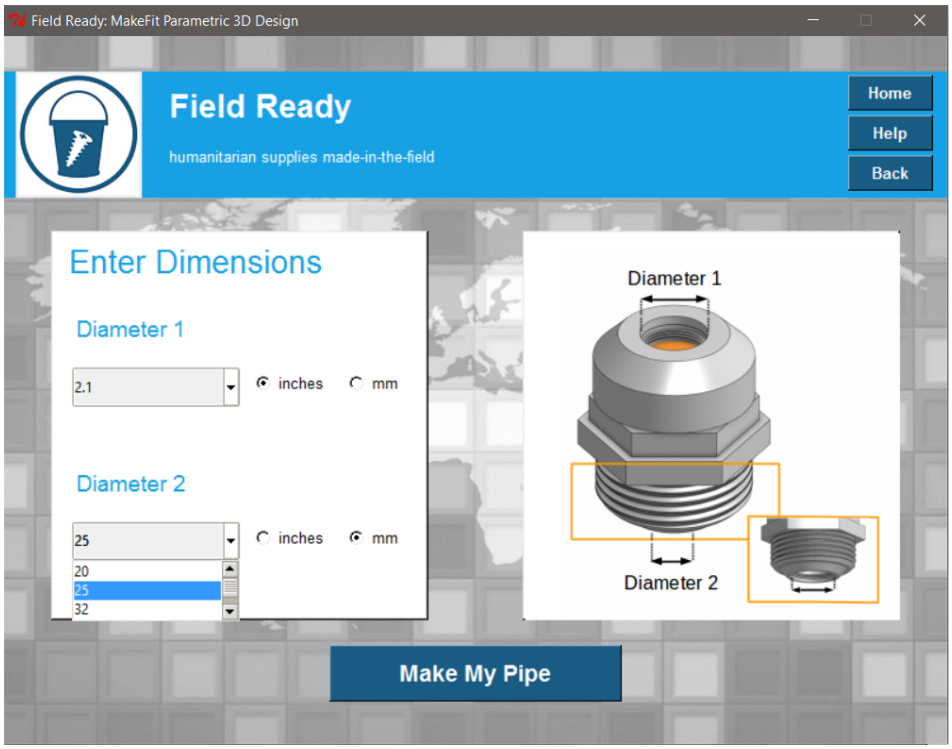


Figure 3: Parameter Input Page for Straight Coupler, Displaying Example Warning

4 Technical Solutions

When determining which language to use for our various modules, we took into account a number of factors, including: the availability of existing modules to simplify programming and testing; accessibility, both for us as relatively inexperienced programmers and for other developers who may need to work on our code at a later date; and compatibility with the intended user environment. We opted to use Python[7] as far as possible, on account of its popularity across the programming community, expansive library of add-on modules, and the widespread availability of supporting resources. In addition, by sticking to one primary language throughout the product, we avoided any compatibility issues which may have arisen had, for example, the front-end and back-end been created in different paradigms.

4.1 Front-End

The GUI was created using Python's Tkinter[8] module. While we could arguably have achieved a better 'look' using wxPython[9], another common Python-based GUI tool, we chose Tkinter primarily because it is shipped with Python. Additionally, wxPython requires a separate download and would thus increase the size of the installer package. Our research also indicated that Tkinter's widgets and geometry manager are easier to work with, with more documentation available to aid our development.

The interface works by first creating a frame for each 'page' in the program's wireframe, all of which run for the entire time the program is running. Events triggered by the user, for example when clicking a button, will call event handling functions to update what the user sees - be that changing the frame, updating text, or displaying a pop-up message box.

Within each frame we have used a combination of text, image, and input widgets, including radial

[7] A high-level computer programming language
[8] A Python module binding to the Tk GUI toolkit
[9] A GUI toolkit that extends on Python

buttons and drop down menus. These were selected and amended based on client feedback.

In the interest of clarity, we have created separate implementation files for each frame which are then navigated through via the master page in which the event loop runs. In accordance with standard programming practice, the majority of repeated code is done through helper functions (for example, to add a home button at the top of most frames). To support further development efforts, the code has been clearly labeled throughout including explicit instructions for what to amend or copy in order to add a new part. Most of the key variables, such as styling and size limits are easily modified throughout the interface via the distinct *variables.py* file.

While we were ultimately able to achieve all of our goals, Tkinter did provide a few obstacles - such as freezing the entire interface while one event handler processes. These are discussed in more detail in the final section of this document, where we have listed the primary challenges we encountered and the measures we took to overcome them.

4.2 Back-End

The back-end modules comprise everything needed to take the user-provided inputs passed from GUI functions and generate the required STL files. They were built using a mixture of Python and OpenSCAD[10], an open-source application which can be used to specify and compile 3D files. We opted to use OpenSCAD because, unlike more advanced Computer-Aided Design (CAD) software, it is free, compact to download, uses an easy-to-use textual description language, performs parametric scaling, and is compatible with Python.

The primary function for each part, found in each part's respective Python file (e.g. *o_ring_mould.py*), is called by the GUI which passes through a number of parameters, including user-input parameters, an array indicating which components are required, and details of where to save the resulting STL files. The part's respective Python file then opens the respective OpenSCAD file through the command-line, passing the necssary variables.

OpenSCAD files use a number of existing STL files in order to make the customised parts. These files were provided by Field Ready engineers, and have undergone rigorous testing to ensure their suitability for the task before scaling. The OpenSCAD script then scales the components of the part according to the command-line variables and creates any additional 'connector' shapes required to bind the scaled components. Finally, the script joins these components together to produce the customised part and saves it as an STL in the designated location.

This approach was not included in our original plan, which had been to create the parts from scratch in Python using a SolidPython[11] library which then compiled to OpenSCAD code. The models created by Field Ready were used as a template only. This task proved to be considerably more challenging than we anticipated and would have required an additional round of mechanical testing to ensure that the parts we were making were fit for purpose.

We therefore changed our plans half-way through the project choosing to use OpenSCAD instead of SolidPython for the parametric resizing. An added advantage of our new method is that it will be easier for Field Ready to add new parts to MakeFit, as the modifications required to make our code suitable for adding new 'base' STL files are less time intensive than having to create those parts in the SolidPython from scratch. The Python scripts for each part are nearly identical, with only minor changes required when adding a new part. The OpenSCAD scripts will have to be customly created for each part. The difficulty of the task depends greatly on the design of the part being scaled.

[10] Software for creating 3D-printable CAD files

[11] A python extension for solid modelling that saves as OpenSCAD files

4.3 Installer and Integration

Packaging MakeFit began by compiling the software into Windows executables using py2exe[12], a Python extension which builds standalone Windows executable files from Python scripts. By using this, we avoided the need for the user to have or install Python. Additionally, Python doesn't need to be included in the software package either, making the overall size much smaller. This was particularly important since the client needed the package to be as small as possible.

The client also required MakeFit to be a standalone application that didn't require any internet access, even for installation. We therefore created a standalone installer using InnoSetup[13], packaging together MakeFit and OpenSCAD. It takes a Pascal[14] script and creates a Windows executable for installing the software.

InnoSetup was selected primarily because it is free, looks professional, and is easy to use. There are other installers that could have been used such as NSIS[15] but after researching the differences, users stated in online reviews that InnoSetup was easier to use and user-friendliness was an important goal for our end product.

In addition to being small (the compressed package is 15 MB, versus 39 MB if we had simply zipped the program) and easy to set up, by using a setup script we were able to add an icon to the user's desktop allowing them to easily locate the MakeFit program.

4.4 Challenges and Changes

To (mis)quote the former US Secretary of Defense Donald Rumsfeld,

"There are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns the ones we don't know we don't know. And... it is the latter category that tend to be the difficult ones."

At the start of this project, our team were aware that we were undertaking a task which would provide a number of challenges - particularly given that the majority of the work was to be done using languages and libraries with which we had little or no experience. As expected, some of the features you see in the final product are the result of many hours of research and follow multiple unsuccessful attempts to use alternative methodologies. Fortunately we were able to overcome all of those "known unknowns".

More difficult to overcome were the "unknown unknowns", the challenges which we had not anticipated. Some of these are listed below, along with the steps we took to address them.

Challenge 1: Client-driven changes to required outputs

When we started the project and set our initial milestones, we anticipated using the simple, 3-component water pipe connector which Field Ready had supplied. Two weeks into the project, just as we were starting to create the parts, the client decided that an alternative design was required, based on mechanical tests run by their engineers. Unfortunately the updated version was considerably more complex than the first, both in terms of the number of components required (seven) and the geometries of those components.

As a result, we were forced to extend our timelines for producing the part. To avoid this having a

[12] A Python extension that converts scripts to Windows executables
[13] A software package for making user-friendly Windows installers
[14] A procedural programming language
[15] Software that creates installers for Windows applications

negative impact on our overall product and in particular our goal of having a MVP to showcase to the client in our interim workshop in March, we re-distributed and re-prioritised all of the back end and integration tasks across the back end team. This fact also influenced our decision to move away from pair programming, affording us more flexibility with resource allocation.

Challenge 2: Difficulties creating the 3D parts

Initially, we had planned to create the STL files required for the connectors from scratch based on prototype files provided by Field Ready engineers. During the first workshop, Field Ready's manufacturing engineer Mark Mellors expressed concern that this could result in structurally different parts, rendering the extensive mechanical testing they had done when designing those prototypes redundant. In addition, at that point in time the parts we had created still had a number of significant issues. There were problems present in the STL which would have proved the STL to be unprintable.

We therefore decided to pause development on those files and investigate options which might allow us to manipulate Field Ready's STL files, rather than create our own from scratch. It was found that the STL file could be split into separated parts, scaled, and then merged back together. This alternative was not without its complications - for example, we needed to create additional connectors to join the scaled parts together - but these were more manageable than our original plan. This is therefore the approach which was used in our final product.

Challenge 3: GUI freezing

In Tkinter, programs run on an asynchronous event processing loop. All responsive programming, such as changing a frame or updating text colour, must be linked to user-triggered events (using lambda functions). This all happens on a single thread of execution, meaning that if one event handler takes a long time to run - such as the creation of the STL files in our program - the entire user interface freezes.

To avoid this being perceived as a problematic 'crash', we added an extra frame to be opened before the function to create the STL files was called, which informs the user that the files would take a few minutes to create and that the screen would update upon completion.

An alternative solution would be to consider creating a new thread to execute the STL file creation while the current thread maintains a progress bar to inform the client of the status. We did not take this approach, since it required a higher complexity which required extensive testing and was deemed to be low priority.

Challenge 4: Uninstallation of OpenSCAD

Initially, we agreed that the optimal way to avoid issues arising from a user trying to run MakeFit without OpenSCAD would be to 'roll back' the installer if the user opted not to install OpenSCAD. However, while running the manual stress tests we identified another potential scenario - namely that the user could uninstall OpenSCAD after successfully installing MakeFit and then try to run the program.

In response to this concern, we adapted our installer package so that OpenSCAD is installed inside the MakeFit software package (as opposed to elsewhere on the user's computer). In addition to stopping the user accidentally deleting OpenSCAD, there was a quicker file creation since MakeFit no longer had to search for OpenSCAD. This modification came at a cost to the installer file size, but was deemed an acceptable trade off.

An alternative solution considered was to have MakeFit check that OpenSCAD was installed upon

startup. However, we determined this option to be too time intensive since it would involve a full search of the users hard drive.

Challenge 5: An unresponsive client

While Field Ready did ultimately provide us with all the critical inputs we needed, throughout the project they proved to be relatively slow to respond to our requests.

We overcame this challenge in a number of ways. Firstly, when inputs that were needed to inform development work were outstanding, we chose to use our own judgement, making any updates required at a later stage. Secondly, rather than add more parts to MakeFit as we had hoped to do, given that we did not receive the required input files until a few days before our project deadline, we focused instead on incorporating other additional features. Finally, to compensate for Field Ready being unable to put together a representative group of target end users to test MakeFit, we put together a document explaining the context for its use and recruited coursemates, friends and family to download the program and provide feedback via an online survey. The situation brief used is included in Appendix H, along with a summary of the survey's results.

5 Approach

5.1 Project Structuring and Planning

As shown in the simplified project work plan below in Figure 4, we decided to split our project timeline into three main phases. The first three weeks were dedicated to research and preparation, including a kick-off workshop with Field Ready to establish product objectives and priorities. During this phase we decided what technical solutions we wanted to implement, created our work plan, including key activities and milestones. We also agreed how we wanted to work together over the course of the project.

During the second phase, which lasted four weeks, we designed the program and built a preliminary version of the MVP to showcase to the client. The third phase was then dedicated to addressing feedback, making additional enhancements and creating handover documentation. Testing took place throughout the second and third phases, and is detailed further in section 6.

Initially, we split the development into three sections, namely front-end, back-end, and integration/installation, each worked on by a team of two. It quickly became apparent, however, that the amount of time needed to design and build the GUI was much greater than the other two modules. We therefore restructured our teams into two groups of three, with Angharad Thomas, Henry Williams and Melinda Chan working on the GUI; and Kathryn Shea, David Joyce and Luca Filipi working on everything else.

5.2 Software Development Strategy

We employed an agile and iterative approach to developing the product, borrowing elements from the Scrum[16] and Extreme Programming[17] methodologies. The work plan was designed in such a way that we would have an MVP as soon as possible, so feedback received could be incorporated in later iterations.

As previously mentioned, we started development in three self-directed pairs, based on Extreme Programming's "pair programming approach", and anticipated each team would share their code with a different team to aid with reviewing and testing each iteration. In reality, however, we found that time constraints and schedule conflicts made pair programming difficult; we also wanted to continue

[16] An iterative and incremental agile software development methodology for managing product development
[17] A software development methodology which encourages short development cycles to improve productivity

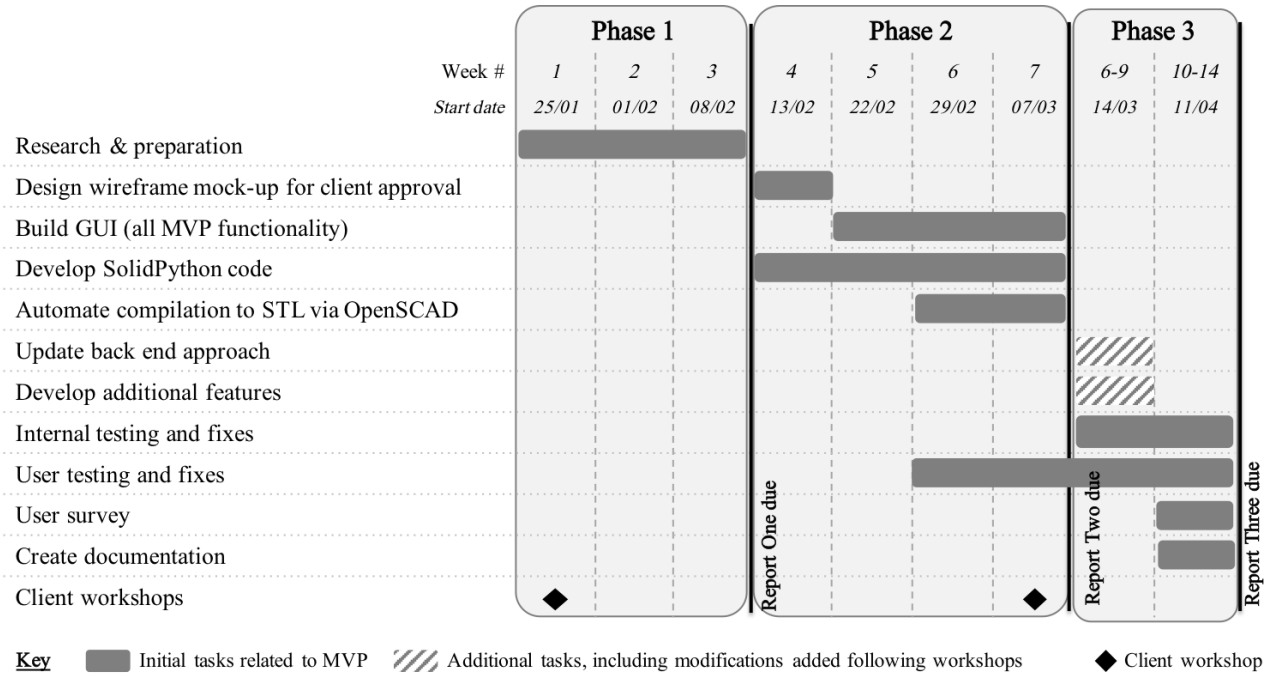


Figure 4: High-level Project Plan

working through the Easter holiday whilst being based in different countries. As a result, we switched to an approach which alternated individual work and team reviews. This also allowed us to split into two groups of three, which represented a fairer division of workload.

5.3 Internal Meetings and Governance

Throughout the majority of the project we held weekly Scrum-type 'briefing' meetings, during which each coding team discussed three key points: progress since the last briefing meeting; tasks to work on before the next briefing; and any issues, roadblocks, or new/escalated risks. As deadlines approached so too did the frequency of these meetings, up to daily in advance of the MVP and final product demonstrations.

In addition, we held a more detailed meeting with our supervisor each week during term times, akin to a Scrum 'sprint review'. In these sessions we reviewed the past weeks progress, including lessons learnt (both technical and related to how we are working as a team); revisited the action log and project work plan, making adjustments as required; and periodically reviewed the project risks and mitigating actions.

5.4 Client Interaction

Initially we hoped to hold four workshops with the client and anticipated regular Skype[18] conference calls with Abi Bush[19], Field Ready's representative in Nepal. The first two went ahead as planned on 26 January and 9 March, with Field Ready's CEO Andrew Lamb[20] present and Manufacturing Engineer Mark Mellors[21] joining by phone. Both are also intending to join us for our final presentation.

The third and fourth workshops did not take place, however, partly as a result of Andrew moving abroad mid-way through the project and partly because Field Ready felt confident that we would be able to deliver the product they desired without the need for further demonstrations. As a result, all

[18] Software that allows instant messaging and voice/video calls
[19] abi.bush@fieldready.org
[20] andrew.lamb@fieldready.org
[21] mark.mellors@fieldready.org

feedback on the updated iterations of the program since 9 March has been provided by email.

5.5 Documentation and Project Management

Throughout the project we used two key programs to plan and prioritise tasks: Asana[22], a web and mobile project management application; and InstaGantt[23], an add-on to Asana which converts project plans into charts plotting activities against timelines. Similar to a Scrum backlog, each task included a priority level (reflected in its order in the work plan), due date, responsible party, a space for comments, and a log recording any changes (e.g. changes to due dates) to aid tracking.

We created a GitHub[24] repository to store our code, taking advantage of its version control functionality to avoid conflict errors. We also used a Google Drive[25] folder to hold all documentation and research, including a meeting minutes and an actions log maintained by documentation editor Henry Williams. Each team member was responsible for maintaining an individual log tracking their time spent working on the project, including the specific tasks worked on. Logbooks are included in Appendix A, with meeting records and workshop minutes in Appendices B and C.

6 Testing Methodology

6.1 Overall Plan

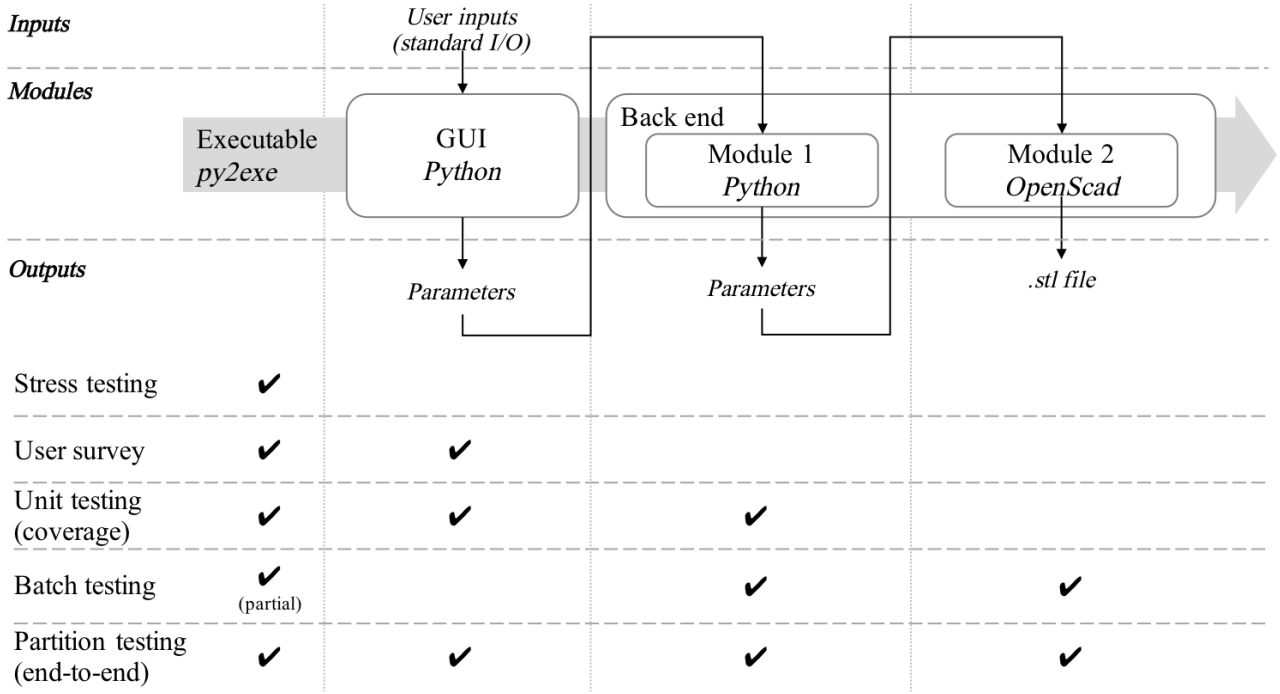


Figure 5: Summary of Testing Techniques

In order to ensure our product is both fully functional and well suited to its purpose, we designed a series of different verification and validation tests to be undertaken by the team, the client, and a representative sample of potential end users. Most of the internal testing was based on a White-box[26]

[22] https://asana.com/
[23] https://instagantt.com/
[24] GitHub is a web-based Git repository hosting service. https://github.com/
[25] https://google.co.uk/drive/
[26] White-box testing is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality

approach, whereas all external testing required no knowledge of the code and is thus classifiable as Black-box testing[27]. A summary of the techniques used is shown in Figure 5.

In addition to the more comprehensive testing detailed below, throughout Phase 2 each pair iteratively combined development work with a mix of static analysis, basic unit testing and debugging.

6.2 Stress-testing the Installer

In order to check the robustness of the installer, we manually stress-tested the installer against a list of all the possible install scenarios (see Appendix G) in an iterative cycle. We worked with the client to determine which of these potential scenarios were realistic and/or serious enough to require a fix, as well as what those fixes ought to be.

6.3 Survey

The primary tool used for validation was a user survey focusing on the program's usefulness, ease of use and overall appeal, an example of a black-box technique as no knowledge of the program's workings is required. We had hoped to use the survey to get feedback from a sample set of potential users as well as from our client; however unfortunately Field Ready were unable to arrange this in time. Instead, we sent the program and survey to friends and family, along with a 'situation brief' explaining the context in which the tool was intended to be used. Figure 6 shows a summary of the results, based on 24 responses. The values shown represent the average score given for each question, selected from a scale of 1 to 5 where 1 was the least desirable and 5 the most desirable score. A more detailed breakdown of the results, including selected comments and the exact questions asked, is included alongside the situation brief in Appendix H.

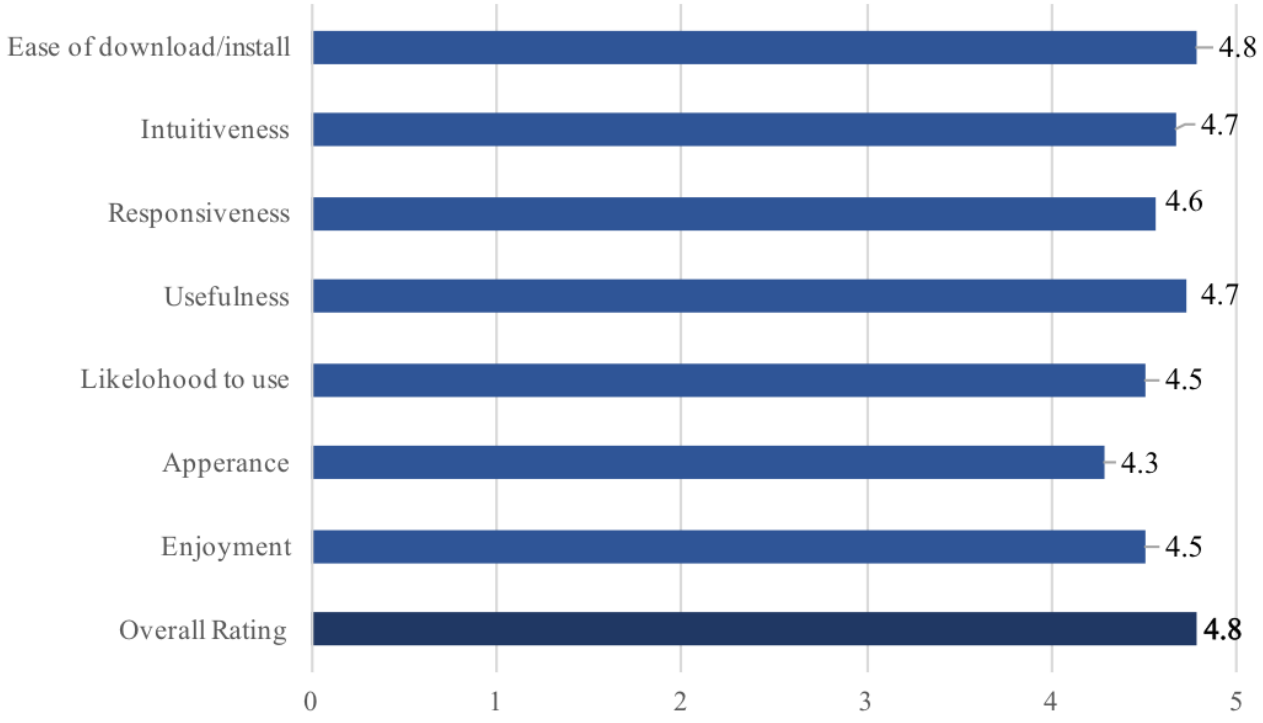


Figure 6: Survey Results: Summary of Average Survey Scores

[27]Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance.

Overall the responses were very positive across all categories, with an average 'Overall Rating' of 4.8. The question regarding the intuitiveness of the program was particularly important given our brief of making the program as easy-to-use as possible, and scored 4.7. The lowest average mark was for 'Attractiveness', at 4.3, which according to the comments and suggested improvements provided was a result of the GUI being 'old looking'. This feedback did not come a surprise to us; however our ability to make significant improvements was curtailed by the limitations of the language we had chosen to develop the GUI (Tkinter). We did however make one slight change based on the feedback provided, namely to change the colour of the buttons from orange to a more subtle navy blue.

6.4 Unit Testing

We used Python's unittest module to write and run unit tests for *functions.py*, which contains helper functions for the front-end (main GUI) and back-end Module 1. The main GUI is made up of several scripts, such as *DownloadingPage.py*, *EndPage.py* etc. more of the pages used in the front-end are shown in Figure 7, the coverage report. The Back-end module mainly consists of *screwthread.py* and *o_ring_mould.py*. More detail on the unit test script is included in Appendix J, alongside information on Partition Testing. Additionally, we have used Python's coverage module tool, *Coverage.py*[28], to measure the coverage of our tests. This report also includes the different frames in our GUI, where we have manually tested the buttons.

Coverage report: 96%

Module	*statements*	*missing*	*excluded*	*coverage*
DownloadingPage.py	61	6	0	90%
EndPage.py	40	1	0	98%
HelpPage.py	40	2	0	95%
MenuPage.py	29	0	0	100%
ORMain.py	81	3	0	96%
SCDownload.py	103	4	0	96%
SCMain.py	97	2	0	98%
StartPage.py	26	0	0	100%
functions.py	147	0	0	100%
gui.py	55	0	0	100%
mbox.py	25	0	0	100%
o_ring_mould.py	31	1	0	97%
screwthread.py	48	12	0	75%
variables.py	66	0	0	100%
Total	**849**	**31**	**0**	**96%**

Figure 7: Coverage.py summary report

Code coverage results are shown in Figure 7, which excludes coverage of third party libraries used by the application. We have tried to remove any redundant lines of code, thus resulting in a high coverage levels.

[28]Coverage.py is a tool for measuring code coverage of Python programs. It monitors your program, noting which parts of the code have been executed, then analyses the source to identify code that could have been executed but was not

6.5 Custom Batch Testing

In order to verify that all the back-end components integrate as expected, and that the final STL files generated are indeed 3D printable, we designed a process to create and validate batches of parts with different input parameters (refer to Appendix I). The main steps we took are as follows:

1. Create a python program to loop through the full back-end process, from taking input values across the whole permitted range through to saving the final STL files for each combination of input values
2. Check each STL file with NetFabb [29] to make sure it is printable
3. Check a subset of the newly generated files with specific input against verified outputs

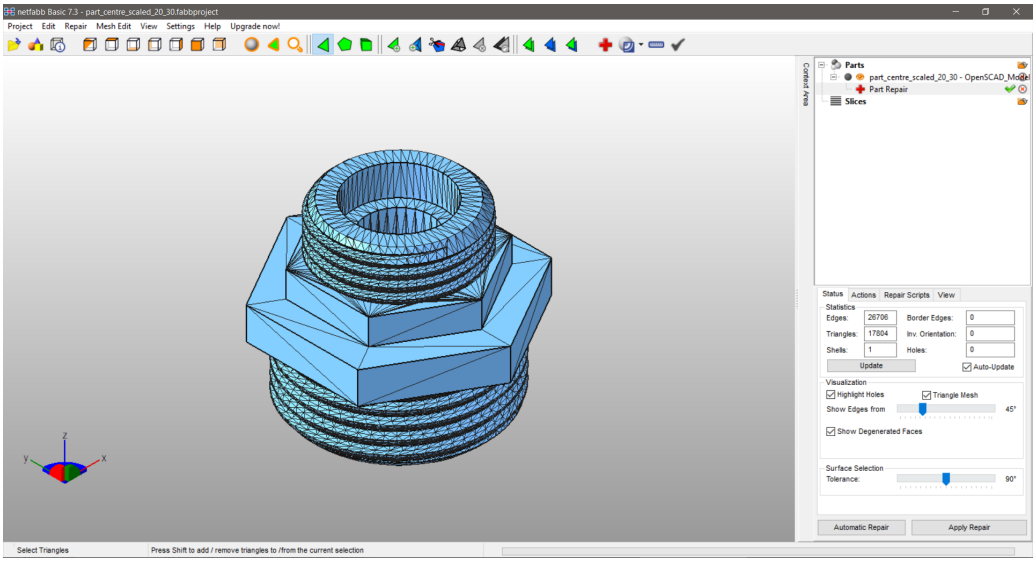


Figure 8: Custom Batch Testing a component using NetFabb

This test method can be classified as Grey-box[30], as while some understanding of the code base was needed to initiate the tests and input the parameters, comprehension of each module was not required. It constituted both verification and validation testing, as it checked both correctness of functionality and confirmed that the end product of the system will be 3D printable.

6.6 End-to-End Partition Testing

While the customised batch testing used for the back end could test for some high-level functional errors, it could not validate whether the parts generated will successfully join two water pipes. As a result, our final planned validation tests involved Field Ready users using the program to generate STL files and then print them.

The test scripts created (see Appendix J) used a partitioned approach, with the input parameters including combinations of parameters which should not be accepted by the program as well as combinations of the most extreme parameter values allowed. The scripts also covered all combinations of button presses / navigation routes through the program.

Unfortunately, Field Ready were unable to provide the resources to run these tests within the project time lines. Instead, we ran the majority of the tests ourselves, up to the point of actually printing the

[29] www.netfabb.com, NetFabb is a tool that checks for errors in STL files that could prevent them from being printed. These errors include: inverted surface normals, non-manifold edges and holes in the mesh

[30] Grey-box testing is a combination of white-box testing and black-box testing. The aim of this testing is to search for the defects if any due to improper structure or improper usage of applications.

files. In addition, we printed one set of components for the Straight Coupler, and at the time of writing Field Ready have printed two more of different sizes. All have passed basic tests (including fitting together and being watertight), although Field Ready intend to conduct further testing including the use of high water pressures before the program is deployed.

7 Group Work

As mentioned previously, the team was split into two key work streams, namely front-end and back-end/integration.

The front-end team consisted of Henry Williams, Angharad Thomas, and Melinda Chan. Henry and Angharad took the lead on development, with Melinda taking responsibility for the associated testing work (unit tests and surveys) as well as contributing to the design process.

Back-end development work was carried out by David Joyce, Katie Shea and Luca Filipi, taking the lead on creating the output STL files, packaging modules into a single executable and testing outputs respectively.

Angharad Thomas took the role of Project Leader, ultimately responsible for ensuring successful and timely delivery of both the product and documentation elements of the project through developing and overseeing workplans, leading meetings, liaising with the client and owning all written reports. She was assisted on all documentation aspects by documentation editor Henry Williams.

In addition to the above, every team member contributed towards research work, the identification of suitable methodologies to achieve our functional goals, testing, and report writing.

8 Final Product

8.1 Comparison Against Specification

Requirements	**Status**
Generate the STL files required to 3D print a single water pipe connector type	Achieved
Use a parametric approach to designing parts, allowing customisation by the user	Achieved
Work as a stand-alone application not requiring internet connection	Achieved
Have a simple, intuitive and graphical user interface requiring little to no user training	Achieved

Table 1: Minimum Viable Product Requirements and Status Update

As indicated in Table 1, we achieved all of the requirements determined as essential at the project offset. While the first three - namely generate the STL files to 3D print a water pipe connector, use

a parametric approach, and work as a stand-alone application without requiring internet connection - are all objective targets, the final requirement of being simple and intuitive is more subjective and thus its achievement harder to determine. Our rating of 'achieved' is based on the feedback we received from Field Ready and our set of independent testers, as the program received an average 'intuitiveness' rating of 4.7/5. To address the shortfall represented by the 0.3 marks we lost, we have created a user guide which can be downloaded from the programs help page.

In addition, we implemented a great deal of the additional features and functionality identified as nice-to-have at the project start, listed in Table 2. Of the nine low-priority features which we did not manage to implement, listed in Table 3, one was not possible following the scope revision to directly edit STL files; one required inputs from the client which were not achievable before the deadlines we supplied; and four were strictly out of scope as they would have required internet connection (against the brief, which was to create a program that worked without internet).

Module	Additional features and functionality	Comments
All	Structure architecture to facilitate the easy addition of further parts for printing, including placeholders	Complete
All	Add O-ring moulds	Complete
All	Allow users to select subset of components for STL generation	Complete
GUI	Text to warn users about time taken for file creation	Complete
GUI	Use screenshots of CAD 3D images	Complete
GUI	Allow inputs in both mm and inches	Complete
GUI	Images for each individual component on final page	Complete
GUI	Include drop down menu with standard measurements	Complete
GUI	Prevent users from providing parameter values outside a pre-designated range	Complete
GUI	Add warnings if users input printable but non-standard values	Complete
GUI	Require users to review and confirm their values before creating STL files	Complete
GUI	Help page (to be populated by client)	Frame added, awaiting content from Field Ready
GUI	Quality checklist (to be populated by client)	Placeholder added, awaiting final checklist document from Field Ready

Table 2: Additional Features and Functionality Included in Final Product

Module	Additional features and functionality	Comments
All	Allow users to directly edit CAD files	Not relevant or possible using revised approach of directly editing STL files
All	Create additional parts	Not done due to low priority; input files received too late to be incorporated and tested before project deadline
All	Create versions for other types of items	Not done due to low priority
All	Add batch numbers and maintain a log of parts made	Not done due to low priority; would require an internet connection
All	Create a cloud-based alternative in addition to laptop application	Not done due to low priority; would require an internet connection
All	Commenting functionality	Not done due to low priority
All	Error reporting functionality	Not done due to low priority; would require an internet connection
GUI	Links to tutorials	Not done due to low priority; would require an internet connection
Non-technical	Investigate options for licencing and potentially commercialising the product	Not done due to low priority

Table 3: Additional Features and Functionality Not Included in Final Product

8.2 Feedback

All the feedback we have received from Field Ready has been incredibly positive, as illustrated by Figure 9 which includes some direct quotes received over email to the group. This includes the news that they plan to showcase our product at the inaugural World Humanitarian Summit in Istanbul on 23-24 May 2016, during which governments, humanitarian organizations and private sector partners will debate solutions to the worlds most pressing challenges under the chairmanship of UN Secretary-General Ban Ki-moon.

> *In short – we really really like what you've made here! Thank you!!! Well done to all of you. We're so very glad that this has worked out so well! Much appreciated.*
>
> *We'd like to show this off at the World Humanitarian Summit in Istanbul later this month. We'll let you know if Ban Ki Moon decides to play with it…*
>
> - Andrew Lamb, Field Ready CEO

> *We've very happy with the software, its exciting to use it as it is to generate new pipe fittings!*
>
> - Mark Mellors, Field Ready Manufacturing Engineer (sic)

Figure 9: Customer Feedback

8.3 Overall Evaluation

Based on the fact that we successfully implemented not only our minimum requirements but also the majority of the add-ons identified as valuable, and the reception that the product has received from the client, we are very proud of what we have achieved with MakeFit and consider it a real success.

That said, there are of course a number of ways in which it could be built upon and improved, as detailed in the next section. We have taken measures to ensure that such a task is as easy as possible, through extensive code commenting, the creation of a technical handover document describing all code modules (including purpose, dependencies and details of what function(s) call them), and the creation of an instruction sheet for anyone who needs to recreate the installer package.

8.4 Next Steps

As previously mentioned, the next step for our client is to demonstrate MakeFit at the World Humanitarian Summit later this month. Their goal is to attract the interest and sponsorship required to deploy MakeFit across the globe, which will include further work to incorporate additional parts as well as potentially widen the tools use to include items other than water pipe connectors. Regardless of the outcome of the summit, later this summer they intend to introduce the preliminary version of the tool to aid workers from partner organisations Oxfam and World Vision, in order to gather

front-line feedback.

We anticipate the next steps from a development perspective to be as follows:

1. Add the functionality required to create the two additional parts marked with placeholders, namely an elbow joint and a T-junction
2. Add functionality to mark the printed parts with batch numbers or other identifying marks, and to keep a log of all parts made in a central database for tracking purposes (requiring internet access)
3. Create a cloud-based version of MakeFit, with benefits including the ability to make and deploy improvements on an ongoing basis
4. Create offline versions of MakeFit compatible with other popular operating systems, including MacOS

Additional testing is also required on the end products, to ensure that the fittings work as intended under a range of conditions and water pressures.

While we do not anticipate taking the lead on any further work at this stage, we have offered to provide ongoing assistance to Field Ready and hope to continue to be involved with MakeFit on some level for as long as it is in use.

9 Appendices

9.1 Appendix A: Logbook

Name	Date	Minutes	Topic
AT	18/1/2016	20	Planning
AT	22/1/2016	120	Planning
AT	25/1/2016	60	Research
AT	25/1/2016	20	Planning
AT	26/1/2016	60	Planning
AT	30/1/2016	60	Report 1
AT	31/1/2016	20	Report 1
AT	2/2/2016	20	Planning
AT	2/2/2016	60	Documentation
AT	4/2/2016	90	Documentation
AT	5/2/2016	60	Documentation
AT	9/2/2016	20	Planning
AT	13/2/2016	90	Front end
AT	22/2/2016	60	Documentation
AT	22/2/2016	215	Front end
AT	23/2/2016	30	Testing
AT	24/2/2016	30	Planning
AT	24/2/2016	150	Front end
AT	29/2/2016	360	Front end
AT	29/2/2016	30	Documentation
AT	1/3/2016	30	Documentation
AT	4/3/2016	90	Front end
AT	5/3/2016	120	Report 2
AT	6/3/2016	120	Report 2
AT	7/3/2016	120	Testing
AT	8/3/2016	300	Report 2
AT	9/3/2016	180	Report 2
AT	10/3/2016	150	Report 2
AT	11/3/2016	330	Report 2
AT	14/4/2016	120	Front end
AT	15/4/2016	300	Front end
AT	16/4/2016	105	Front end
AT	20/4/2016	120	Front end
AT	21/4/2016	165	Front end
AT	22/4/2016	135	Front end
AT	9/5/2016	90	Planning
AT	10/5/2016	300	Report 3
AT	11/5/2016	360	Report 3
AT	11/5/2016	120	Testing
AT	12/5/2016	540	Report 3
AT	13/5/2016	360	Presentation
MC	26/1/16	120	Research
MC	2/2/2016	120	Research
MC	9/2/2016	60	GUI
MC	16/2/16	120	GUI
MC	23/2/16	120	Testing
MC	1/3/2016	60	Testing
MC	2/3/2016	120	Testing
MC	8/3/2016	120	Testing
MC	15/3/2016	120	Testing
MC	5/4/2016	120	GUI
MC	12/4/2016	60	GUI
MC	19/4/2016	120	Testing
MC	9/5/2016	180	Testing
MC	10/5/2016	180	Report 3
MC	11/5/2016	180	Report 3
MC	12/5/2016	180	Report 3

Name	Date	Minutes	Topic
LF	25/1/2016	120	Back End
LF	2/2/2016	120	Back End
LF	8/2/2016	120	Back End
LF	14/2/2016	60	Back End
LF	15/2/2016	60	Back End
LF	15/2/2016	60	Back End
LF	20/2/2016	120	Back End
LF	24/2/2016	120	Back End
LF	25/2/2016	120	Back End
LF	28/2/2016	60	Back End
LF	28/2/2016	120	Back End
LF	1/3/2016	180	Back End
LF	4/3/2016	60	Back End
LF	10/3/2016	120	Report 2
LF	12/3/2016	120	Back End
LF	10/4/2016	120	Back End
LF	19/4/2016	120	Testing
LF	9/5/2016	60	Testing
LF	9/5/2016	180	Back End
LF	10/5/2016	180	Report 3
LF	11/5/2016	180	Report 3
LF	12/5/2016	360	Report 3
HW	2/2/2016	60	Report 1
HW	4/2/2016	90	Report 1
HW	5/2/2016	300	Report 1
HW	23/2/2016	100	GUI
HW	24/2/2016	50	GUI
HW	24/2/2016	40	GUI
HW	29/2/2016	212	GUI
HW	29/2/2016	300	GUI
HW	1/3/2016	61	GUI
HW	4/3/2016	35	GUI
HW	8/3/2016	300	Report 2
HW	9/3/2016	180	Report 2
HW	10/3/2016	150	Report 2
HW	11/3/2016	330	Report 2
HW	14/4/2016	175	GUI
HW	15/4/2016	60	GUI
HW	15/4/2016	232	GUI
HW	20/4/2016	225	GUI
HW	21/4/2016	210	GUI
HW	9/5/2016	279	GUI
HW	10/5/2016	155	GUI
HW	10/5/2016	60	GUI
HW	10/5/2016	60	GUI
HW	10/5/2016	203	GUI
HW	11/5/2016	150	GUI
HW	11/5/2016	150	GUI
HW	11/5/2016	60	Report 3
HW	12/5/2016	400	Report 3

Name	Date	Minutes	Topic
KS	22/1/2016	60	Research
KS	2/2/2016	60	Research
KS	14/2/2016	120	Installer
KS	15/2/2016	60	Installer
KS	16/2/2016	150	Installer
KS	24/2/2016	30	Testing
KS	27/2/2016	60	Back End
KS	29/2/2016	30	Installer
KS	1/3/2016	30	Installer
KS	3/3/2016	60	Installer
KS	8/4/2016	120	Back End
KS	11/4/2016	240	Back End
KS	12/4/2016	360	Back End
KS	12/4/2016	120	Back End
KS	13/4/2016	240	Back End
KS	13/4/2016	30	Back End
KS	13/4/2016	120	Back End
KS	9/5/2016	180	Installer
KS	10/5/2016	360	Installer
KS	10/5/2016	360	Installer
KS	11/5/2016	360	Report 3
KS	12/5/2016	360	Report 3
DJ	1/3/2016	180	Back End
DJ	28/2/2016	150	Back End
DJ	23/2/2016	300	Back End
DJ	19/2/2016	180	Back End
DJ	15/2/2016	120	Back End
DJ	14/2/2016	180	Back End
DJ	20/2/2016	180	Back End
DJ	24/2/2016	240	Back End
DJ	26/2/2016	240	Back End
DJ	28/2/2016	180	Back End
DJ	2/3/2016	240	Back End
DJ	5/3/2016	180	Back End
DJ	7/3/2016	120	Back End
DJ	8/4/2016	120	Back End
DJ	11/4/2016	240	Back End
DJ	12/4/2016	360	Back End
DJ	12/4/2016	120	Back End
DJ	13/4/2016	180	Back End
DJ	13/4/2016	60	Back End
DJ	13/4/2016	60	Back End
DJ	10/5/2016	60	Back End
DJ	10/5/2016	240	Report 3
DJ	11/5/2016	240	Report 3
DJ	12/5/2016	360	Report 3

Key	
AT	- Angharad Thomas
MC	- Melinda Chan
LF	- Luca Filipi
HW	- Henry Williams
KS	- Kathryn Shea
DJ	- David Joyce

Figure 10: Individual Time Logs excluding Team Meetings

9.2 Appendix B: Meeting Records

Figure 12 shows the records of our weekly group meeting minutes. It does not contain details of our shorter Scrum meetings, or meetings held within programming teams; it also excludes agenda items common to every Scrum meeting including progress updates and activities for the next week. We used this alongside an action log to outline required next steps from team members. A snapshot of the action log can be seen in Figure 11 below.

Item	Date	Action	Responsible	Comments	Date Due	Status	Importance
1	18/01/16	Arrange meeting with Andrew	KS	Coming to Tuesday meeting	22/01/16	Complete	5
2	18/01/16	Arrange meeting with Anandha	HW	Meeting 22/1	22/01/2016	Complete	5
3	18/01/16	Set up github repository	DJ	Await CSG link	05/02/16	Complete	2
4	18/01/16	Set up google drive	LF	Include AG	22/01/2016	Complete	4
5	18/01/16	Add PM docs	AT		22/01/2016	Complete	3
6	18/01/16	Create log template	KS	Share	22/01/2016	Complete	2
7	18/01/16	Prepare for workshop session on 26/1	All	All need to be ready to 'map out' our end-to-end product including interfaces, code types, potential roadblocks etc	26/01/16	Complete	5
8	22/01/16	Research input to OpenSCAD	DJ / LF		26/01/16	Complete	4
9	22/01/16	Research competing / helpful software	KS	Check out the research notes, good stuff	26/01/16	Complete	2
10	22/01/16	Research GUI	HW	Docker / Software exe	26/01/16	Complete	3
11	22/01/16	Research stl verifier	MC		26/01/16	Complete	4
12	26/01/16	Schedule call with Abi	DJ		28/01/16	Complete	4
13	26/01/16	Draft requirements for review by FR	AT		29/01/16	Complete	4
14	26/01/16	Validate group email	HW		02/02/16	Complete	1
15	28/01/16	Get files requested from Abi	DJ		05/02/16	Complete	2
16	28/01/16	Create and send template for tasks, milestones, risks etc for team to fill in	AT		28/01/16	Complete	2
17	28/01/16	Front end: Milestones, tasks, KPIs, risks	HW, AT, KS		02/02/16	Complete	3
18	28/01/16	Back end: Milestones, tasks, KPIs, risks	DJ, LF, MC		02/02/16	Complete	3
19	26/01/16	Write up dev strategy and intro sections for report	AT, HW		02/02/16	Complete	3
20	28/01/16	Print pipe	MC		29/02/16	Complete	1
21	09/02/16	Get projector for next (each) meeting	HW		09/02/16	Complete	1
22	09/02/16	Familiarise with Python - Toolchain	Team		16/02/16	Complete	4

Figure 11: Log of Team Meetings with Key Points

Date	Attendees	Duration (m)	Agenda	Key points agreed / discussed	Further comments
18/01/16	Team	45	- Identify and allocate initial tasks	- AT project leader, HW documentation editor - Agreed to use github and its built-in VCS - Agreed to create research folder and share all docs for ease of reference	Booked room for 1.30-3pm every Tuesday for regular team meetings Planning 'workshop' session for next Tuesday, ideally with Anandha and/or Andrew
22/01/16	Team (excl. AT) + AG	70	- Meet with AG to discuss next research steps	- Initial research tasks defined (see actions for allocation) - Added AG to google drive - AG to join Tuesday meeting "Field Ready" team - Discuss with Andrew, Robotics team, Fariba (can allocate budget) about potential for 3D printer access - Team to go through calendar thoroughly and fill in availability	AG Comments: - Concentrate on defining minimum viable product - Need to able to show (and crucially, verify) a result - Showing that pipe stl through software matches pre-existing stl could be example of proof of concept - Highest risk could be setting up a good GUI - Timeline should be highly conservative with time left at end just for checks - Pair programming at a minimum (better for debugging)
25/01/16	Team	20	- Update AT on 22/01 meeting - Discuss client meeting 26/01	- Basic structure of client meeting agreed - Avoid focus on technical issues and focus on client needs while managing expectations - Identify personnel to contact for further requirements (e.g. for example pipe spec.)	
26/01/16	Team, AG, AL, AB (skype)	90	- Discuss product requirements with customer (Field Ready) - Risks (from FR perspective) - Identify next steps/how to work together		See separate document for detailed agenda and meeting notes
28/01/16	AT, DJ, KS, AB (skype)	30	- Shared vision for product and our involvement (i.e. we concentrate on the software versus engineering the parts being created)	- AB to send us a 2D CAD and list of changeable parameters for our initial part	
28/01/16	Team (excl. MC)	30	- Product features - Module split (initial) - Software development strategy	- Agreed prioritised list of features (minimum, potential add ons, 'longer term enhancements') to send to FR - Agreed dev strategy will be a hybrid of various agile methodologies (see further comments) - Split into two teams to identify tasks, milestones, timelines and risks: Front end (HW, AT, KS) and back end (DJ, LF, MC)	Pair programming, self-directed teams, 'product backlog' type documentation (i.e. prioritised tasks with deadlines, names, comment logs etc - using Asana), sprint mentality (reflection/review meetings at each milestone), ongoing customer communication (sharing product each time a milestone is reached for feedback purposes) SCRUM meetings twice weekly in addition to more detailed weekly meeting (in which plans etc will be reviewed and amended)
02/02/16	Team, AG	60	- Finalise Report One content: focus on activities, deadlines, risks, and KPIs	Agreed content: see Report One for final details	
09/02/16	Team, AG	60	- Finalise division of labour - Share research highlights - Introduction to Asana (AT)	See actions log	
16/02/16	Team, AG	90	- Review Asana - Meeting dates with FR - Blog post - Updates from research - Discuss parameters to back-end	See actions log	
24/02/16	Team, AG	60	- Discuss Testing Research - Review wireframe	Everyone needs to use git and Asana Agreed testing procedure going forward, MC in charge	
25/02/16	Team	60	- Working session; main goals (1) to ensure everyone is using git the same way, and (2) to produce end to end (simple) exe	Drew up 'map' of product to clarify all understand integration and determine how to split for unit testing	
01/03/16	Team, AG	60	- Testing update and plan - Share GUI - Reminder about logs - Agree on next steps for discussion with Andrew	Client Workshop meeting agenda	
07/03/16	Team	30	- Review of Report 1 Feedback	Stick to framework strictly in future	
08/03/16	Team, AG	90	- Preparing for client meeting with trial run	Force key questions to be answered	
09/03/16	Team, AG, Field Ready	120	- Status update - Demo - Prioritisation of next steps - Testing - Survey		See separate document for detailed agenda and meeting notes
10/03/16	Team	60	- Client Meeting Review	Positive Initial feedback with key points to follow up summarised in meeting notes	
23/03/16	Team	15	- Pre-holiday review	Continue with individual roles over the holiday	
14/04/16	Team (excl. LF)	60	- Updates from each sub-team and next steps	Back end mvp mostly done following mark's comments	
22/04/16	Team	30	- Catch up before exams	Reviewed status	
08/05/16	Team (excl. LF)	60	- Report 3 - Full review of product - User testing	- Consider additional part - Installer Testing: scan for OpenSCAD and pop up request to install it if not installed - Code coverage testing - Partition testing: end-to-end - Reply to Andrew email	Received T junction from Abi (further part) - too difficult to do in time given
09/05/16	Team	45	- Presentation - Handover documents - Final debugging	- Naming of files: create sub-directory - O-ring mould to be included - Report outline	Potential Extras: - Handbook - Handover documentation
09/05/16	Team + AG	60	- Review with supervisor	Focus on presentation key	AG: - have as an option being installed on USB - presentation: 20mins strict plus 10mins questions

Figure 12: Log of Team Meetings with Key Points

9.3 Appendix C: Client Workshop Minutes

Client Workshop 1 - 26th January 2016
Location: Room 219 Huxley Building
Present: Group 4, Anandha Gopalan, Andrew Lamb, Abi Bush (Skype)

Key points in the meeting are indicated in italics in chronological order

Introductions from both sides and brief summary of project

Andrew Lamb (AL):

- Has a background in systems and software engineering
- Worked for Engineers Without Borders, now part of numerous charitable organisations including Field Ready

Abi Bush (Abi):

- Manufacturing engineering graduate (2014)
- Field Ready's technical advisor based in Nepal

Project Context

Following earthquakes in Nepal, the general consensus from aid agencies on the ground in the disaster areas was that villages needed water pipe fittings.

The villages, often embedded near the base of hill slopes, attempt to pipe water from springs, or containers, from higher down to the housing below. Many of the connections between the pipes are completely improvised (tape, sticks, bike wheel, plastic bag stuffed in to prevent leaks).

Thus far, Field Ready have designed a simple 3 part pipe fitting which can be 3D printed on site using the power from a car battery. The project aim is to enable aid workers, who will most likely have no technical training and often minimal English language proficiency, to 3D print basic pipe fittings on-site.

The 3 phases of the aid process, related to water pipe fittings are as follows:

1. Identify need
2. Design piece
3. Manufacture piece

The goal of the proposed software would be to automate the design phase as much as possible, since training in 3D modelling and CAD design is time and resource intensive.

The following are the priority pipe types:

1. End-to-end connections with potentially asymmetrical diameters
2. T-junctions
3. Pipe shoulders
4. O-ring moulds

The pipe sizes will range from 0.5 to 4 inches

Field Ready Additional Concepts - Biggest Risks (from Field Ready's point of view)

Key Considerations

- The software must be available offline as the network connections are unreliable
- The software will most likely be run off a windows laptop and be transferred/shared via USB
- The installation, interface and usability must be as simple and as graphical as possible
- We will need to consider efficient creation of o-rings or design fittings with minimum use of o-rings

Quality Control

- Many of the workers are not capable of discerning between a well constructed pipe and a poor one
- Software could include check-list for workers to go through as quality control

Accountability

- Who: Account for Who made the pipes
- What: Account for precisely what was made
- When: Account for when the pipes were made (to understand lifetime)

A possible solution to address the aforementioned issues will be to add a cut name, identifier, date etc. into pipe fitting.

Field Ready's business model

- Field Ready is currently supported by innovation grants - looking for business model
- Potential to commercially use this software at a later stage

Next Steps

- Converse with Abi to further discuss part considerations, proof of concept and minimum viable product ('MVP')
- Following call, further research before agreeing on MVP, other possible ventures, and future projects

Client Workshop 2 - 9th March 2016
Location: Room 554 Huxley Building
Present: Group 4, Anandha Gopalan, Andrew Lamb, Mark Mellors (Skype)

Demonstration of Product: Feedback on demo

- CAD screen-shot would be better than current diagram on parameter input page.
- Need separate files for the components of a part, allowing user to select a subset for download.
- Request to investigate the potential to directly address STL file rather than reconstruction in OpenSCAD. This will reduce the need to test output each time since a lot of effort has gone into the provided design already.

- Would like something to notify the user that the program is working in the background when constructing files (e.g. estimated download time or progressbar).
- Try to keep file below 30MB so file can be emailed if required.
- Parameter limits to be supplied by Field Ready. Positive response to team suggestion of warning's for more unusual sizes in conjunction with hard limits for infeasible parameters.
- Message boxes should be restricted to errors since they are not as nice a user experience. Suggested alternative of red text appearing under entry box.
- Use parameter drop down boxes to show common pipe sizes as well as allowing custom input.
- Leave unavailable parts on MenuPage as Field Ready will also be using the software for demonstration purposes and this will illustrate that it is a work in progress.

Future Developments (Priority 5 being the highest priority) Priority 5 - Help / About Page
Priority 5 - Dual Units - clear of difference between metric and imperial
Priority 5 - Representative images (CAD screenshot images) for each part
Priority 4 - O ring mould
Priority 4 - Drop down menus
Priority 4 - Timing warning / progressbar (need to check threads)
Priority 3 - Quality checklist placeholder
Priority 3 - Buffer parameter values
Priority 2 - Error messages inside software
Priority 2 - Log: batch numbers and part design
Priority 1 - Image previews

Testing

- Client will try to access 3D printer to facilitate end-to-end testing
- Client to arrange team of target end-users for testing and feedback

Overall, Field Ready is pleased with the progress. Andrew does not feel the need to have another internal meeting with us (he is moving to Geneva so it is harder to arrange) but may meet in person with Mark to discuss product output files if needed.

9.4 Appendix D: Additional GUI Screenshots

The screenshots below show a number of the GUI's pages and features, which follow the pages shown in Section 3.2.

The Component Selection page follows the Parameter Input page (shown in Section 3.2) and allows the user to select which particular components they wish to create STL files for. This is done by ticking the appropriate boxes next to each component.

Pressing on the 'Download Selected Files' button in the Component Selection page brings up the Save Location pop-up window, which prompts the user to select a destination directory where the generated STL files will be saved.

This is followed by the Run page, where the user is given the opportunity to review the chosen part as well as the selected components before choosing to create the STL files, which is done by pressing the 'Run' button.

Note that, while the screenshots included below show the aforementioned pages for the straight coupler part, they would appear analogously for any other part (except for the Component Selection page, which is only relevant for parts with more than one component).

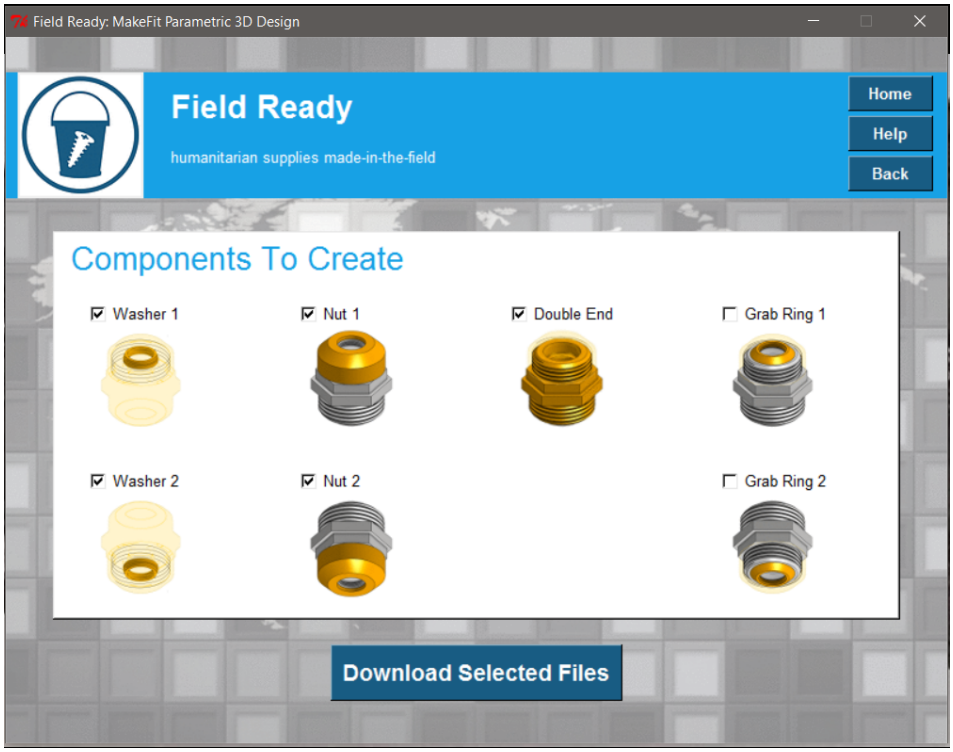


Figure 13: Component Selection Page

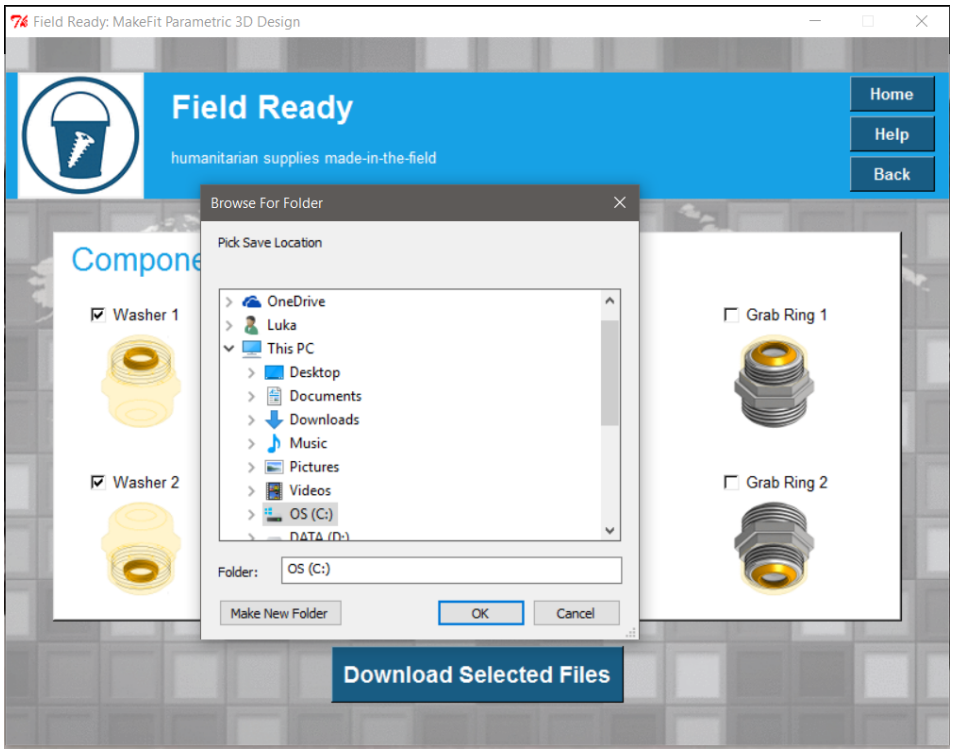


Figure 14: Save Location Pop-Up Window

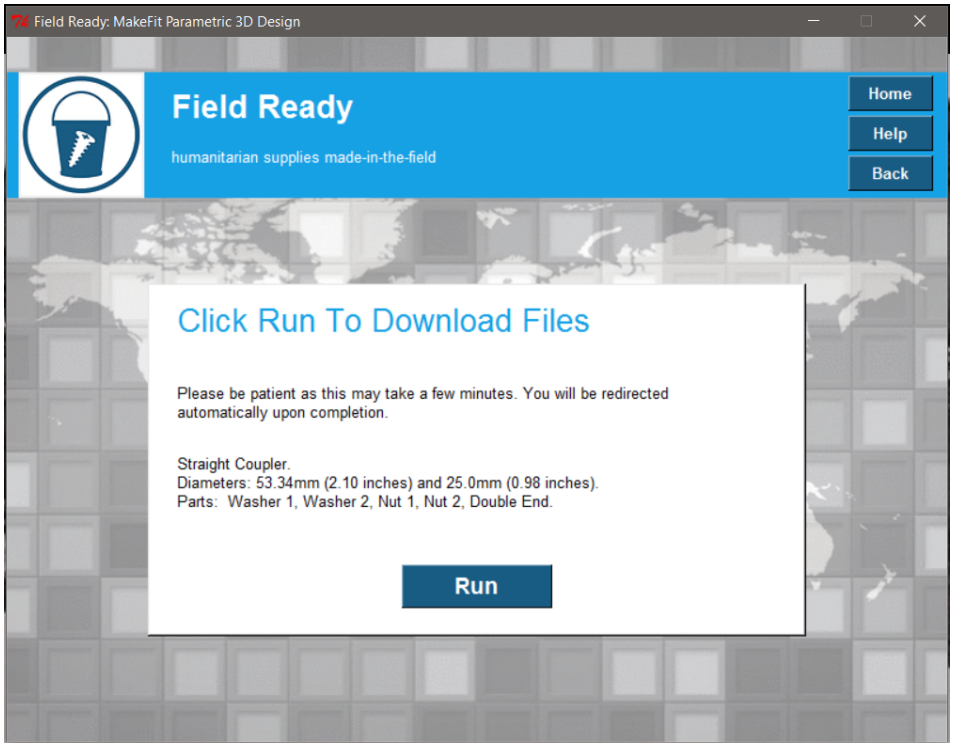


Figure 15: Run Page

9.5 Appendix E: Original Feature Prioritisation

The following feature prioritisation was the result of our kick-off workshop with Field Ready, as shown in report one.

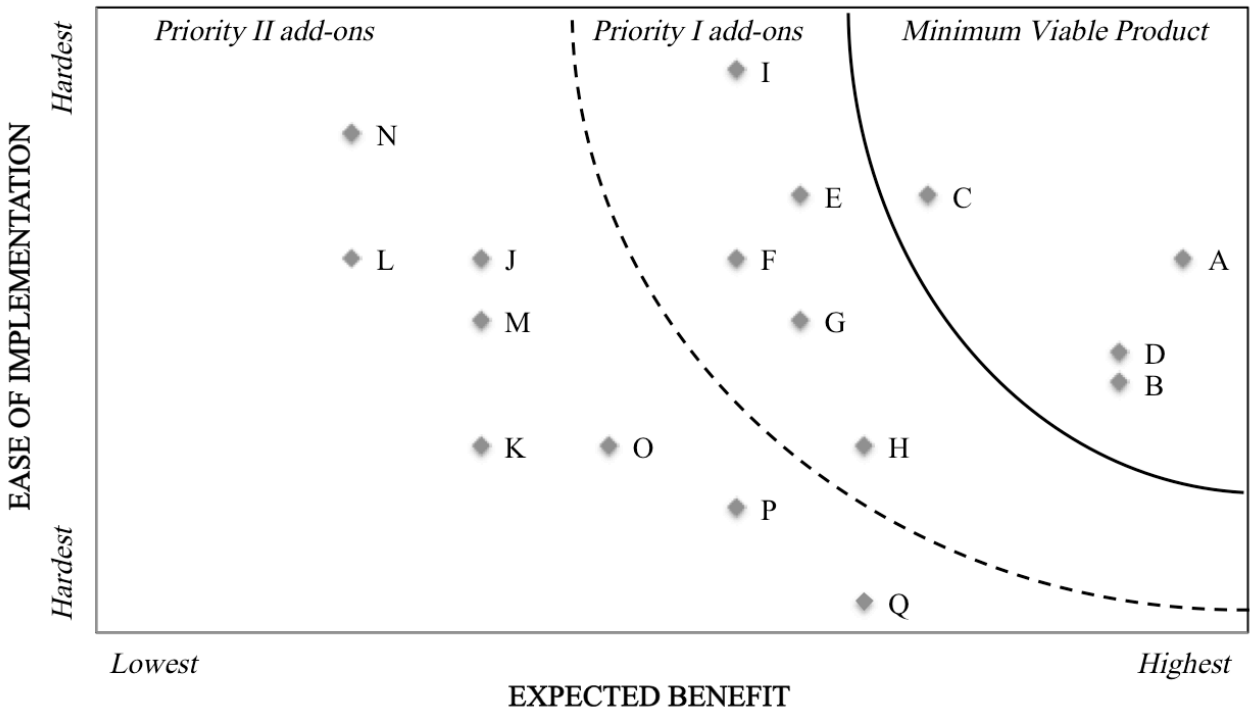


Figure 16: Potential Features and Functionality

Minimum Viable Product Requirements

The product created must:

A Generate the STL files required to 3D print a single water pipe connector type

B Use a parametric approach to designing parts, allowing customisation by the user

C Work as a stand-alone application not requiring internet connection

D Have a simple, intuitive and graphical user interface requiring little to no user training

Additional Features and Functionality

If time permits, additional work we could undertake includes the following:

Priority I

E Add functionality allowing advanced users to directly access, view and customise CAD files

F Amend the architecture to facilitate easy addition of further parts for printing

G Add identifying marks or batch numbers to connectors for accountability and tracking

H Create additional pipe connectors, including T-junctions and shoulder connections

I Add a quality checklist for printed connectors

Priority II

J Create a cloud-based alternative to laptop application

K Investigate options for licensing and potentially commercialising the product

L Add commenting functionality to record user notes

M Add error reporting functionality (requires internet connection)

N Add links to tutorials (requires internet connection)

O Create moulds (e.g. for o-rings)

P Create other types of items, such as medical supplies

Q Incorporate image recognition software to identify part and parameter requirements

9.6 Appendix F: Re-prioritisation of Additional Features from Workshop Two

The specification for extension work was revised following the client meeting on 9 March 2016. Work in Phase 3 focused on the following list of prioritised amendments and enhancements, starting from the top and working down as far as time constraints allowed. Note that 'priority' here represents the impact of the change and its alignment with the project's overall goals, with 'feasibility' taking into account both the difficulty of the amendment from a technical perspective and the expected time requirement.

The only modules that have not been completed are marked as NOT COMPLETED in the last three entries in Table 4.

Module	Amendment	Categorisation	Priority (1-5)*	Feasibility (1-5)*
GUI	Add text to warn users about time taken for file creation	Low-level, low-risk, functional	5	5
GUI	More explicit diagrams	Low-level, low-risk, non-functional	5	4
GUI	Allow inputs in both mm and inches	Low-level, low-risk, functional	4	4
GUI	Add images for each individual component on final page	Low-level, low-risk, non-functional	4	4
GUI	Add Help page (to be populated by client)	High-level, low-risk, functional	4	4
All	Allow users to select subset of components for STL generation	High-level, high-risk, functional	4	3
GUI	Create drop down menu with standard measurements	Low-level, low-risk, functional	4	3
GUI	Add quality checklist (to be populated by client)	High-level, low-risk, functional	3	4
GUI	Add functionality for users to validate custom parameters	Low-level, low-risk, functional	3	3
GUI, Back End	Add O-ring moulds to existing part	High-level, high-risk, functional	3	2
GUI	Replace message alerts with responsive text in window	Low-level, low-risk, functional	2	3
All	Investigate adding functionality to directly edit STL files	High-level, high-risk, functional	3	1
GUI, Back End	Add batch numbers and maintain a log of parts made (NOT COMPLETED)	High-level, high-risk, functional	3	2
GUI, Back End	Create additional part (NOT COMPLETED)	High-level, high-risk, functional	3	1
GUI	Update images of components with scaled versions (NOT COMPLETED)	Low-level, low-risk, non-functional	2	1

Table 4: Prioritised Table of Next Development Steps *(1=Very low, 5=Very high)

9.7 Appendix G: Installer and Executable Stress-Testing

Stress Testing

A series of stress tests were designed, summarised in Table 4, to be run by the team and external testers provided by the client. MakeFit is intended for use on Windows OS, and the project boundaries state that compatibility with other environments is not required. As such, all of the listed tests relate to Windows OS. The application passes all of the tests in the table.

Summary of Installation Process

The installer is a single executable that the user will download and open. It asks where to save the program and whether or not to include a shortcut on the desktop. The user can opt to cancel the download at any point during the installation, rolling back the process.

Test	Expected Result	Internal Test
Run installer; opts to install icon	Successful installation of program; icon on desktop	Pass
Run installer; opts not to install icon	Successful installation of program; no icon on desktop	Pass
Run installer without internet connection	Successful installation of program	Pass
Install on a machine which does not already have OpenSCAD or Python libraries	Program runs without any errors resulting from 'missing' software or libraries	Pass
Install on a machine which already has older versions of OpenSCAD and Python libraries	Program runs successfully, independently of preinstalled software	Pass
User uninstalls application	All elements of installation package are removed and the user's version of OpenSCAD (if exists) is unharmed	Pass
User cancels installation before complete	Installation halts with error message; any elements already installed are removed	Pass
User opts to install OpenSCAD in different directory to main program	Program able to locate and use OpenSCAD	Pass
Installer run on a machine which already has program installed	Installer deletes and replaces existing instance of program	Pass

Table 5: Installer and Executable Test Results

9.8 Appendix H: User Survey Results

MakeFit Testing: Situation Brief

Firstly, if you're reading this then you must be at least considering testing out our product to give us some feedback – so thank you! Not only does this product form part of our Masters evaluation (due for submission on Friday), but it's also being showcased at the World Humanitarian Summit in Istanbul later this month and then rolled out to aid workers in Nepal later this summer – so we really want to get it right.

It's worth highlighting at this stage that the installer has been configured for Windows machines only, and will require that you have permission to install a program. It's small, self-contained, virus-free, and you can easily delete it afterwards – so please don't be put off!

Background

In a humanitarian disaster, long supply chains and logistical constraints can make it difficult to quickly source relatively simple but critical and potentially life-saving items. In response to this need, the non-profit organisation Field Ready is working with a number of leading global humanitarian aid agencies, including Oxfam, to start printing parts needed on-the-ground using aid workers' 3D printers. Recently they have been focusing on creating connectors for water pipes, as the makeshift alternatives widely used can lead to water loss or contamination.

At present the software required to customise files for 3D printing is complex, and requires a relatively high degree of technical training – preventing this approach being rolled out on a wider scale. To address this we have created MakeFit, which is a (hopefully!) very easy-to-use program which lets the user create customised water pipe connector fittings.

What we need

Since we now know the program inside out, we need feedback from objective individuals with no training at all in using MakeFit to let us know what they think – and in particular, how easy and appealing it is. It's really basic, so it will only take a few minutes to click through once it's downloaded and fill in our very straightforward survey. Even if you ignore all text inputs and just rate it out of 5 for attractiveness and usefulness, that will give us useful statistics we can use to gauge how well we have done!

Instructions

1. Download the file entitled 'setup.exe' to your computer, then double click it to start the installer. If you follow the instructions, it should create a shortcut on your desktop
2. Open up this survey:
   http://goo.gl/forms/bz2BCU2t3r
3. Imagine that you're an aid worker trying to find a way to connect two water pipes of different sizes. The background setting is up to you – let your imagination go wild! You will have a laptop and a 3D printer in your van, although there's a good chance you won't have internet connection.
4. Now open up the MakeFit program, have a play around, and let us know what you think using the survey. There are a few placeholders for content from our client, so ignore these for now. *Warning: Creating the STL files can take a few minutes, so in the interests of time it's probably worth limiting the components you select for creation to a nut*

Thank you for your help!!

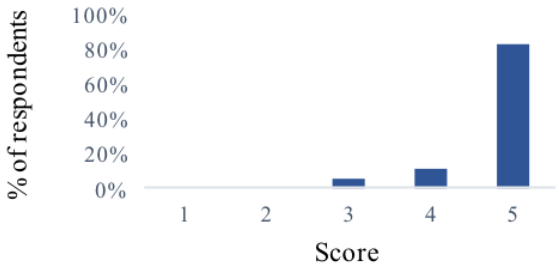
Angharad, Katie, Henry, David, Luca and Melinda

Figure 17: Situation Brief Provided to Survey Respondents

Ease of use

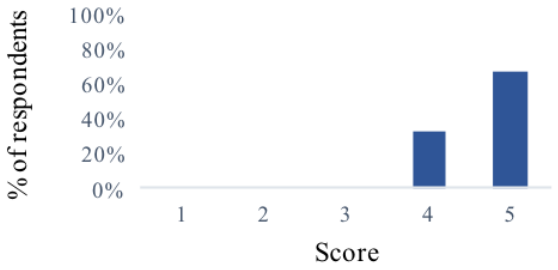
How easy was the program to download and install?

(1 – Very difficult, 5 – Very easy)



How intuitive did you find MakeFit to use?

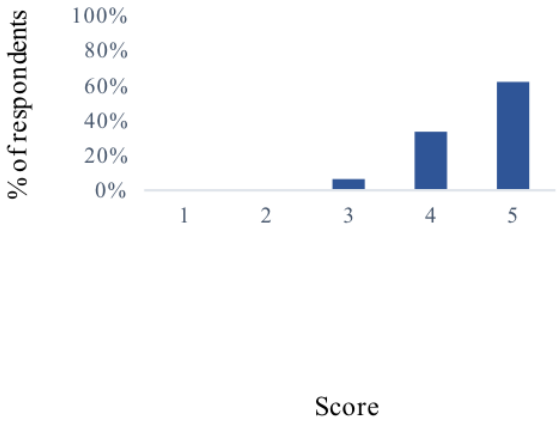
(1 – Not intuitive at all, 5 – Very intuitive)



Responsiveness

How responsive did you find the program? (i.e. how quick was it to respond to events such as button clicks)

(1 – Not responsive, 5 – Very responsive)



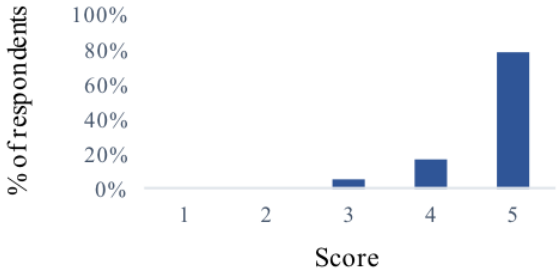
If you found MakeFit unresponsive, please give more detail regarding the issues experienced *(representative selection of comments)*

- Took quite a while to download the files for the parts, although that may be down to a number of other factors unrelated to your program!
- When it takes a bit of time to run the download to get the files, it wasn't clear to me if it had actually started - perhaps you could grey out the button so that the person doesn't keep clicking? (I also tried a case when it worked almost immediately, which is fine)
- When downloading the parts, there is a long delay. I think if you hadn't written in advance that it would take a while, I would have assumed it was broken

Usefulness

How useful is this program, assuming you are looking to acquire a customised 3D water pipe fitting?

(1 – Not useful, 5 – Very useful)



Would you use this program to create water pipe fittings?

(1 – Very unlikely to use, 5 – Very likely to use)

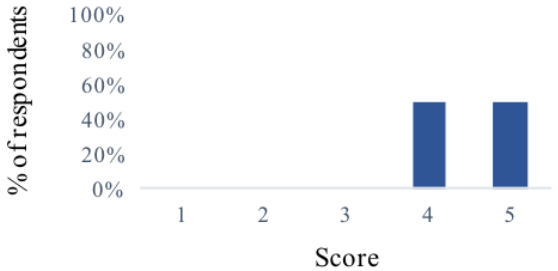
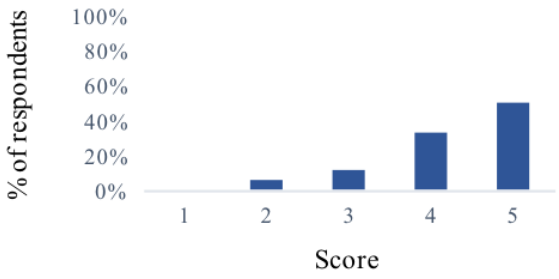


Figure 18: Survey Results Part 1

Appeal

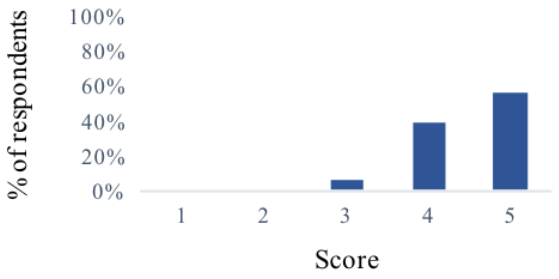
Please rate MakeFit's overall appearance

(1 – Very unattractive, 5 – Very attractive)



Did you enjoy using MakeFit?

(1 – Not at all, 5 – Very much so)



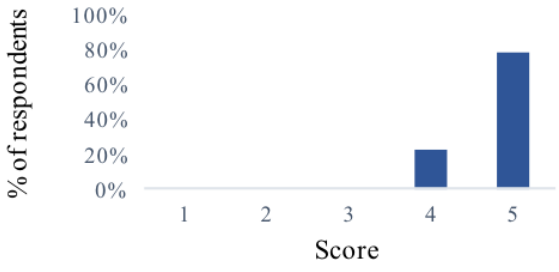
Do you have any suggestions for how the look/feel of MakeFit could be improved? *(representative selection of comments)*

- The feel is great, but the aesthetics could be more appealing
- A loading bar/wheel would be nice when downloading the files
- Interface is a little bit old looking, it could be prettier to look at
- Maybe different fonts / different colour for the button instead of yellow
- Rounded Corners, the orange colour for buttons is a bit much

Overall rating

What rating would you give MakeFit overall, given its brief?

(1 – Unsatisfactory, 5 – Excellent)



Please let us know any other suggestions for changes or improvements, or any other feedback you may have

(representative selection of comments)

- I guess the only bit that might confuse people is the various parts of the actual connector. People might just print all the parts and keep the unused parts as spares. Hence, it might be worthwhile somehow labelling each part with its diameter
- It's great! Only suggestions I would have are small aesthetic issues. It's hard to assess its versatility though because (as you may have guessed) I am not an aid worker. Maybe later iterations of the app could include an advanced option which would allow for more options when designing your pipe? (and other such issues which I'm sure you've considered). Well done though! A really cool applicable app

Figure 19: Survey Results Part 2

9.9 Appendix I: Custom Batch Testing

In order to verify whether the generated files are actually 3D printable, a script was written that generates a large number of parts with different dimensions across the range of permissible values. Each file was then manually tested using the software package NetFabb.

The commented Python script that generates the test files is shown below. It varies the inner diameters of both ends of the pipe fitting from the minimum of 20 mm to the maximum of 80 mm and creates all appropriate components for each combination.

Step 1: Batch Code

The batch generation script below produces a range of STL files with different parameters. In this version it creates 100 different files, varying the inner radius of both ends of the pipe fittings between the two extreme values permitted, set to 10 mm and 100 mm.

```
import sys
import re
import time
import optparse
from ast import literal_eval
from solid import *
from solid.utils import *
from solid import screw_thread
import subprocess

#for testing (don't forget to comment out the new_rad = variables lines)
parts = [0, 1, 4, 5, 6]
directory = 'C:\Users\dell\Desktop\TestParts'

rad1 = 20
rad2 = 25
scadparts = ["part_washer_small.scad","part_washer_big.scad","grabring_small.scad",
             "grabring_big.scad","part_nut_small.scad","part_nut_big.scad","part_centre.scad"]
scaled_stlparts = ["part_washer_small_scaled.stl","part_washer_big_scaled.stl","grabring_small_scaled.stl",
                   "grabring_big_scaled.stl","part_nut_small_scaled.stl","part_nut_big_scaled.stl","part_centre_scaled.stl"]



# iterate from min to max allowed diameter (in mm) in steps of 10
# create the appropriate set of stl files for each combination
for new_rad2 in range(20, 90, 10):
    for new_rad1 in range(20, 90, 10):

        # avoid duplicates
        if(new_rad1 > new_rad2):
            break

        scale1 = new_rad1/float(rad1)
        scale2 = new_rad2/float(rad2)
        lookfor = "openscad.exe"

        # find where OpenSCAD is installed
        for root, dirs, files in os.walk('C:\\'):
            if lookfor in files:
                scadpath = os.path.join(root, lookfor)
                print scadpath
                break

        for file_num in parts:
            stlname = os.path.normpath(directory)
            filename = scaled_stlparts[file_num][:-4] + '_{0}_{1}'.format(new_rad1, new_rad2) + scaled_stlparts[file_num][-4:]
            stlname = os.path.join(stlname, filename)
            proc = subprocess.Popen('"%s" -o "%s" -D scale1=%s -D scale2=%s "%s"'%
                                    (scadpath, stlname, str(scale1), str(scale2), scadparts[file_num]), shell=True)
            while proc.poll()==None:
                time.sleep(0.1)
```

Figure 20: Batch generation script

The software package NetFabb was then used to check each generated STL file for printability. NetFabb performs the following tests:

- Inverted normals - do the designated normals of all faces point outwards
- Non-manifold geometry - are there any unattached faces/vertices or any areas of zero thickness
- Water-tightness - are there any holes in the mesh
- Uniformness - is the STL file a single, uniform mesh

An example output is shown below for the central tube and the nuts of a straight coupler with inner diameters of 20 mm and 30 mm, as well as for a 20 mm O-ring mould.

As can be seen in the figures below, the generated STL files satisfy all the requirements for 3D printability. They are all single, uniform meshes with no holes or inverted normals. The same results were observed for all the other generated files.

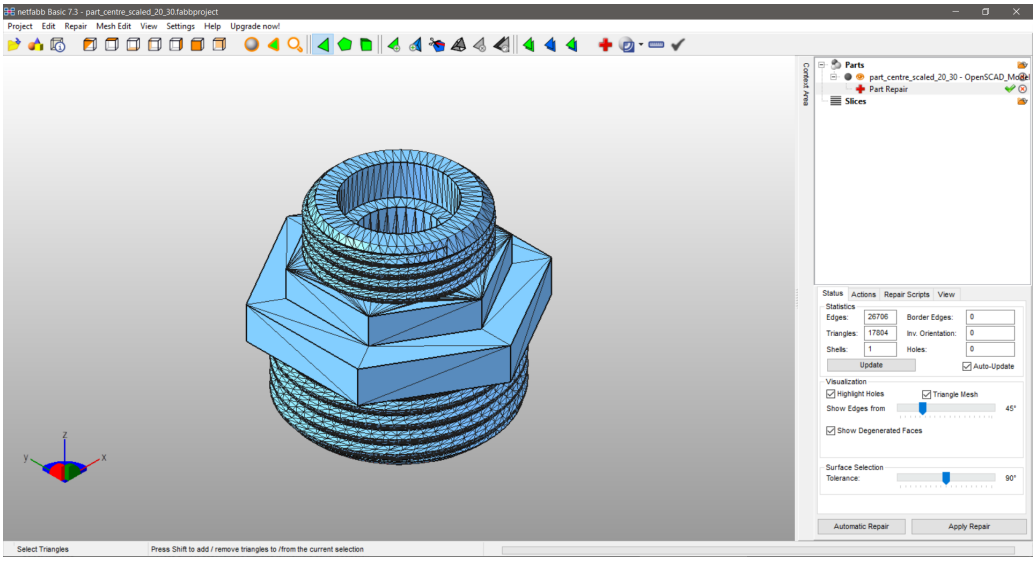


Figure 21: A double-ended threaded centre tube

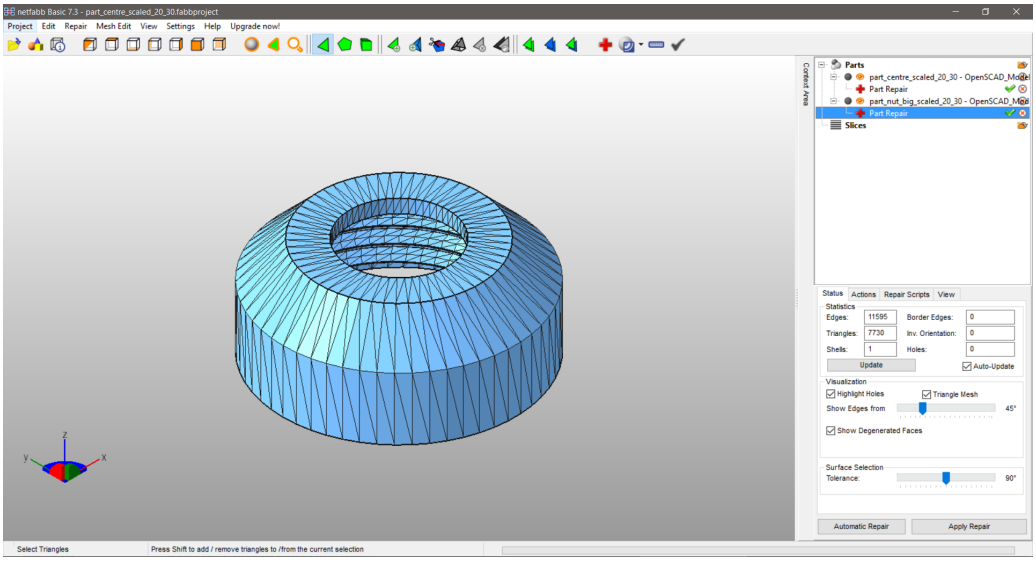


Figure 22: A nut (30 mm diameter)

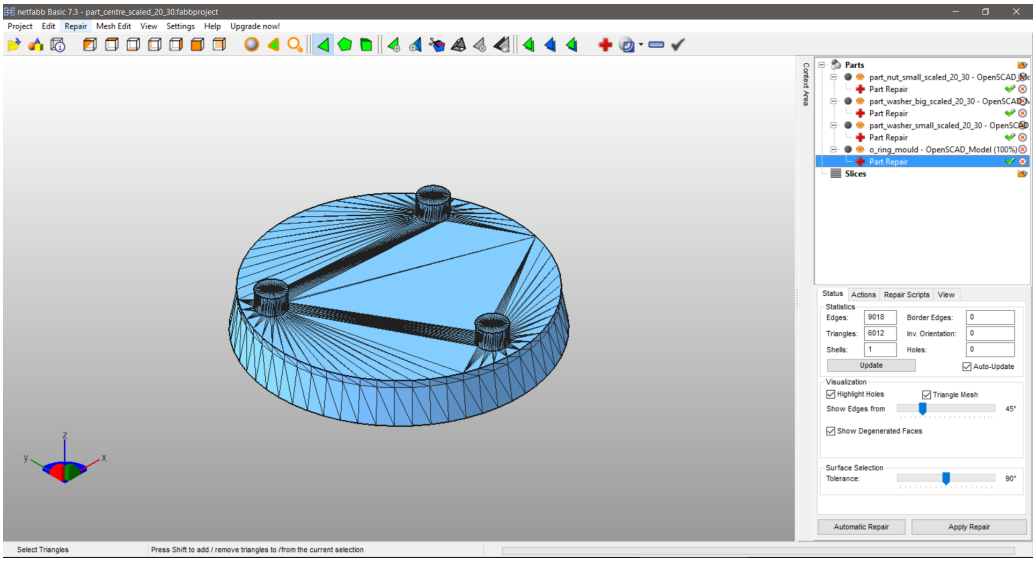


Figure 23: An o-ring mould (20 mm diameter)

9.10 Appendix J: Partition Testing

The test scripts shown in Figure 24 below show the unit test functions used in end-to-end partition testing.

```
#!/usr/bin/python

import os
import unittest

from functions import *
from screwthread import *
from mbox import *

class CheckConverter(unittest.TestCase):

    def test_in_to_mm_conver(self):
        self.assertEqual(convert_in_to_mm(1.0), (1.0*25.4))
        self.assertFalse(convert_in_to_mm(-1.0))
        self.assertFalse(convert_in_to_mm("hello"))
        self.assertEqual(convert_in_to_mm(4.9), (4.9*25.4))
        self.assertEqual(convert_in_to_mm(50.0), (50.0*25.4))
        self.assertEqual(convert_in_to_mm(0.0), (0.0*25.4))

class CheckMakePipe(unittest.TestCase):

    def test_make_sc_pipe(self):
        self.assertTrue(make_SC_pipe("C:\Users\MElin\Desktop\makefit", 25.0, 25.0, [0,1]))
        self.assertFalse(make_SC_pipe("C:\Users\MElin\Desktop\makefit", -0.1, 25.0, [0,1]))
        self.assertFalse(make_SC_pipe("C:\Users\MElin\Desktop\makefit", 25.0, 25.0, [7,9]))
        self.assertFalse(make_SC_pipe("C:\Users\MElin\Desktop\makefitfalse", 25.0, 25.0, [0,1]))

class CheckConversion(unittest.TestCase):

    def test_conversion(self):
        # Normal conversion - should pass
        self.assertTrue(conversion("C:\Users\MElin\Desktop\makefit", [25.0, 25.0], [0,1,5,6]))
        self.assertTrue(conversion("C:\Users\MElin\Desktop\makefit", [25.0, 25.0], [1,5]))
        self.assertTrue(conversion("C:\Users\MElin\Desktop\makefit", [25.0,25.0], [1,4,6]))
        self.assertTrue(conversion("C:\Users\MElin\Desktop\makefit", [30.0,30.0], [0,1]))
        # Testing invalid variables
        self.assertFalse(conversion("C:\Users\MElin\Desktop\makefit", [-10, 0], [0,1]))
        self.assertFalse(conversion("C:\Users\MElin\Desktop\makefit", [0, "a"], [0,1]))
        self.assertFalse(conversion("C:\Users\MElin\Desktop\makefit", [0, -1222], [0,1]))
        # Testing invalid array
        self.assertFalse(conversion("C:\Users\MElin\Desktop\makefit", [30.0,30.0], [0,7,"a"]))
        self.assertFalse(conversion("C:\Users\MElin\Desktop\makefit", [30.0,30.0], []))
        # Testing invalid directory
        self.assertFalse(conversion("C:\Users\MElin\Desktop\makefitfalse", [30.0,30.0], [0,1]))

if __name__ == '__main__':
    unittest.main()
    subprocess.Popen('gui.py')

```

Figure 24: Partition Test Script

For each test, we have included input parameters known to be unacceptable to the function we are testing, as well as a range of possible values which are acceptable. Each function was tested using Python's in-built unit test library. We wrote unit test scripts for each helper function in *functions.py* and also tested the main function in *screwthread.py*, which returned true if the values were acceptable, and false if otherwise.

With regards to GUI testing, we ran through a pre-determined order of buttons to ensure that all buttons were functioning correctly. We have used Python's coverage tool, Coverage.py to determine lines in the code which were executed in the GUI testing

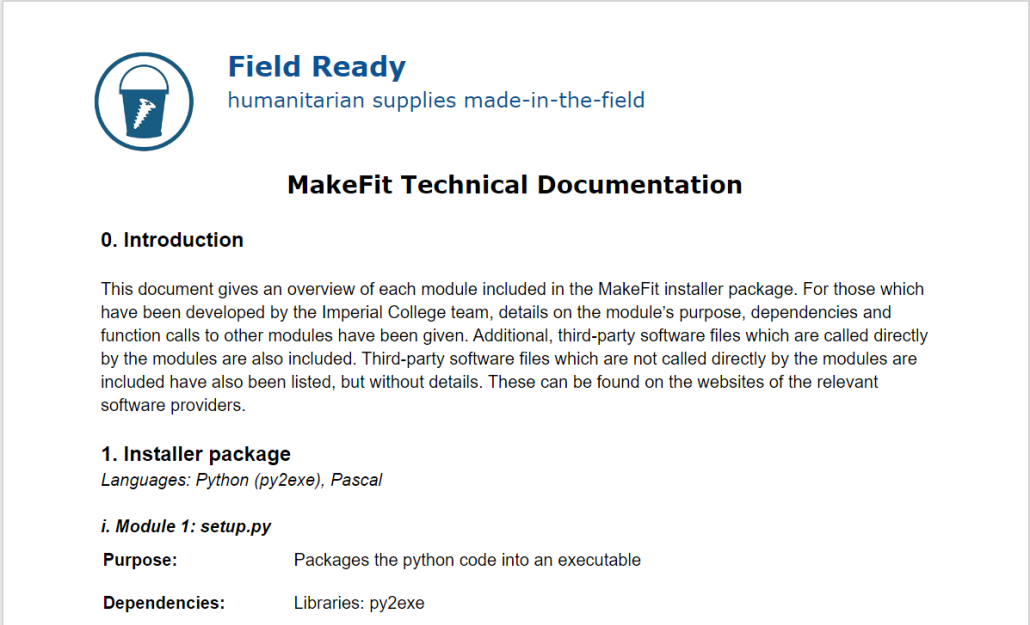
9.11 Appendix K: Handover Documentation

In order to make the handover process as seamless as possible and to facilitate further development work, including adding new features to MakeFit, we have created two documents to be given to Field Ready along with the source code and other required files.

The first of these is entitled 'Technical Documentation', including details of each module's functionality and dependencies, and what would need to change if additional parts were added.

The other document is the 'Installer Manual', which contains step-by-step instructions on how to create the installer for MakeFit using the Windows command line and InnoSetup. This is necessary, as the installer needs to be re-created every time changes are made to MakeFit.

Excerpts from both documents are shown in the figures below.



Field Ready
humanitarian supplies made-in-the-field

MakeFit Technical Documentation

0. Introduction

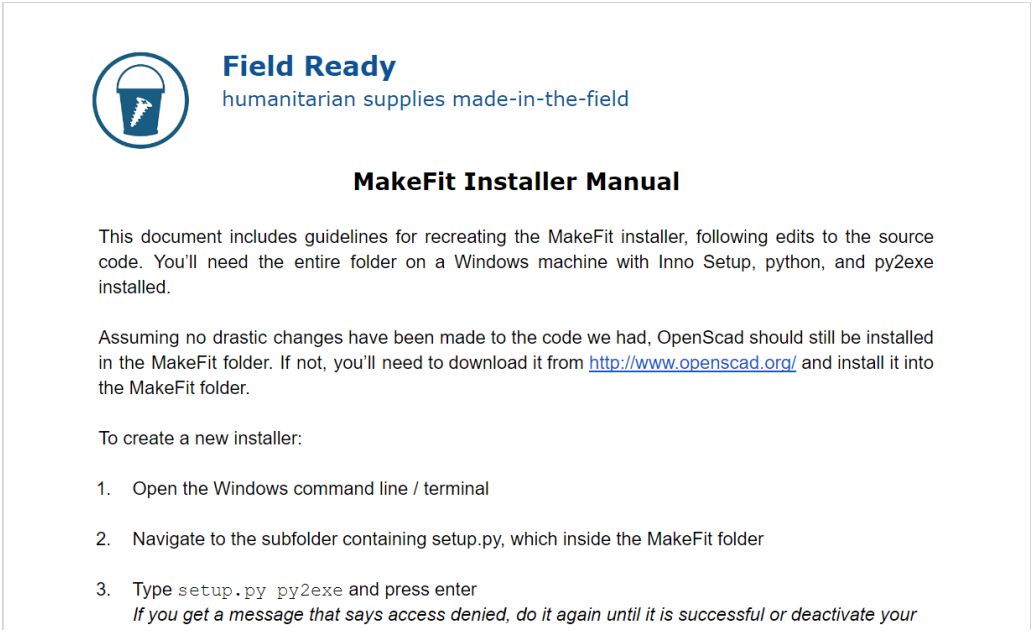
This document gives an overview of each module included in the MakeFit installer package. For those which have been developed by the Imperial College team, details on the module's purpose, dependencies and function calls to other modules have been given. Additional, third-party software files which are called directly by the modules are also included. Third-party software files which are not called directly by the modules are included have also been listed, but without details. These can be found on the websites of the relevant software providers.

1. Installer package
Languages: Python (py2exe), Pascal

i. Module 1: setup.py

Purpose:	Packages the python code into an executable
Dependencies:	Libraries: py2exe

Figure 25: Technical Documentation Excerpt



Field Ready
humanitarian supplies made-in-the-field

MakeFit Installer Manual

This document includes guidelines for recreating the MakeFit installer, following edits to the source code. You'll need the entire folder on a Windows machine with Inno Setup, python, and py2exe installed.

Assuming no drastic changes have been made to the code we had, OpenScad should still be installed in the MakeFit folder. If not, you'll need to download it from http://www.openscad.org/ and install it into the MakeFit folder.

To create a new installer:

1. Open the Windows command line / terminal
2. Navigate to the subfolder containing setup.py, which inside the MakeFit folder
3. Type `setup.py py2exe` and press enter
   If you get a message that says access denied, do it again until it is successful or deactivate your

Figure 26: Installer Manual Excerpt