**SolidWorks DFM Plug-In and SculptPrint Feedback Application**

**Making machining knowledge more accessible to novice designers**

1. Overview

SongTelenkoDFM is a custom plug-in for SolidWorks. The plug-in was written in Visual C# inside of the Microsoft Visual Studio integrated development environment (IDE). The code makes heavy use of the official SolidWorks application programming interface (API) and use of a custom, easy-to-use SolidWorks SDK designed by Luke Malpass (<https://github.com/angelsix>, <https://www.angelsix.com/>). The plug-in itself is compiled into a dynamic-link library, or DLL (a file with a .dll extension). This DLL is loaded by SolidWorks during startup, after which the user has the option to interact with the custom plug-in in the right-hand-side sidebar.

The layout of the plug-in was written in the Extensible Application Markup Language (XAML) which is loosely similar to the common Hypertext Markup Language (HTML). The plug-in contains one XAML file (with the extension .xaml) which describes the layout of buttons and text within the user interface (UI). The XAML file makes use of the Windows Presentation Foundation, and the plug-in requires that at least .NET Framework 4.7.1 be installed. This is a key dependency, and is the reason that the plug in “looks and feels” like the Windows UI.

This project was completed under Ruoyu Song, a Ph.D. student of Dr. Cassandra Telenko’s Computational and Advancement of Sustainable Systems Lab. This project’s files are hosted on GitHub ([github.com/amarellapudi/Solidworks-DFM-PlugIn](https://github.com/amarellapudi/Solidworks-DFM-PlugIn)). The following documentation details the project installation procedure, general usage and functionality, and procedures for customizing and extending the functionality of the current code.

The black textbox below shows the hierarchy of the GitHub repository. The code for the plug-in and the researcher feedback application are contained within **“PATH\SolidWorks-DFM-PlugIn\Code\Prototypes\”**, where **PATH** refers to the directory into which the GitHub repository is downloaded. The **“Weekly Updates”** folder includes software design updates from the summer, and the folder named **“SculptPrint”** will be explained in the following section.

SolidWorks-DFM-PlugIn

├───Code

│ ├───Prototypes

│ │ ├───SculptPrint\_Feedback

│ │ ├───SongTelenkoDFM

│ │ └───SongTelenkoDFM\_Conference

│ └───Resources

| └───Icon

|

├───Information

| ├───Doorstop Screenshots

│ └───Pawn Screenshots

|

├───SculptPrint

│ ├───Experiment Files

│ ├───Parts

│ ├───Scripts

│ └───SculptPrint Feedback Tool

|

└───Weekly Updates

2. Installation

2a. SolidWorks Plug-In

All code can be downloaded from the following GitHub repository ([github.com/amarellapudi/Solidworks-DFM-PlugIn](https://github.com/amarellapudi/Solidworks-DFM-PlugIn)). The location of the Visual Studio Solution corresponding to the plug-in is **“PATH\Solidworks-DFM-PlugIn\Code\Prototypes\SongTelenkoDFM\SongTelenkoDFM.sln”**, where **PATH** refers to the initial folder into which the GitHub repository has been downloaded. Once downloaded, the code in this solution needs to be recompiled within Microsoft Visual Studio because the DLL output file can have contents specific to a particular machine.

Once the DLL is recompiled, the DLL must be registered with SolidWorks and the Windows Registry. Open “SolidWorksAddinInstaller.exe”, located in the root directory of the GitHub folder, and allow administrator access. This program should automatically find the SolidWorks executable and the RegAsm.exe location (the first two text fields). The third field must be filled with the location of the recompiled DLL. This is shown in Figure 1a below.

Finally, click the install button, and the plug-in will be registered with SolidWorks. The plug-in needs to be enabled within the SolidWorks “Add-Ins” menu before it is displayed in the sidebar. This is shown in Figure 1b below.

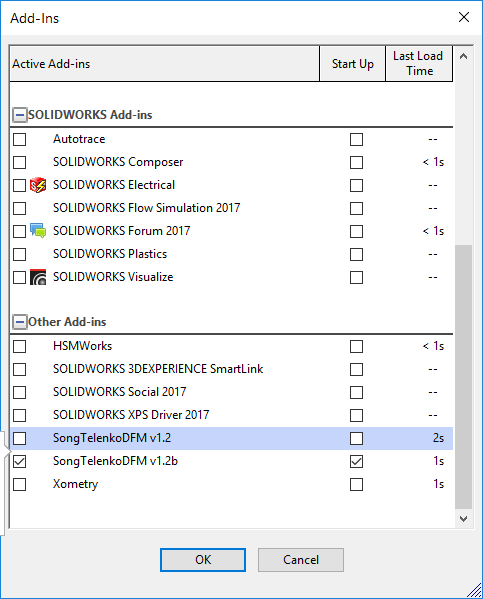
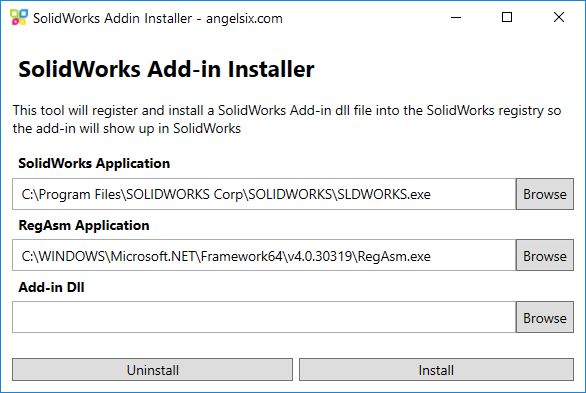


Figure 1a (left): SolidWorks Addin Installer – the third field needs to point to the compiled DLL

Figure 1b (right): Once registered using the installer, the plug in must be enabled in the SolidWorks “Add-Ins” menu

2b. SculptPrint Feedback Application

The SculptPrint Feedback application was also written in C# and XAML, and allows the researcher to communicate design feedback to the subject. The location of this application’s Visual Studio Solution is **“PATH\Solidworks-DFM-PlugIn\Code\Prototypes\SculptPrint\_Feedback\SculptPrint\_Feedback.sln”** where **PATH** is the folder into which the GitHub repository has been downloaded.

The installation of the SculptPrint Feedback Application involves the following steps:

1. Recompile the application’s code in Visual Studio
2. Using the same machine that was used to recompile the code, log into a ME-VXL virtual machine (either ME-VXL-M60-PROD or ME-VXL-P40)
3. Copy the **“PATH\Solidworks-DFM-PlugIn\SculptPrint\”** folder from the GitHub repository to the desktop of the virtual machine
4. Copy all files of the recompiled feedback application, located at “**PATH**\**Solidworks-DFM-PlugIn\Code\Prototypes\SculptPrint\_Feedback\SculptPrint\_Feedback\bin\Debug\”** into the run location on the virtual machine, **“VM\_PATH\Desktop\SculptPrint\SculptPrint Feedback Tool\”.** When done on Ruoyu Song’s account (rsong8), this location on the virtual machine is **“\\prism.nas.gatech.edu\rsong8\vlab\desktop\SculptPrint\SculptPrint Feedback Tool\”**

Steps 2-4 should be done by logging into the virtual machine using the same machine that compiled the application in Step 1. The remote-log-in software, Citrix Receiver, mounts the local computer as an external disk on the virtual machine when logged in. This makes it easy to find and copy over the files of interest in Steps 3-4.

The procedure for finding the application files on the local computer is shown in Figure 2. Once logged into the virtual machine, open an explorer window to “This PC” in the left-hand-side sidebar. Figure 2 was produced by logging into the virtual machine using the MRDC 3405 lab computer so the local computer’s storage drive is denoted with the title “Local Disk (C: on ME01W3405LAB01)”.

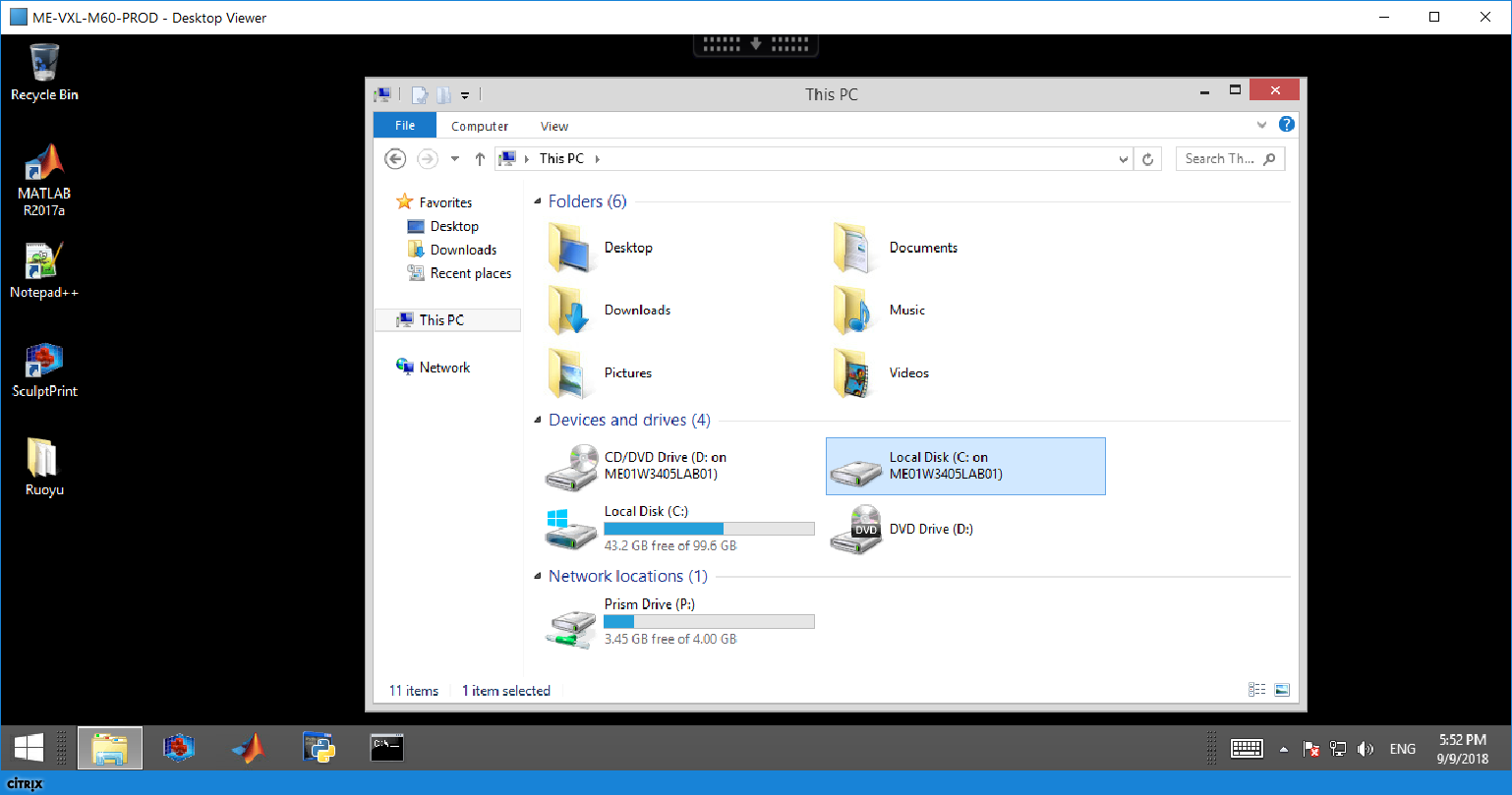


Figure 2: Windows Explorer window opened once logged into the virtual machine, showing the local machine’s main storage drive as a mounted external drive. This makes transferring files between the local machine and virtual machine easy.

3. Usage and Functionality

3a. General Usage Instructions

The SolidWorks DFM plug-in and the SculptPrint Researcher Feedback application work in coordination, feeding model-design information from the subject to the researcher, and sending design feedback from the researcher to the subject. The two software components, the plug-in and the feedback application, were designed to be able to run on two separate machines.

The general usage procedure is as follows. Each of the following steps has a corresponding figure that shows the view from the subject or researcher at the given stage in the feedback process.

1. **Figure 3a,b**

First, the researcher feedback application must be opened on the virtual machine. The feedback application runs a continuous loop, waiting for the subject to provide the design to be checked.

1. **Figure 4**

The subject opens the reference part. If the subject is unsure about the part’s design errors, the subject can immediately click “Check for Manufacturability” to send the design to the virtual machine for feedback. Otherwise, the subject can make the first round of edits and then click the same button for feedback.

1. **Figure 5**

When the subject submits the part for feedback, the researcher feedback application will automatically download the part STL file and the view-screenshots generated from the subject’s screen. The researcher now has the ability to draw rectangles around design violations and warnings in the design. When finished highlighting all errors, the researcher must send the feedback by clicking the “Submit” button in the bottom left.

1. **Figure 6**

The feedback is transferred to subject as an image that automatically opens when the researcher has submitted the feedback. The subject has the ability to close and re-open the previous researcher feedback image as a reference. The subject must then make any additional changes to fix the cited design errors. After making these changes the subject must click “Check for Manufacturability” again.

1. Then, steps 3 and 4 repeat until all of the design errors are fixed by the subject. The subject sends revised parts to the researcher who then highlights design errors and returns feedback.
2. **Figure 7**

After all design errors are fixed by the subject and the feedback from the researcher includes the green “DFM Check Passed” message box on the bottom, the subject should submit the final part. This is done by clicking the final button in the plug-in interface which is labelled “Submit Final Design”. This will create a prompt on the researcher feedback application informing the researcher that the experiment has concluded and that the feedback application will close.

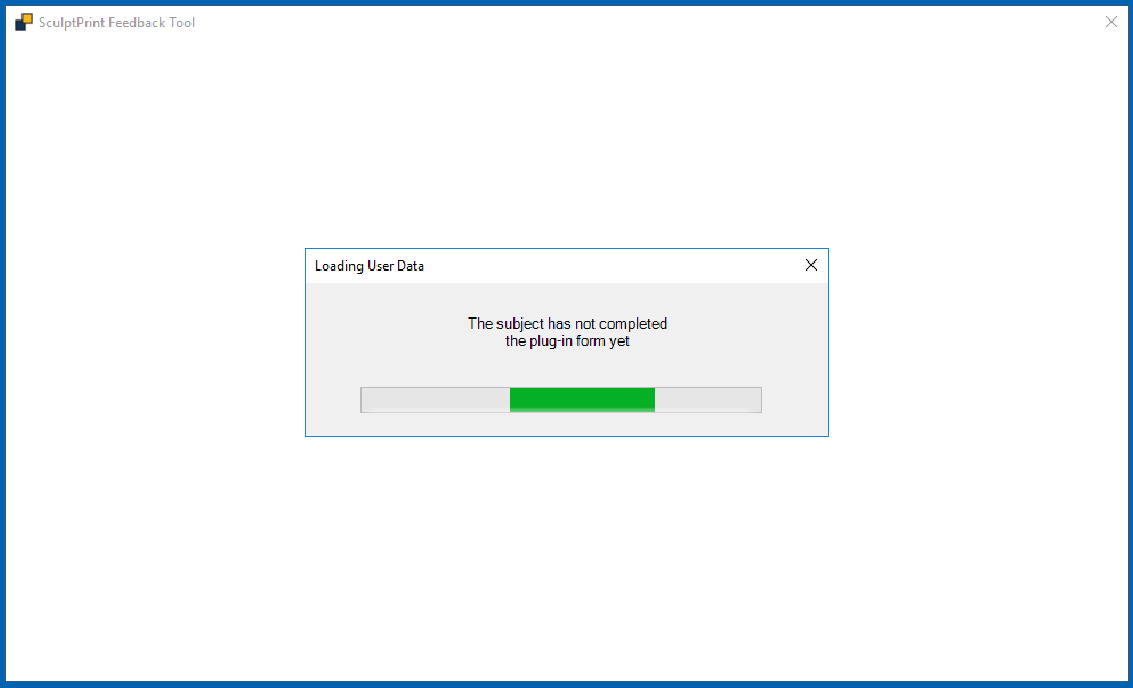
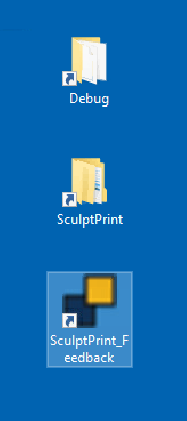


Figure 3a (left): The first step is to open the SculptPrint Feedback application on the virtual machine

Figure 3b (right): The feedback application will operate in a continuous loop until the subject submits the initial design to check

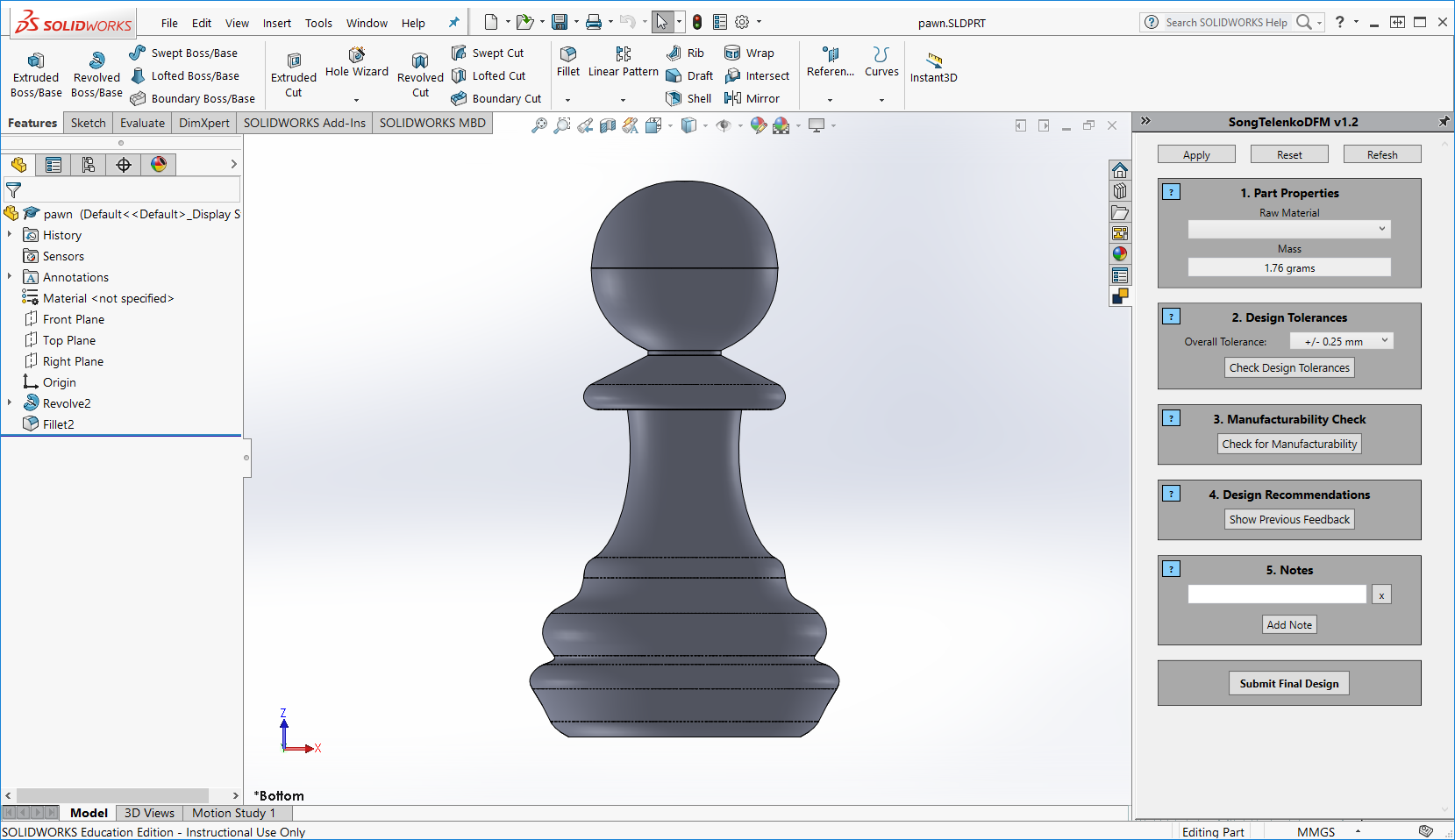


Figure 4: View of the SolidWorks plug-in when the subject first opens the reference part, in this case, the pawn. The subject has the option to make edits or leave the part as is if he/she is unsure of the design issues. Then the subject must press the “Check for Manufacturability” button to send the design to the researcher for feedback.

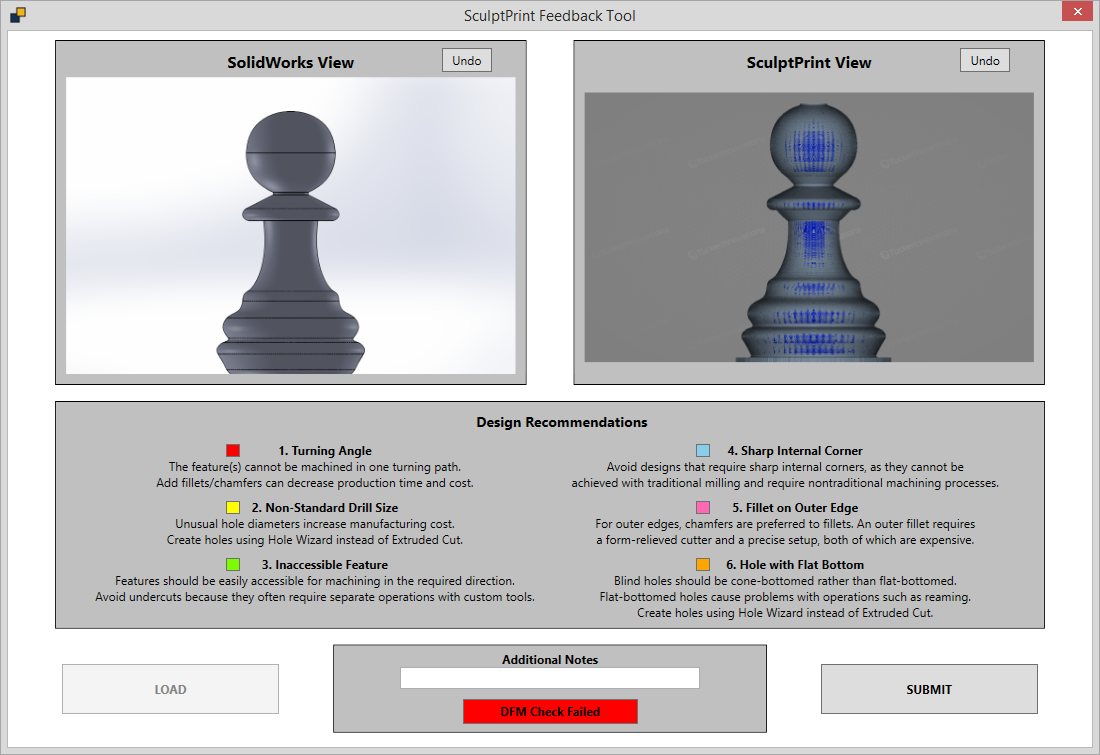


Figure 5: View of the researcher feedback application on the first manufacturability check submission by the subject. The researcher has the ability to draw rectangles around design errors and violations. Then the researcher must submit the design feedback by clicking the “Submit” button.

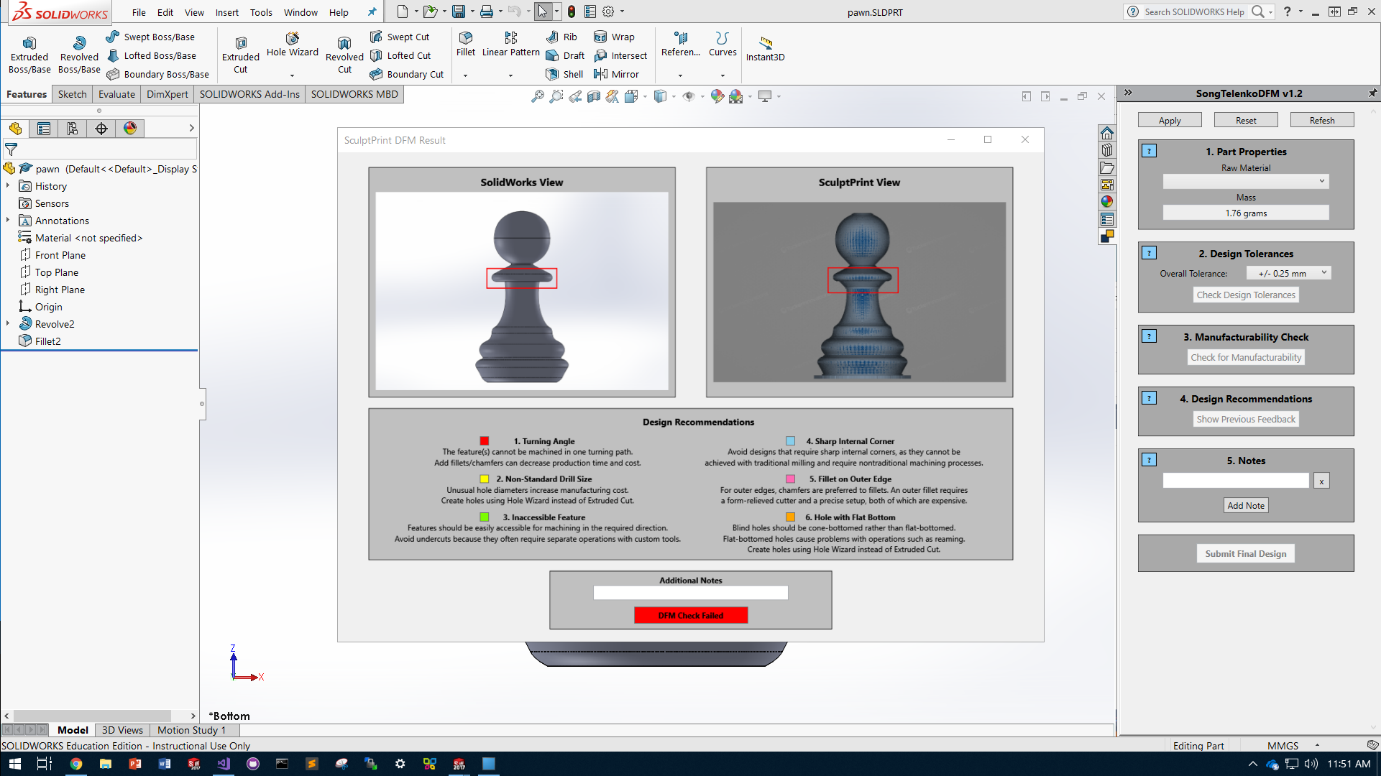


Figure 6: Feedback from the researcher is presented to the subject as an image. This window may be closed and can be reopened by clicking the “Show Previous Feedback” button. The subject must use this feedback, make changes, and then send the design back to the researcher for design feedback.

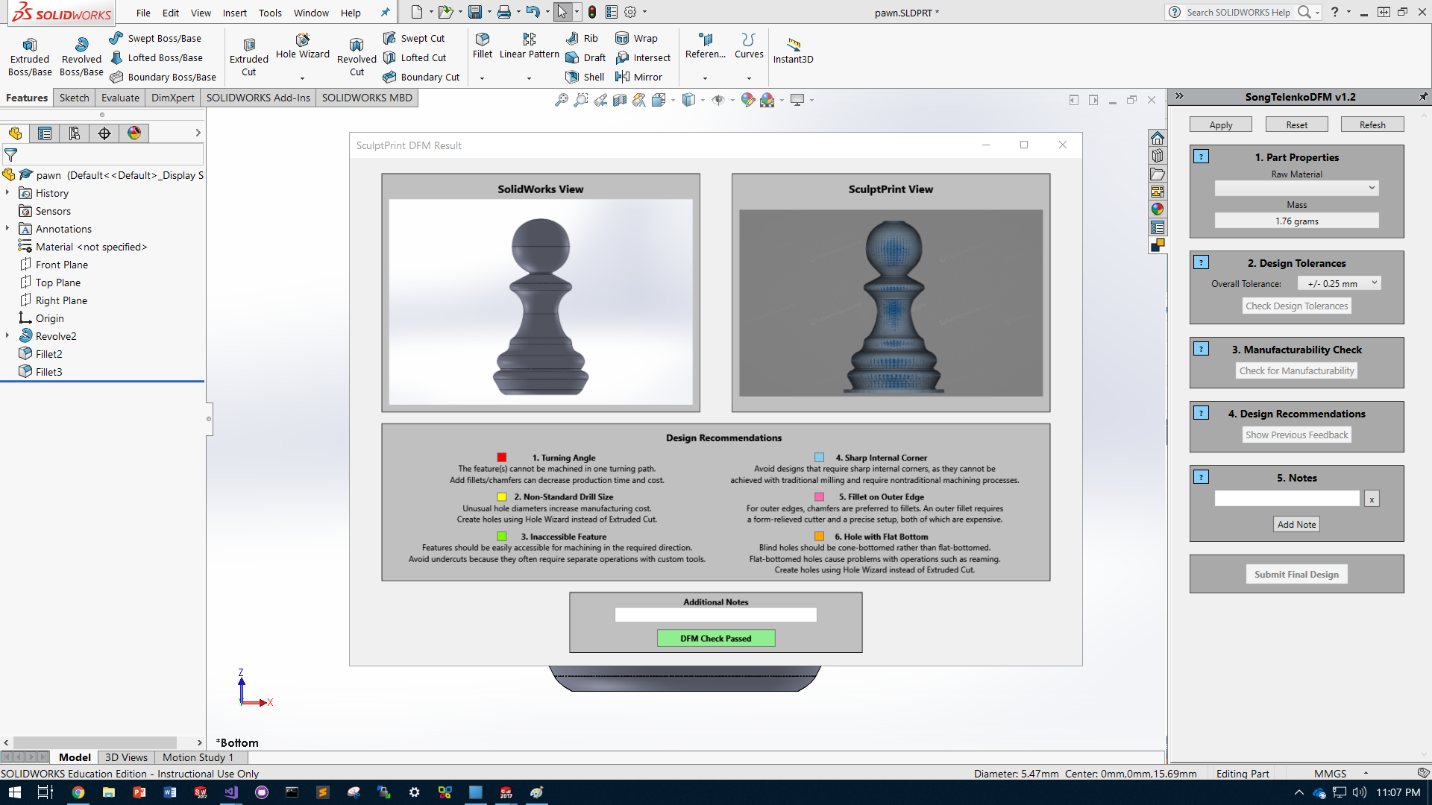


Figure 7: View of the subject’s SolidWorks window after receiving the “DFM Check Passed” message from the researcher. The subject is now ready to submit the final part.

4. Customization of Functionality and Technical Information

4a. Web Host and SFTP

Both the researcher feedback application and the SolidWorks plug-in use a third-party web host as the intermediary for file sharing. Files are uploaded to the web host by the subject when the subject clicks the “Check for Manufacturability” button. These files are downloaded by the researcher feedback app, and then placed into a time-stamped folder on the web host.

The exact list of transferred files is:

1. From the subject:
   1. **test.stl**

The part the subject is editing as an STL file

* 1. **View\_SW.png**

A screenshot of the view of the part on the subject’s screen

* 1. **DONE\_subject\_info.txt**

A file containing information about whether the part is a mill or lathe piece. This text file can be configured to contain more information that is transferred to the researcher

* 1. **DONE\_subject**

A content-less and extension-less completion flag file that is uploaded after each of the previous three files are successfully uploaded

1. From the researcher:
   1. **View\_Researcher\_Feedback.png**

A view similar to that in Figure 7 that is a screenshot of the feedback application’s view. This serves as the design feedback to the subject

The webserver is currently a domain with the name “marellapudi.com” and the hosting service is provided by SiteGround. To log in, use the username “apollome” and the password “Aniruddh.123” in an FTP or SCP client. All experiment files are located in **“/public\_ftp”.**

This will need to be migrated by January 2019, as this web space will expire by this time. A very similar web server can be set up to continue experimentation with this software, and would cost roughly $50 for 12 months. Alternatively, there may be a way of getting web space through Georgia Tech IT, which would likely be free.

4b. SolidWorks Plug-In

The SolidWorks plug-in was written in Visual C# inside of Visual Studio. The following YouTube instructional series, developed by Luke Malpass, is an incredibly useful tool to start learning how to use the SolidWorks API (<https://www.youtube.com/playlist?list=PLrW43fNmjaQVMN1-lsB29ECnHRlA4ebYn>). Additionally, Aniruddh Marellapudi (am123@gatech.edu) will be a point of contact for support when specific SolidWorks API functionality knowledge is needed.

The software overall is organized into class (.cs) files and layout-markup files (.xaml). The plug-in’s button layout and UI is defined in “UI\_SolidWorks\_SideBar\_PlugIn.xaml”. Some background in HTML or XML may be helpful to parse some of the structures, but the basic idea is that all UI elements are defined as containers. High-level containers include the <Grid>, a grid defined with a given number of rows and columns, and the <StackPanel>, a container that simply stacks its children items vertically. Within these high-level containers are buttons and text fields (both for input and output). Buttons can be configured to call functions when clicked. The blue buttons containing question marks do not call functions but rather display informational tooltips to guide the user through the UI. The text fields come in two types, the <TextBlock> and the <TextBox>, with the difference being that <TextBox> allows the user to type into a field, while <TextBlock> is simply read-only text that is displayed at a given location. Overall, Visual Studio makes it easy to learn XAML because of the intuitive, double-window UI (Figure 8) that lets the programmer see the changes to the XAML code instantly.

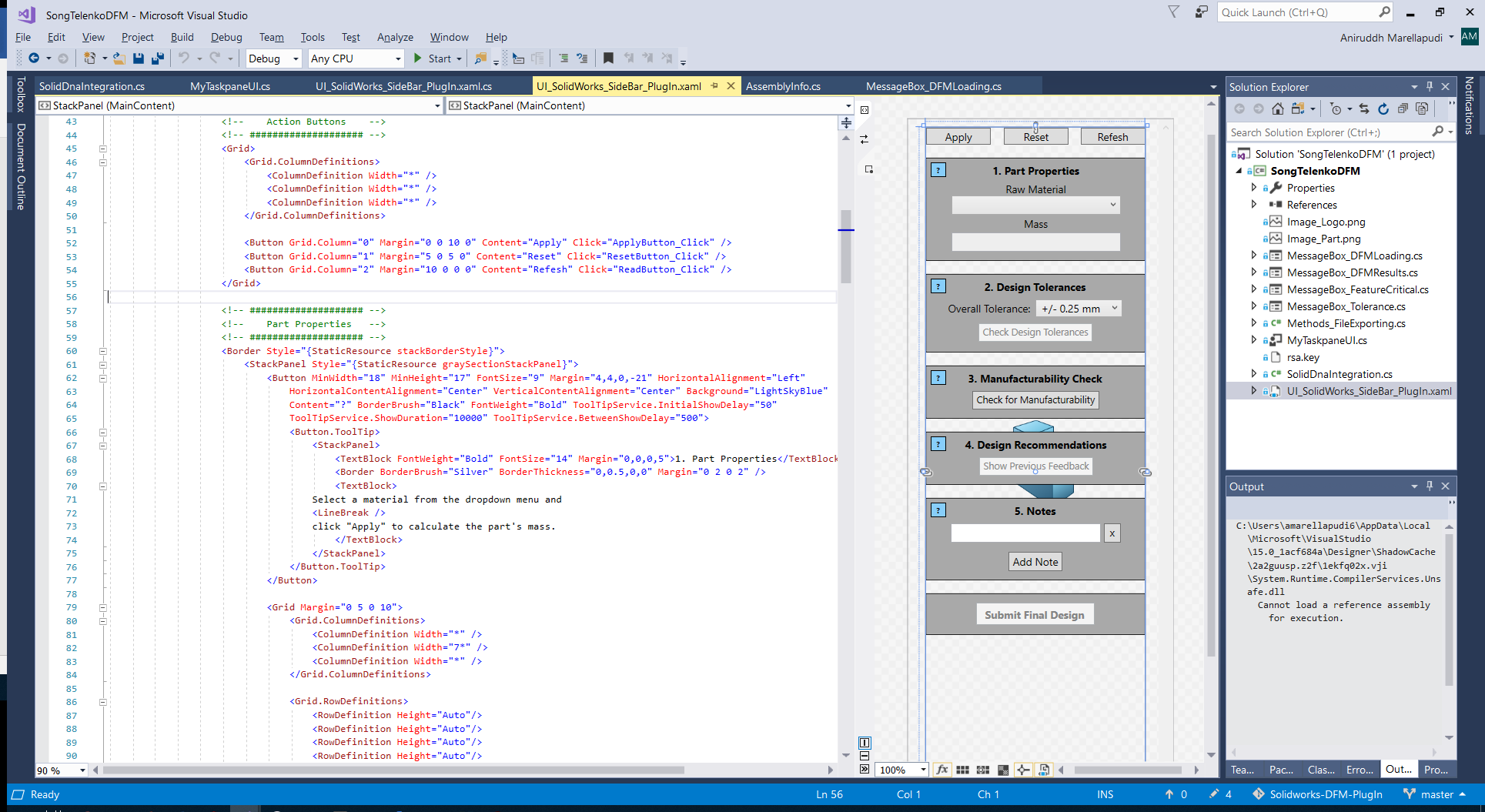


Figure 8: Editing the main SolidWorks plug-in XAML file in Visual Studio. Changes to the XAML code can immediately be seen in the preview window to the right.

The logic and functionality underlying the UI is contained in “UI\_SolidWorks\_SideBar\_PlugIn.xaml.cs”. This is a long class, so the recommended method to understand its functionality is to read the code, and then run the code in debugging mode so that reader can step through the various function calls. The class includes interaction logic for the XAML UI, meaning that the functions corresponding to the button click events defined in the XAML UI are defined in this class. In addition, this class handles SolidWorks Custom Property maintenance, SFTP file management, and several SolidWorks-specific functions that read information from the part and the overall SolidWorks environment.

In addition to the interaction logic underlying the XAML UI, many of the button click events create dialogs of their own. For example, when the “Check for Manufacturability” button is clicked, a small window with a loading bar opens and prevents the user from interacting with the SolidWorks window before the feedback is downloaded from the researcher (this is called a modal window). These classes are named to have “MessageBox\_” as a pre-fix. Each MessageBox dialog contains a design file and interaction log of its own, which can be viewed by left-clicking the class name and selecting “View Code”.

An understanding of this code is key to moving this project forward, and Aniruddh Marellapudi (am123@gatech.edu) will be the point of contact to help with any confusing or unclear aspects of the code as it currently stands.

4c. SculptPrint Feedback Application

The feedback application is built using the common Windows Presentation Foundation (WPF) application framework. Similar to the plug-in, the feedback application has a UI that is defined in XAML (MainWindow.xaml), a class that defines the interaction logic (MainWindow.xaml.cs), and several MessageBox classes that create dialogs when certain button click events occur.

The basic framework for understanding the operation of the feedback application is similar to that for the plug-in. The programmer should read the code and run the code in debugging mode to follow function calls.