

Building Seesaw around a pipeline we can inspect

Our October 3D-printing work connects established engines, validates the handoffs, and keeps the remaining printer checks visible.



By Edwin Rodriguez

Published Oct 14, 2026 · 4 min read

October launch collection · Story context 2026-10-02



A 3D-printing application has a lot to get right between opening a model and producing a file a printer can consume. Geometry, supports, layer images, exposure settings, encoding, and export all have their own failure modes.

[Seesaw's October foundation](#) starts with an inspectable chain of existing tools. The Python-driven Ubuntu workspace uses PrusaSlicer for geometry preparation and slicing, then UVTools for resin-layer inspection and encoding. The app's job includes making those handoffs understandable and checking the result.

The software pipeline has been tested; the printer and material combinations still need physical testing. Start with calibration geometry and inspect the output before taking on a real print.

Draw the pipeline before the interface

For the resin path, the documented chain is:

```
STL model
-> PrusaSlicer preparation and slicing
-> SL1 layer archive
-> UVTools inspection and encoding
-> validated PM4N file
-> USB transfer and printer review
```

Each arrow is a contract. The next tool needs a particular kind of input. UVTools' role in this integration begins with sliced layer data; treating it as a complete STL-to-printer replacement would skip important preparation work.

Drawing that chain first gives the UI a useful vocabulary. A model is loaded. A slice is prepared. Layers are inspected. An export is validated. Those states should be visible rather than collapsed into a single "Done" indicator.

Reuse upstream work with specific versions

Seesaw currently pairs PrusaSlicer 2.9.4 with UVTools core 7.0.1. Version pinning makes a result easier to reproduce and gives a bug report a clear starting point.

It also creates maintenance work. When an upstream version changes, revisit the interface assumptions and rerun representative fixtures before replacing the pinned engine. A new executable appearing in a folder doesn't establish that the product's input, output, and cancellation behavior still match.

Credit and licensing belong in the package as well as the article. Existing open-source engines represent substantial work. A wrapper should preserve the relevant notices and make its own contribution clear.

Keep the user's settings attached to the result

A model's scale, placement, copies, printer selection, and material settings affect the output. If those inputs change after a successful slice, the old result needs to lose its ready-to-export status.

That may sound obvious until an interface lets someone rotate a model while still showing a green export button for the previous layers. Link readiness to the exact inputs that produced the result.

For an editable project, preserve the user's intended settings. On reopening, require the appropriate validation before calling a fresh output ready. Saving the project and validating a printer file are separate user actions with separate evidence.

Inspect the encoded result

A successful backend exit is one signal. Readback gives another: open the generated file and check the properties that matter, including layer data and settings.

The useful engineering habit is to compare the final representation with the intended one. If a conversion silently changes layer height or exposure, the pipeline can appear successful while producing an unwanted result.

Create small positive and negative fixtures. Include a model that works, a layer problem that must block export, and an input that exceeds the application's supported bounds. Keep failure messages tied to a stage so the person reviewing the job knows where to look.

Start hardware work conservatively

Software checks help catch predictable mistakes, but resin, printer condition, material behavior, and physical calibration introduce additional variables. Seesaw's documentation keeps that remaining work explicit and directs initial hardware work toward calibration geometry.

Keep a small job summary: input revision, engine version, exit result, output digest, and the checks that passed. When a file looks wrong, that summary helps you trace it back to the preparation that produced it.

For developers, the broader lesson is useful far beyond printing. Build around clear stage contracts, preserve exact inputs, inspect final outputs, and give upstream projects credit. The result is a workflow a person can understand when it succeeds and investigate when it doesn't.

Sources & further reading

Checked 2026-10-14 · Story context 2026-10-02

1. [Seesaw source and current scope](#) (2026-10-14)
2. [Seesaw backend validation](#) (2026-10-14)
3. [PrusaSlicer](#) (2026-10-14)

[Read the current article online →](#)

Support the workshop

Explore software and learning offers at [StackSocial](#). Affiliate link: ProgreTech may earn a commission from purchases. Check current price, license and renewal terms before buying.

Find our projects at [progretech.com](#).

ProgreTech LLC · Build / Learn / Improve / Together

This file contains its text, illustrations and fonts. No account, scripts or network connection required to read. External reference links need internet access.