

GETTING THE USER'S JOB DONE

Empathy, iteration, and self-learning in the library

Brian Zelip

Emerging Technologies Librarian

University of Maryland, Baltimore

zelip.me/talks/the-job.pdf



**Today's education and research
needs more from us than
text-based services and resources.**

**Historically, the “IT” group has been
the library’s front lines for the cutting edge.**

**Now, more and more (solo) librarians
are working at the cutting edge as well.**

- Bioinformatics
- Data services
- Digital humanities
- Health and life sciences
- Journalism
- Makerspaces
- etc.

Innovation Space



Upcoming Workshops:

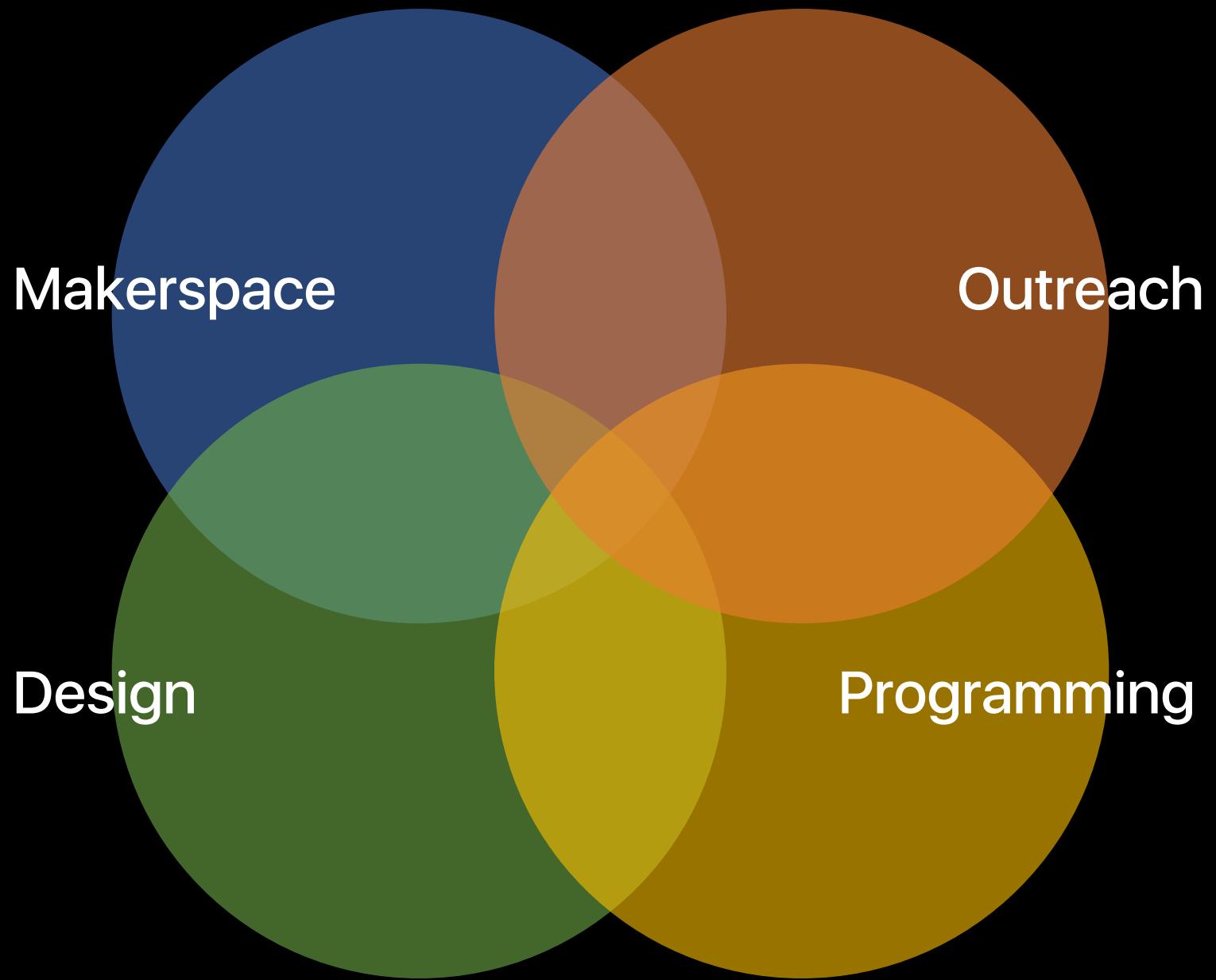
Introduction to
3D Printing
Aug 4, 9 12p-1p

Introduction to
3D Modeling
July 28 > 12p-1p
Aug 8

What's Available
Lulzbot Taz 5
Makita Drill Driver 2x

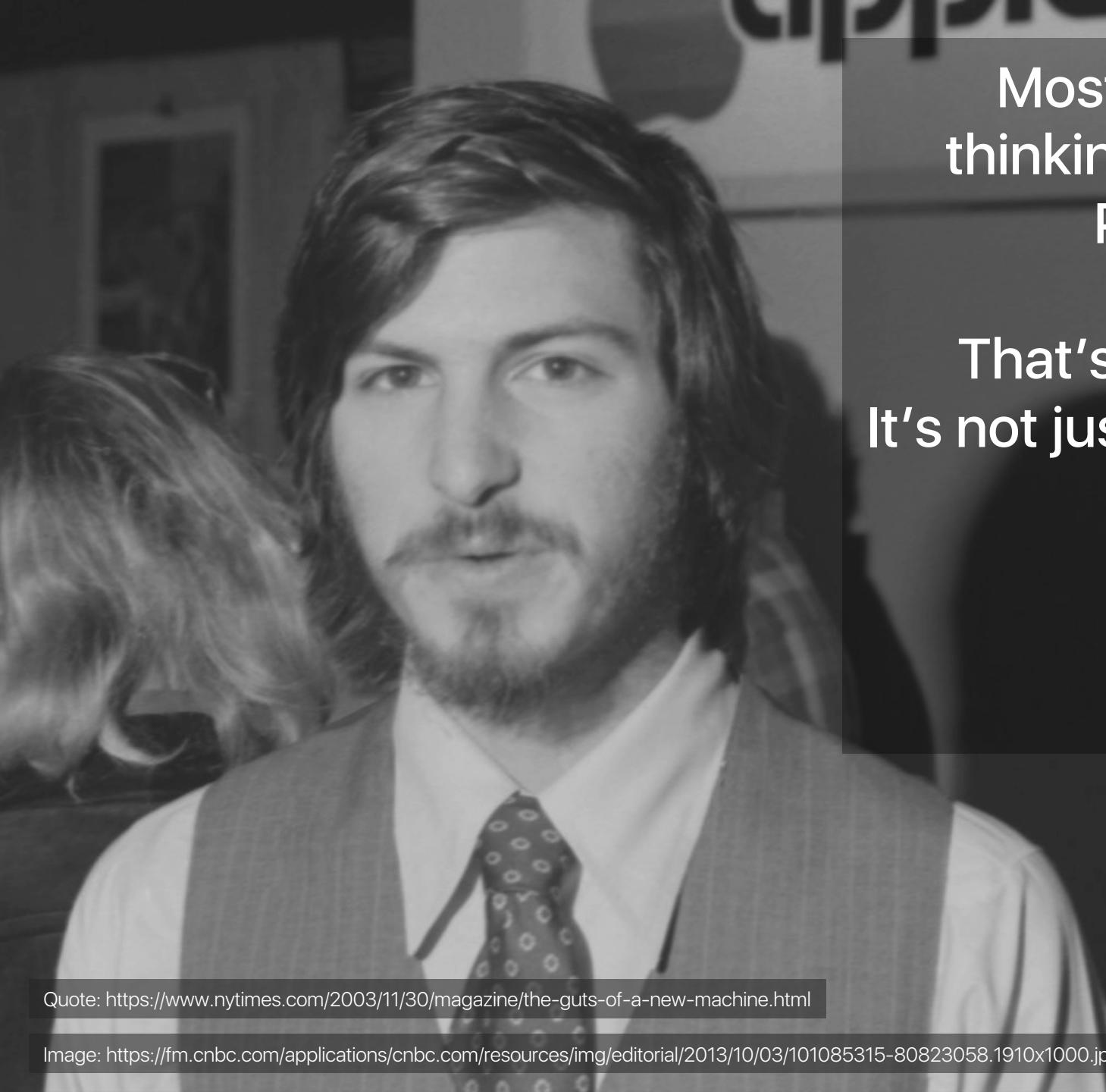
Tutorials from
Lynda.com

Great learning resource
that offers 3,500 video
tutorials by experts.
Topics range from 3D
modeling to computer
programming.



Design

**Component
thinking**



Most people make the mistake of
thinking design is what it looks like.
People think it's this veneer...

That's not what we think design is.
It's not just what it looks and feels like.

Design is how it works.

Steve Jobs

Design

**Component
thinking**



**Do One Thing
and Do It Well**

The Unix philosophy

Service design thinking

Services should be:

- designed based on customer needs rather than the internal needs of the business.
- designed with input from the users of the service
- prototyped before being developed in full

Design thinking

- Empathy: *understanding the needs of those you're designing for*
- Ideation: *generating many ideas*
- Experimentation: *testing those ideas with prototyping.*



Design thinking

- Empathy: *understanding the needs of those you're designing for*
- Ideation: *generating many ideas*
- Experimentation: *testing those ideas with prototyping.*

Trying CSS Grid for the first time (Adam Argyle)



Browser Preview (file:///Users/mpj/codetemp/fff-twitch-overlays/nav.html) — fff-twitch-overlays

```
1 <html>
2   <head>
3
4
5   </head>
6   <body>
7     dasjasdhkjasdh
8     dsajdkasljadskljsd
9     <nav>
10       <picture>
11         <img src="" alt="">
12       </picture>
13       <a href="#">Stream starting soon</a>
14       <a href="#">Left desk only</a>
15       <a href="#">Remote fika</a>
16       <a href="#">Remote coding</a>
17       <a href="#">BRB</a>
18       <a href="#">BYE</a>
19     </nav>
20   </body>
21 </html>
```

Browser Preview (file:///Users/mpj/codetemp/fff-twitch-overlays/nav.html) — fff-twitch-overlays

dasjasdhkjasdh dsajdkasljadskljsd
[Stream starting soon](#) [Left desk only](#) [Remote fika](#) [Remote coding](#) [BRB](#) [BYE](#)



mpjme



argyleink



**Failing together
at programming**

**Live every Monday
7-9AM Pacific Time
twitch.tv/funfunfunction**

Job to be done theory

- People buy products and services to get a “job” done
- Jobs are functional, with emotional and social components
- Success comes from making the “job”, rather than the product or the customer, the unit of analysis

You have [users],

they need your help,

they often have no idea how to describe what they need,

they have no idea how to evaluate what they're looking at,

and most importantly, they have no way of knowing whether or not they're working with the right [librarian].

@monteiro

Quote: <https://vimeo.com/88764543>

Image: <https://chifoo.org/wp-content/uploads/2017/12/Mike-Monteiro-April.png>



Tachyons, et al

Even before learning to code, I paid attention to the open source world because it provided me insight into solving my own problems. Paying attention long enough eventually taught me how to code.

I like paying attention to the programming world, and using its ideas and tools, because it helps inform other aspects of life. It shows how to look for abstraction points, pain points of repetition, pain points of a lack of documentation about an idea or a project, pain points of where we're inaccessible to our users [SEE COLOR-BASED INFORMATION on A T – "the UX inside of me yells out DON'T USE ONLY COLOR TO COMMUNICATE MEANING!"]

React

A JavaScript library for building user interfaces

[Get Started](#)[Take the Tutorial >](#)

Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

Component-Based

Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

Learn Once, Write Anywhere

We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using [React Native](#).

Components Basics

[Base Example](#)

[Reusing Components](#)

[data Must Be a Function](#)

[Organizing Components](#)

[Passing Data to Child Components
with Props](#)

[A Single Root Element](#)

[Listening to Child Components](#)

[Events](#)

[Emitting a Value With an Event](#)

[Using v-model on Components](#)

[Content Distribution with Slots](#)

[Dynamic Components](#)

[DOM Template Parsing Caveats](#)

Components In-Depth

[Component Registration](#)

[Props](#)

[Custom Events](#)

[Slots](#)

[Dynamic & Async Components](#)

[Handling Edge Cases](#)

Components Basics

Base Example

Here's an example of a Vue component:

```
// Define a new component called button-counter
Vue.component('button-counter', {
  data: function () {
    return {
      count: 0
    },
    template: '<button v-on:click="count++>You clicked me {{ count }} times.</button>'
  }
})
```

JS

Components are reusable Vue instances with a name: in this case, `<button-counter>`. We can use this component as a custom element inside a root Vue instance created with `new Vue`:

```
<div id="components-demo">
  <button-counter></button-counter>
</div>
```

HTML

```
new Vue({ el: '#components-demo' })
```

JS

[Introduction](#)[Getting Started](#)[Next.js](#)[Gatsby](#)[Create React App](#)[React Static](#)[Webpack](#)[Parcel](#)[Zero](#)[Migrating from v0 to v1](#)[Playground ALPHA](#)[Guides](#)[Syntax highlighting](#)[Live code editor](#)[Table of contents](#)[Writing a plugin](#)[Custom loader](#)[Wrapper customization](#)[Vue ALPHA](#)[Advanced](#)

MDX

Markdown for the component era

MDX is an authorable format that lets you seamlessly write JSX in your Markdown documents. You can import components, such as interactive charts or alerts, and embed them within your content. This makes writing long-form content with components a blast .

Try it

Hello, world!

Below is an example of JSX embedded in Markdown.

Try and change the background color!

This is JSX

```
# Hello, *world*!
```

PRINCIPLES

Responsive

Everything should be 100% responsive. Your website should work regardless of a user's device or screensize.

Readable

No matter the lighting, or the device, font-sizes should be large enough and contrast should be high enough for your users to easily read your content.

Modular

Tachyons isn't just a monolithic framework. It's a collection of small modules that can be mixed and matched or used independently. Use what you need. Leave the rest.

Accessible

Accessibility is important to us. Throughout both the css and the documentation we provide numerous tools and methods for making it easier to maximize the accessibility of your site.

Performant

Code should be optimized for performance.

Reusable

Clear documentation helps create a shared understanding of design patterns amongst your team. This helps promote reuse and reduces the amount of redundancy in a codebase.

FEATURES

Open source component library

There is a [growing library of open source components](#) written in static html that are easy to use as is, customize with your own theme, or port to a templating language.



Search or jump to...

Pull requests Issues Marketplace Explore



css-modules / css-modules

Watch 207

Unstar 12,023

Fork 392

Code

Issues 71

Pull requests 10

Projects 2

Wiki

Insights

Branch: master ▾

css-modules / README.md

Find file Copy path



juanca Documentation on `from global` (#270)

9922d6e on Oct 27, 2017

16 contributors

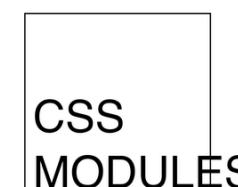


146 lines (95 sloc) | 5.41 KB

Raw

Blame

History



CSS Modules

A **CSS Module** is a CSS file in which all class names and animation names are scoped locally by default. All URLs (`url(...)`) and `@imports` are in module request format (`./xxx` and `../xxx` means relative, `xxx` and `xxx/yyy` means in modules folder, i. e. in `node_modules`).

CSS Modules compile to a low-level interchange format called ICSS or [Interoperable CSS](#), but are written like normal CSS files:

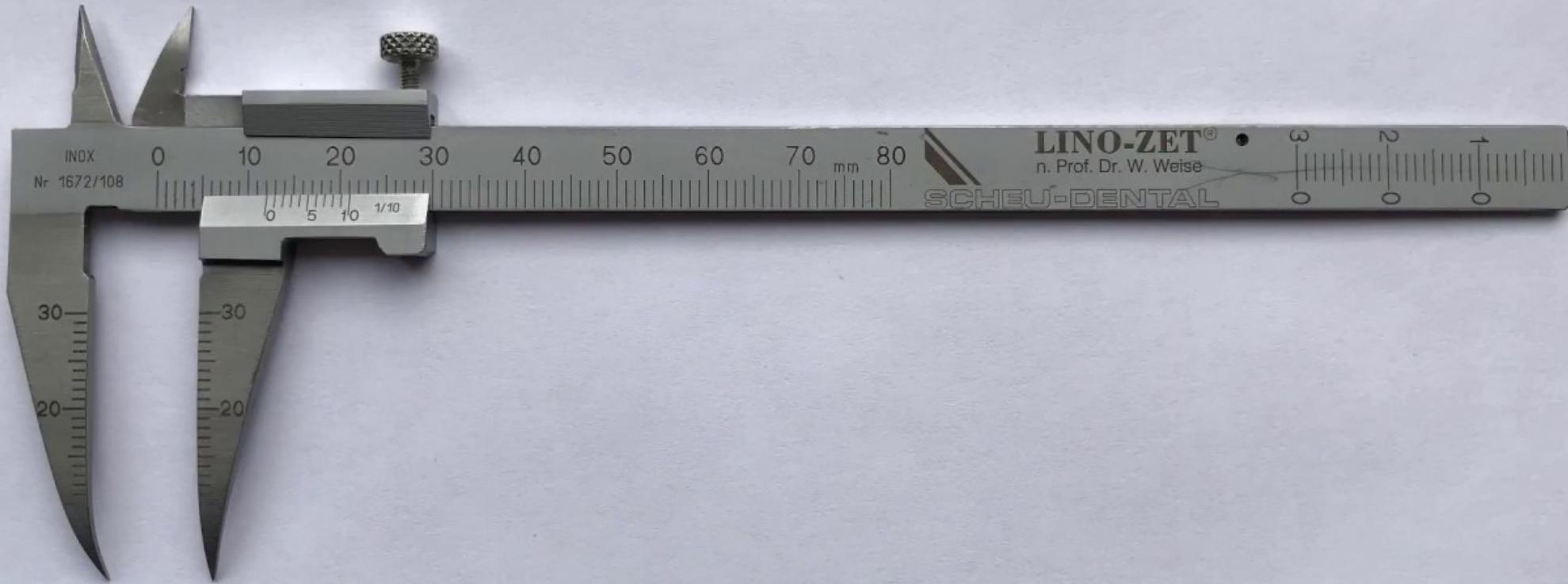
**Hands-on prototyping:
iteratively design, build, and test with users**

Users dream it



Getty

we make it



**TIN
KER
CAD**

Making Dreams Come True demo



What's New



TOP



Workplane



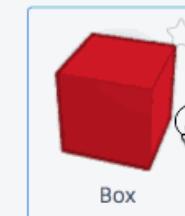
Ruler

Tinkercad
Basic Shapes

Box



Cylinder



Box



Cylinder



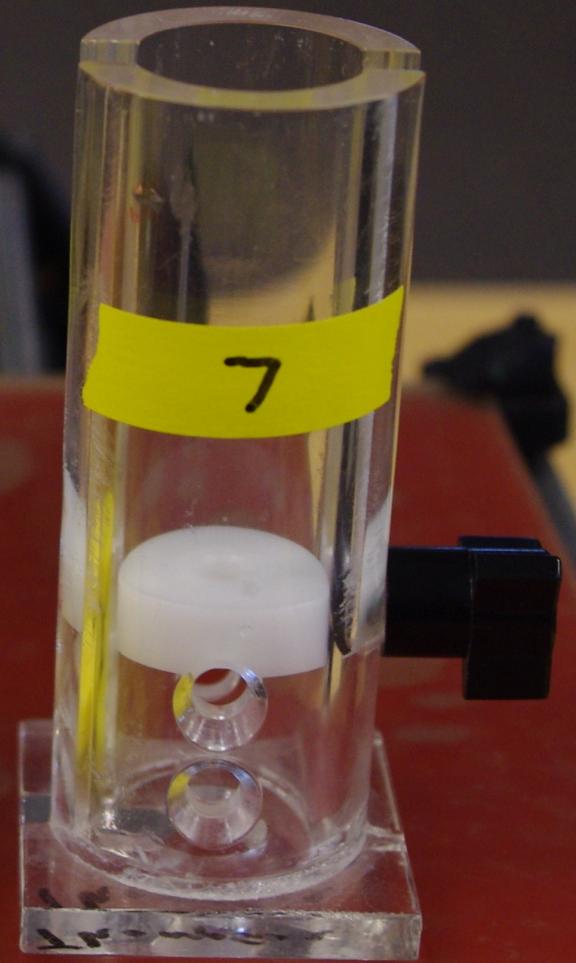
Sphere



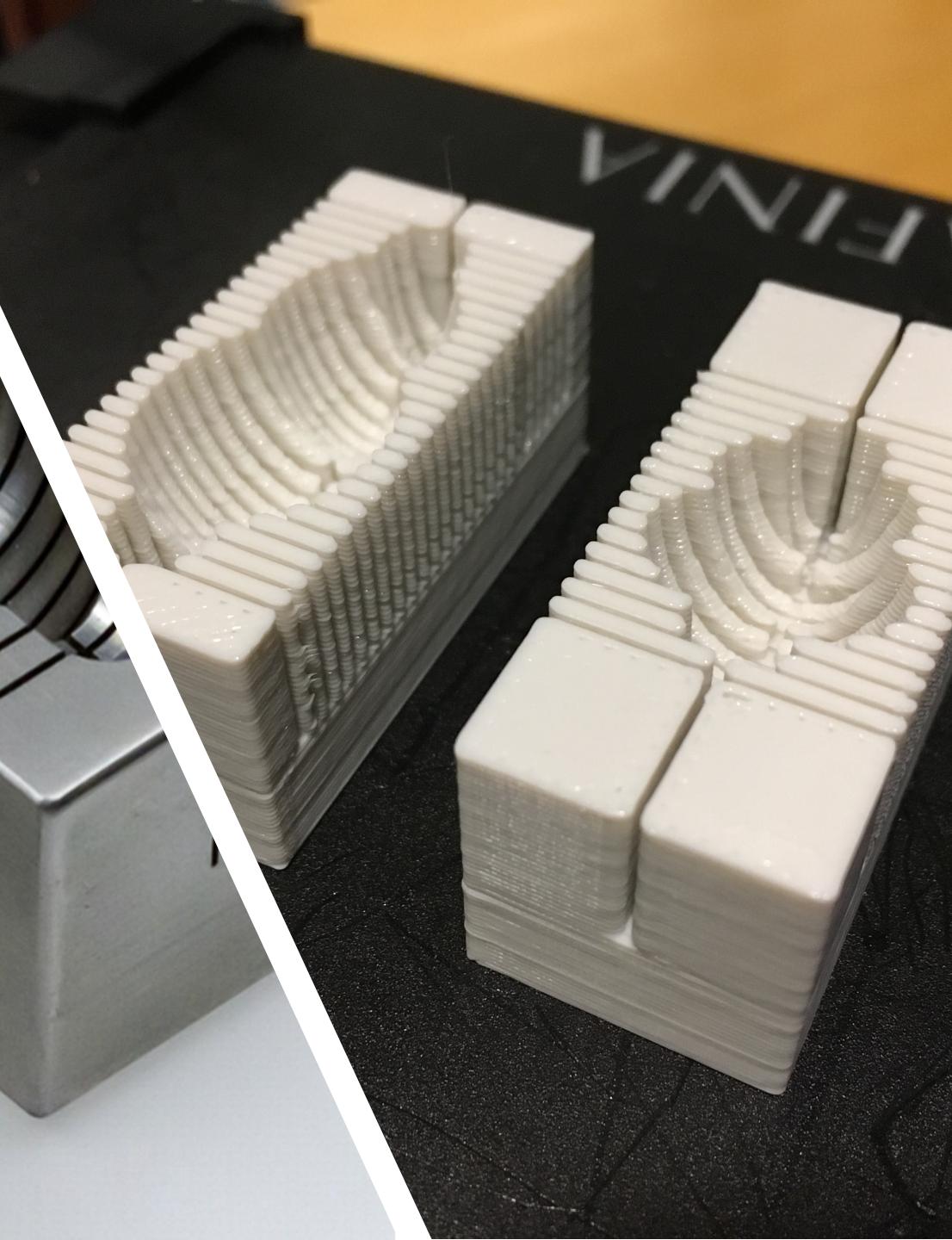
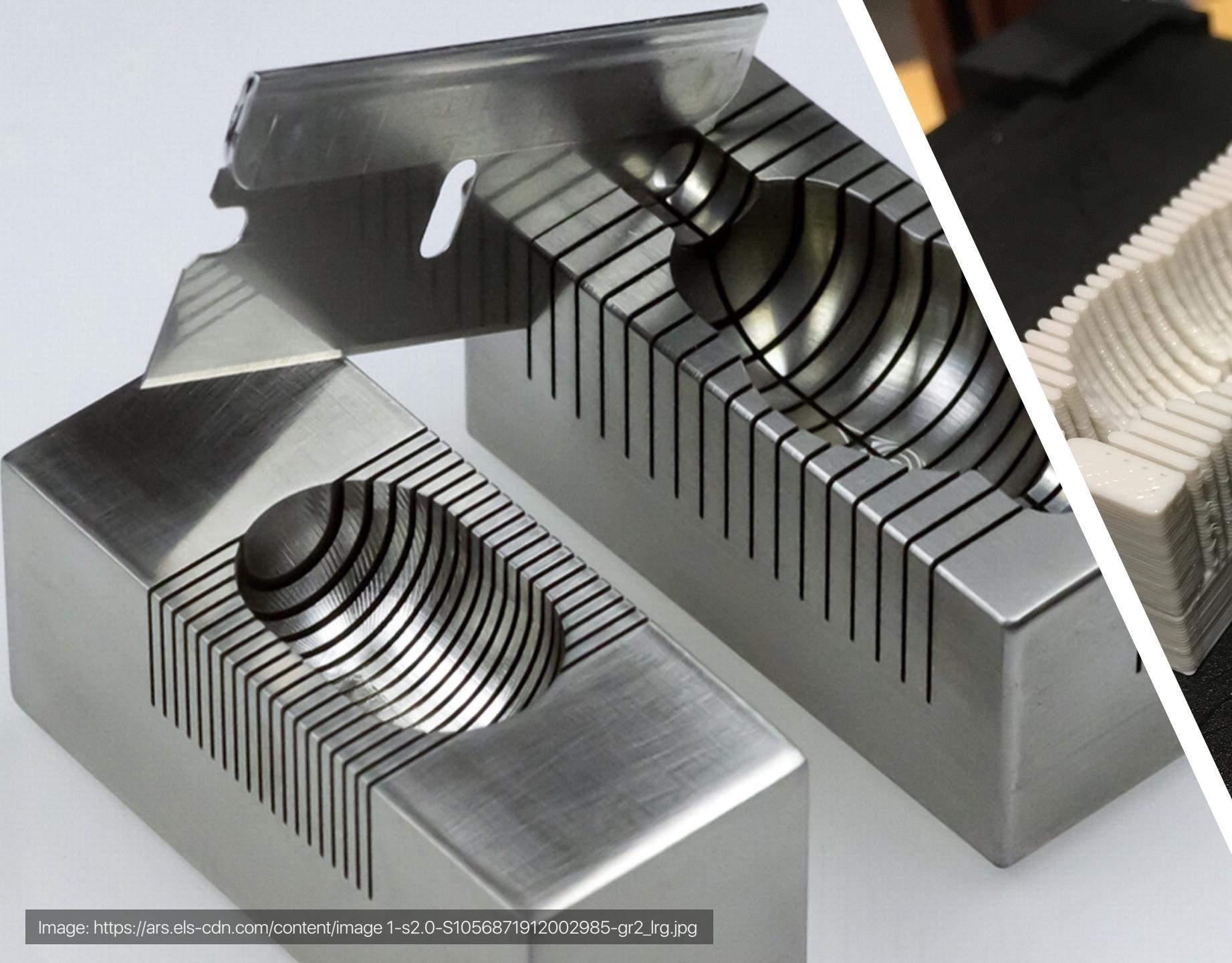
Scribble

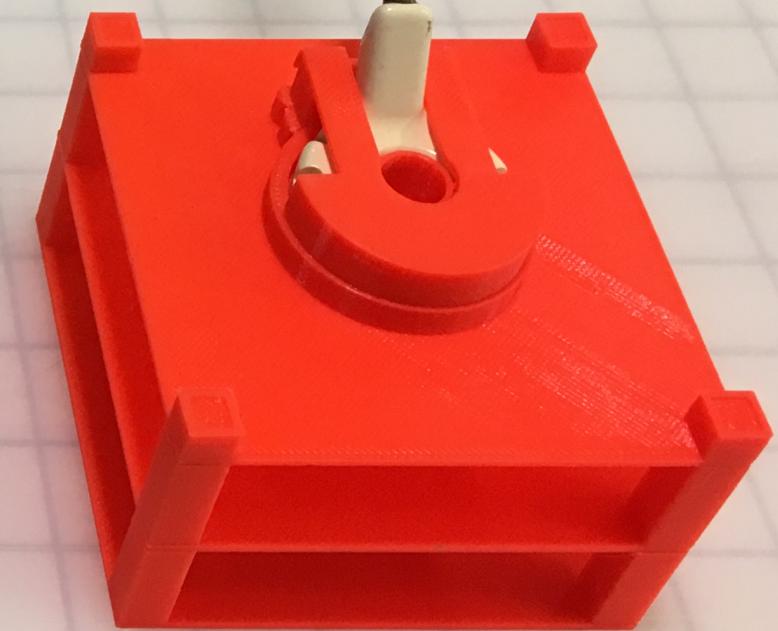
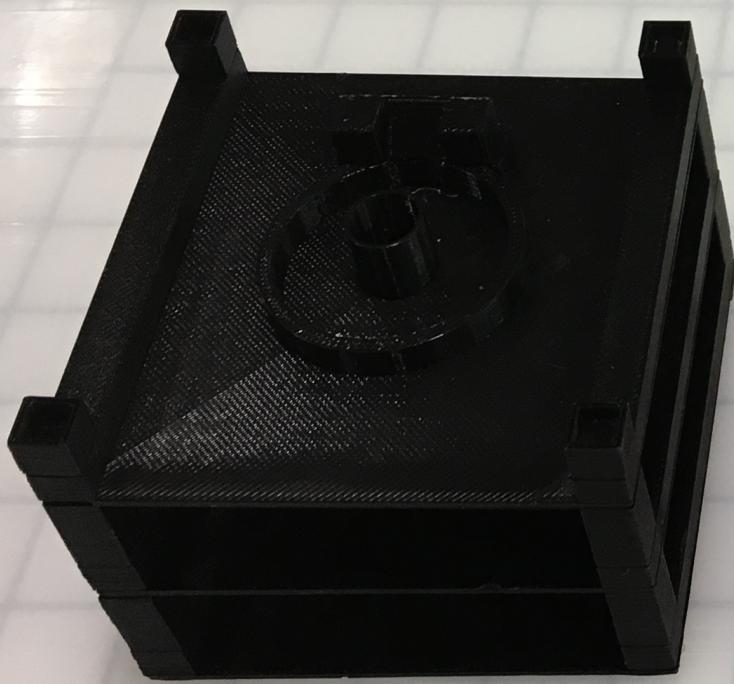
Edit Grid

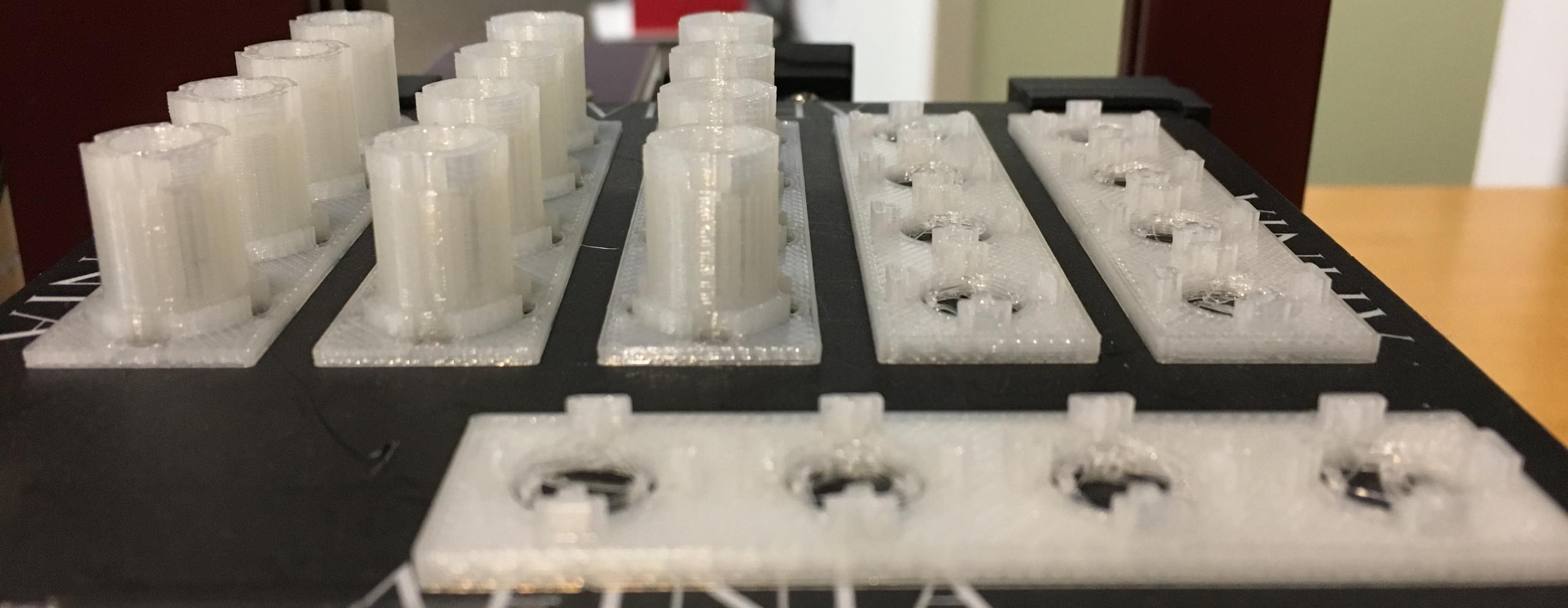
Snap Grid 1.0 mm ▲

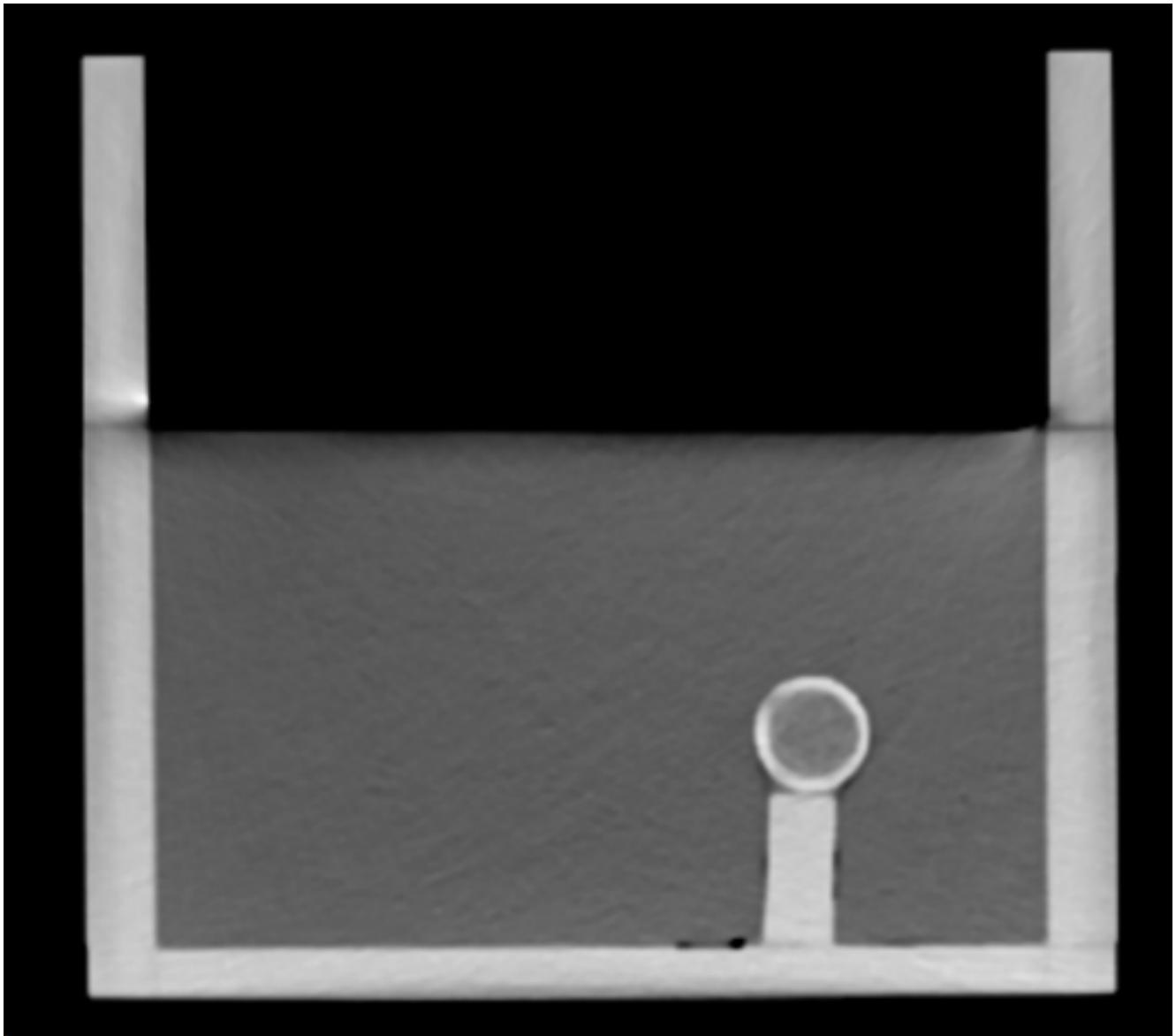
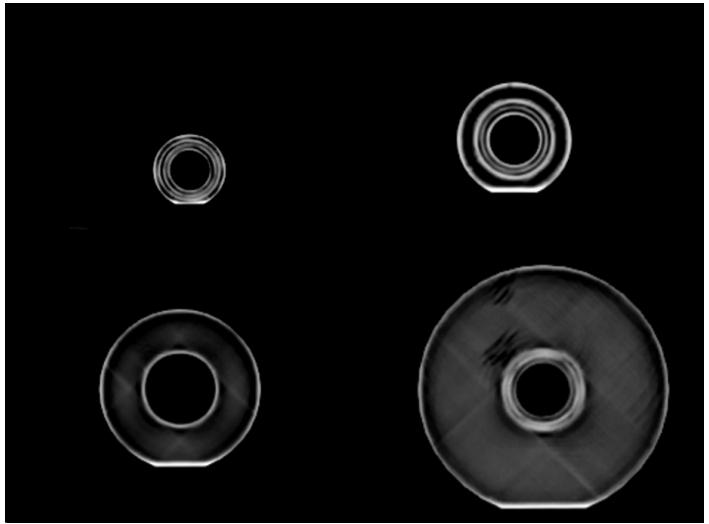












www.hshsl.umaryland.edu/ispace/#newsletter

HS/HSL

innovation space

Newsletter



**This talk is about successfully meeting
the increasingly cutting edge needs
academic librarians face.**

Empathy

Iteration

Design

Component thinking

Life-long learning

A wide-angle photograph of a suspension bridge, likely the Chesapeake Bay Bridge, at sunset. The bridge's towers and cables are silhouetted against a sky filled with warm, orange and yellow hues. The calm water in the foreground reflects the bridge's structure and the vibrant colors of the sunset.

THANK YOU

zeli.p.me/talks/the-job.pdf