A Computer Engineering Approach To Design For 3D-Printing Manufacturability

Professor Henry (Hank) Dietz

Session 3A, 2:15PM, July 31, 2019

University of Kentucky
Electrical & Computer Engineering
What Is 3D Printing?

Subtractive Building:

“Every block of stone has a statue inside it and it is the task of the sculptor to discover it.”

– Michelangelo

Additive Building:

“The whole is greater than the sum of its parts.”

– Aristotle
What Is Design For Manufacturability (DFM)?

Design product so that it is easy to manufacture.

- Lego doesn’t easily do curves...
- Some methods don’t easily do unsupported...
  - Extrusion: Fused Deposition Modeling (FDM, aka FFF)
  - Material Jetting (MJ), Drop On Demand (DOD)
- Some methods don’t easily do cavities...
  - Stereolithography (SLA and DLP systems)
  - Selective Laser Sintering/Melting (SLS/SLM, also EBM)
  - Binder Jetting (BJ)
- Can decompose into parts made separately
What Is A Design?

- A 3D drawing of an object isn’t sufficient
  - Material, tolerance, & other constraints
  - Functional requirements (e.g., processors)
  - **Means to adjust the design for DFM**

- We suggest **a design should be a program**:
  - Parameterized (e.g., by tolerances)
  - Structured, hierarchical, & composable
- Programs can be automatically transformed
How Is This Different?

• Normal process for 3D printing:
  1. Create design by drafting in CAD system
  2. Convert design into “portable” STL file (polygonal surface patches)
  3. Slice STL into G code X,Y,Z,E movements

• Proposed process:
  1. Create parametric design as a program
  2. Compile design + parameter values into DFM-optimized machine-specific design
  3. Convert design into G code (STL optional)
Designs As Programs

• Not really a new idea
  – **G code** is a low-level program
  – Most CAD systems *internally* specify a design as a program composing solids

• Leverage what we know about programming
  – Language design, programming practices
  – Parameters & selection of DFM options
  – **Compiler optimization** technology: “correctness-preserving transformations”
An Example Using *OpenSCAD*

difference() {
    scale([0.5, 1, 2])
    sphere(d=100);
    translate([0, 0, 20])
    rotate([30, -115, 0])
    cylinder(d1=80, d2=20,  
          h=100, center=true);
}


How About A Base Fitting This?

- Make this a module:
  ```
  module statue() {
  ...
  }
  ```

- Make a base module too
- Just difference ‘em:
  ```
  difference {base(); statue();}
  ```

A printer-dependent tolerance between them for best fit?
module tol(xt=defxt, yt=defyt, zt=defzt) {
  for(c=[0:1:$children-1]) minkowski() {
    children(c); scale([xt, yt, zt]) cylinder();
  }
}

difference() {base(80); tol() statue(80);}
difference() {base(); tol() statue();}
difference() {base(); tol(yt=2) statue();}
A Manufacturability Example

- The Unified Thread Standard (UTS) specs a 30° angle for screw threads.
- Most FDMs can’t print that without droop.
- So, replace 30° angle with a printable one...

45° is safe.
Could allow for droop...
A 3D-Printed UTS-Compatible Thread

Lens adapter M42 x 1mm pitch to Sony E

on $225 printer, 0.25mm layers!
Another Manufacturability Example: **Spanless Hinges**

- There are many types of hinge, but most require some type of trapped pin... which generally implies an unsupported span.

- This doesn’t... the 45° angle is its own inverse and is self-supporting (and can print-assembled):
3D-Printed Spanless Hinges
Multiple Substitutions

- In 2016, researchers at the Hasso Plattner Institute made “metamaterial pliers”: a single part with stiffness, spring, & bending hinge.

- Our metamaterial version has a **spring** and a **spanless hinge** and it works...
Status & Future Work

- A design should allow DFM transformations
- Optimizing compiler technology can transform designs expressed as programs
- Creating the library of transformations is hard, we welcome collaborators
Other Stuff We’re Doing

- Quantum computing Education & Research In Kentucky – QERKY.ORG
- Optimizing / parallelizing compilers
- Nanocontrollers to cluster supercomputing
  (we built the world’s 1st Linux cluster back in 1994)
- Computational photography