Prototyping

Table of contents

Prototyping as experimentation 1
Four axes 1
Paper prototyping 2
Wireframes and Wizard of Oz 2
The economics of design 2
Have code generation tools killed wireframing? 3

Prototyping as experimentation

Prototyping forms a key part of the design cycle, can be **exploratory or evaluatory**, and — the point most often missed — **does not have to start with writing code**.

It sits inside the design loop:

```
Establish customer problems → Design value propositions
        → Prototype value propositions → Evaluate value propositions → MVP
```

So a prototype is a way of running an experiment (experimentation[1]) about a value proposition, before committing to the MVP[2].

Four axes

Prototyping can be thought of along four axes:

[1] courses/comp1100/experimentation.pdf
[2] courses/comp1100/minimum-viable-products.pdf

Axis	Range
Representation	Online vs offline; high vs low fidelity
Precision	Level of detail
Interactivity	Fixed or open
Evolution	Rapid to iterative

Placing a prototype on these four axes is a more useful question than "is it good enough yet?" — it says what the prototype is *for*.

Paper prototyping

> An offline, interactive, rapid method of exploring aspects such as layout, information architecture, interactive design and more.

Low cost, easy to do, interactive, and fun. It's the default first move, and it deliberately scores low on representation and precision while scoring high on interactivity and speed.

Wireframes and Wizard of Oz

- **Digital prototyping using wireframes** — online, still low fidelity, focused on structure rather than looks.
- **"Wizard of Oz" prototyping** — the user experiences a working system; a human is doing the work behind the curtain. Lets you test a value proposition that you have no idea how to build yet.

The economics of design

The reason to start cheap:

	Low fidelity prototypes	High fidelity prototypes	Code prototypes	MVP
Cost	Super cheap	Quite cheap	More expensive	Most expensive
Focus	Flow & interactivity	Look & feel, usability	Code feasibility and usability	Getting to market

Each step right costs more and answers a narrower question. Spending code-prototype money to learn something a paper prototype would have told you is the mistake this table exists to prevent.

Have code generation tools killed wireframing?

A genuinely open question the seminar puts both sides of:

Yes?	No?
Quicker to generate interfaces than to draw boxes	Wireframes help focus on user experience: "what are they trying to build?"
The gap to code prototypes is smaller	LLM designs are more mediocre
Real interactions instead of simulated ones	Wireframes are *intentionally* low fidelity — to get feedback on the user **experience**, not the user **interface**

The strongest argument in the right column is the last one: low fidelity isn't a limitation to be overcome, it's what stops your interviewee commenting on the colour scheme instead of the flow. See prompt-engineering[3] for the tooling this question is about.

[3] courses/comp1100/prompt-engineering.pdf