Test Doubles

Improving Developer Confidence

Jeffry Gonzalez - Hypertheory

Who Am I?

- Software Developer
- Educator
- https://github.com/hypertheorytraining
- https://github.com/jeffrygonzalez
- https://www.hypertheory.com
- jeff@hypertheory.com

Circa 1983

My Agenda

- Helping you gain confidence in a super hard way to make a living.
- Repenting for my own arrogance and the sins of my “youth”.

What I Am Not Teaching

- We are using Angular. This has nothing to do with Angular.
- We are using .NET. This has nothing to do with .NET
- We are learning about testing. I am not teaching automated testing.
    - However, if there was a part 2, *almost* everything we do here would become the basis for your automated tests - and make them much more *meaningful.*

Coding Interview

Sample Question

Questions:

- "Who Won?"
- "Who Lost?"

Name	Score
Jeff	127
Violet	87
Henry	92
Stacey	212

Coding Interview

Sample Question

Questions:

- “Who Won?”
- “Who Lost?”

Name	Score
Jeff	127
Violet	212
Henry	92
Stacey	212

Coding Interview

Sample Question

Questions:

- "Who Won?"
- "Who Lost?"

Name	Score
Jeff	150
Violet	150
Henry	150
Stacey	150

Coding Interview

Sample Question

Questions:

- "Who Won?"
- "Who Lost?"

Name	Score
Jeff	300
Violet	300
Henry	300
Stacey	300
Jeff	200

Coding Interview

Sample Question

Questions:

- "Who Won?"
- "Who Lost?"

Name	**Score**
`Null`	300
Violet	Avocado

Our Job

- Coding is fooling around and finding out.
- Mapping a domain of inputs to correct outputs
- Our job is the output - anticipating (and controlling) inputs

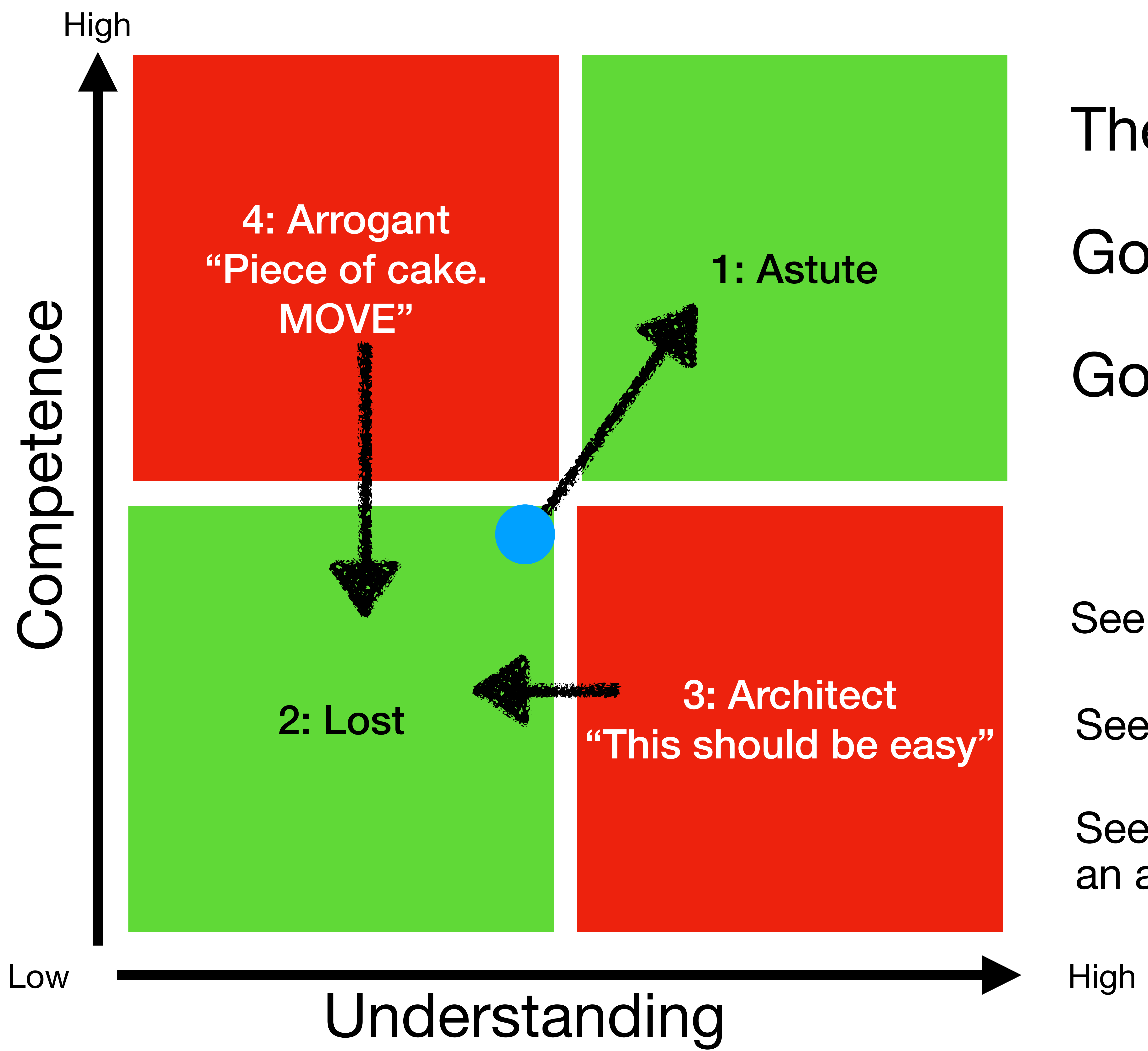
Confidence

Some Definitions

- “The state of feeling certain about the truth of something”
- In coding (at least) based on your *competence* and *understanding*
- **Competence**: Skillz. How to write the code, familiarity with language, framework, library, patterns, process, etc.
- **Understanding**: Having clarity about the thing you are building, how it should look, work, “feel”. The *vibes*.

Your Confidence Zone

For Any Given Task, Find Your Zone And Work From There



The blue dot is the path to being astute.

Going from 4 - 1: Enterprise Fizzbuzz

Going from 3 - 1: ChatGPT Wrote This Mess

See: https://martinfowler.com/bliki/BeckDesignRules.html

See: “Make it Right Then Make It Good”

See: “Everyone who has said to you ‘I have this great idea for an app, just need someone to code it up’”

In Other Words

- Code at the lowest common denominator between your understanding and competence.
- If you don't have much understanding, write simple, dumb, obvious code until you have understanding.
  - No “layers”, no “clean code”, no patterns, no Span<T> would be better here, none of that.
- If you don't have much competence, fake it until you make it.
  - What would it look like if it *did* work?
  - Write in plain text if you have to. Use a drawing tool.
  - Do **not** start slinging code you don't understand using stuff you copied from StackOverflow, other code, CoPilot, whatever.
  - Talk it through with someone (or your imaginary friend)

No Silver Bullet

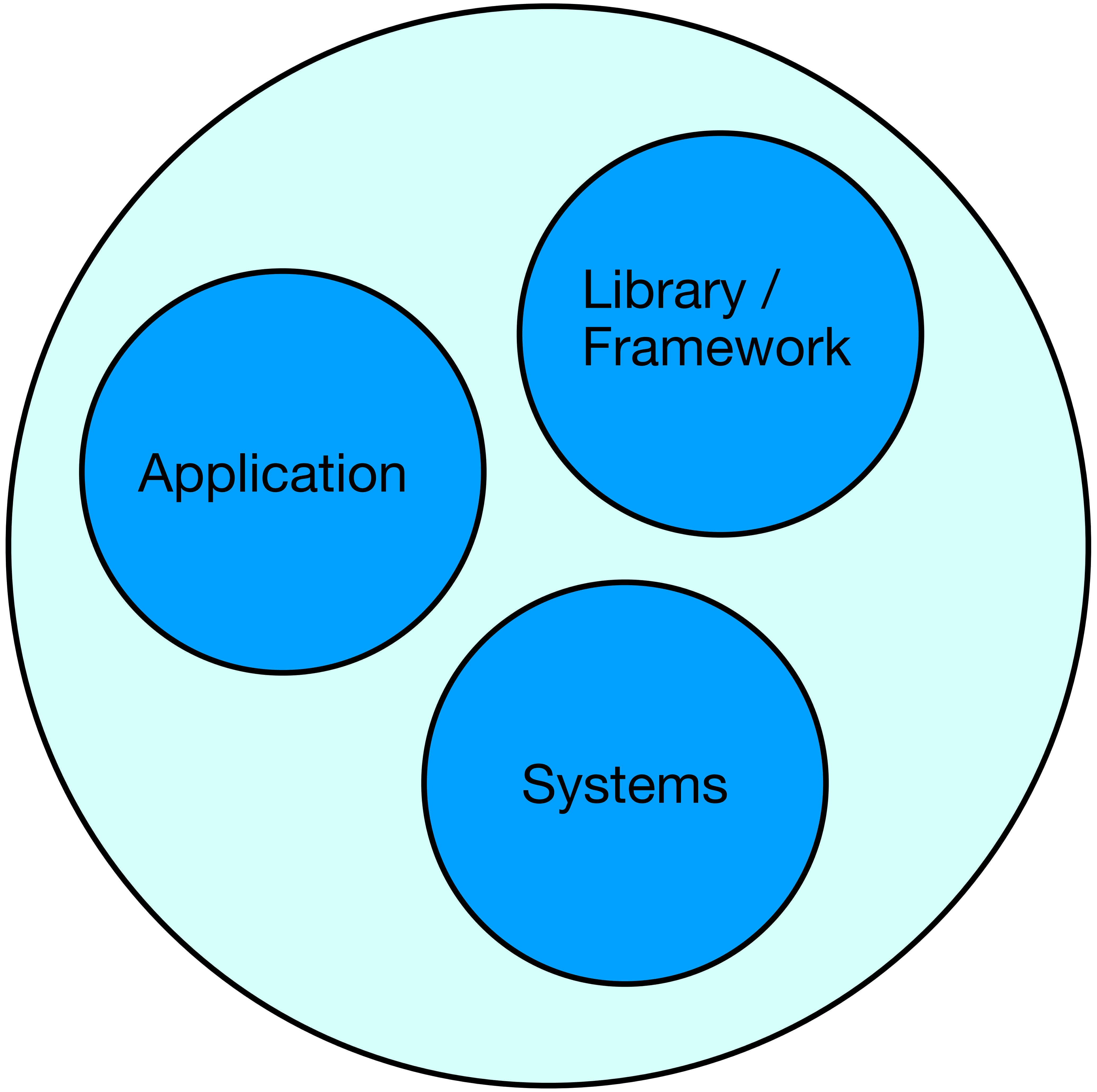
Fred Brooks - 1988

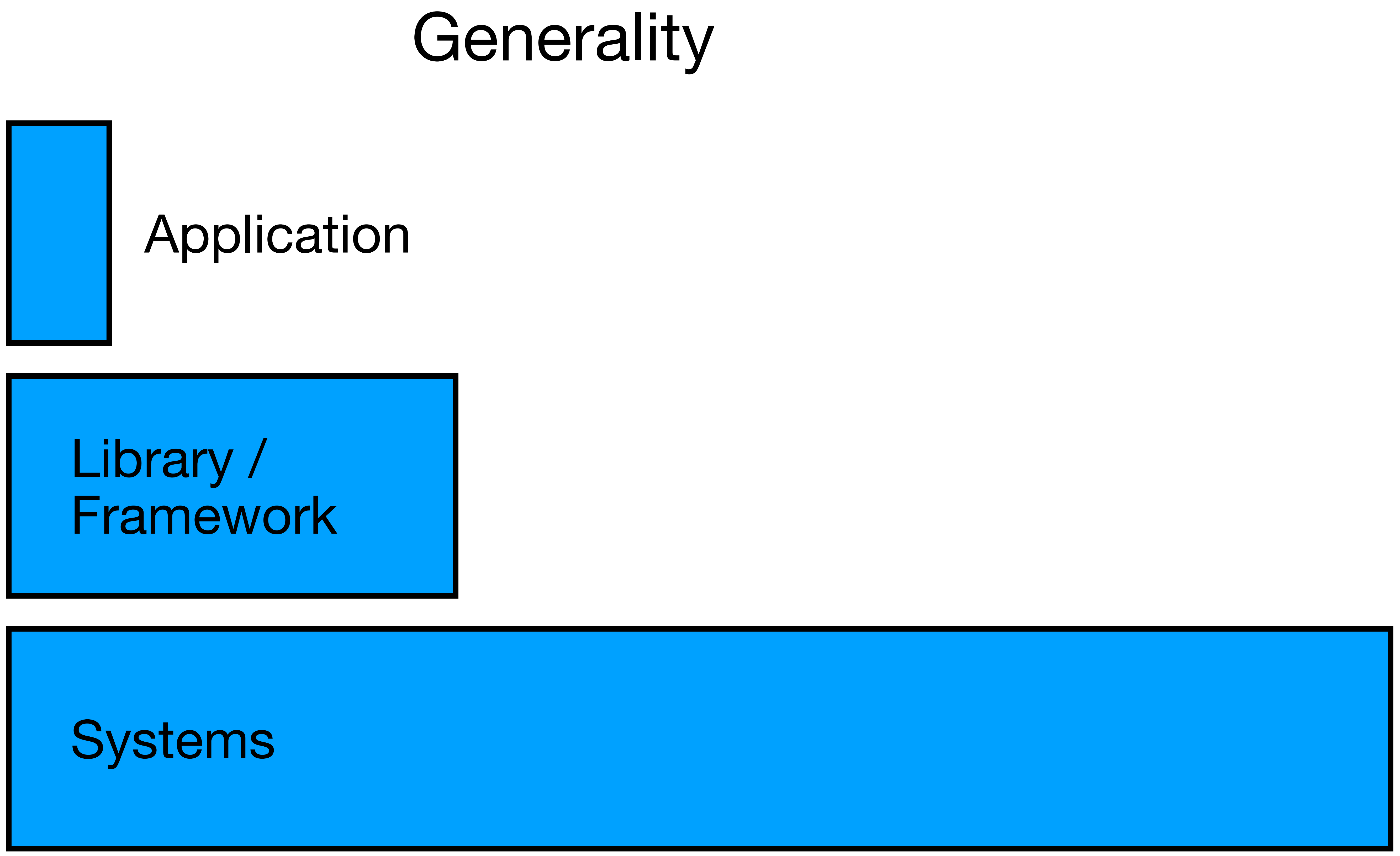
- Coding is always going to be hard because the hard thing is a moving goal post, and it is about thinking conceptually and abstractly.
- Recommendations:
    - Don’t write code. Use something off the shelf.
    - Rapid Prototyping
    - Grow Software Organically
    - Train new developers to think conceptually and abstractly

https://www.cs.unc.edu/techreports/86-020.pdf

Spheres

A Rough Categorization Of "Software Development"





Testing Guidance

Approach Depends on Sphere

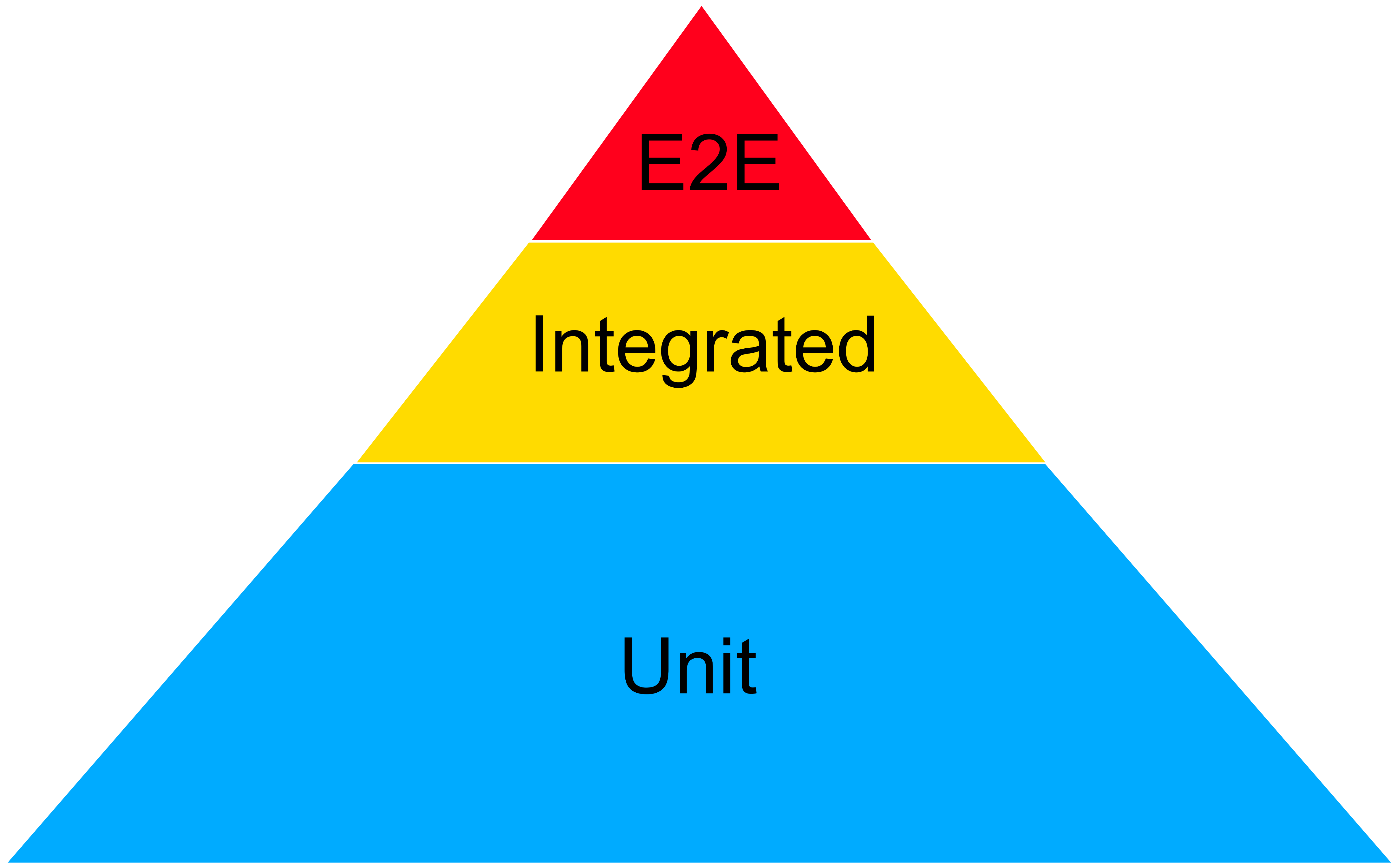
Test with the finest grained mechanism that tells you something important

https://jeremydmiller.com/2012/10/11/test-with-the-finest-grai/

What is important as a systems programmer, a library/framework programmer, or an application developer is not the same thing.

The Test Pyramid

Gigachad Advice for Software Developers



You are a gigachad programmer if you have lots of low-level unit tests, some integration tests, and a few e2e tests.

“Simulating productivity through the reproduction of the system and values I am part of.”

Backing Services

Integrating With Other People’s Services

- Late Bound
- Could be “shared language” (e.g. SQL, OIDC/Oauth2, etc.)
- Could be “contract” based (HTTP with OpenAPI, etc.)

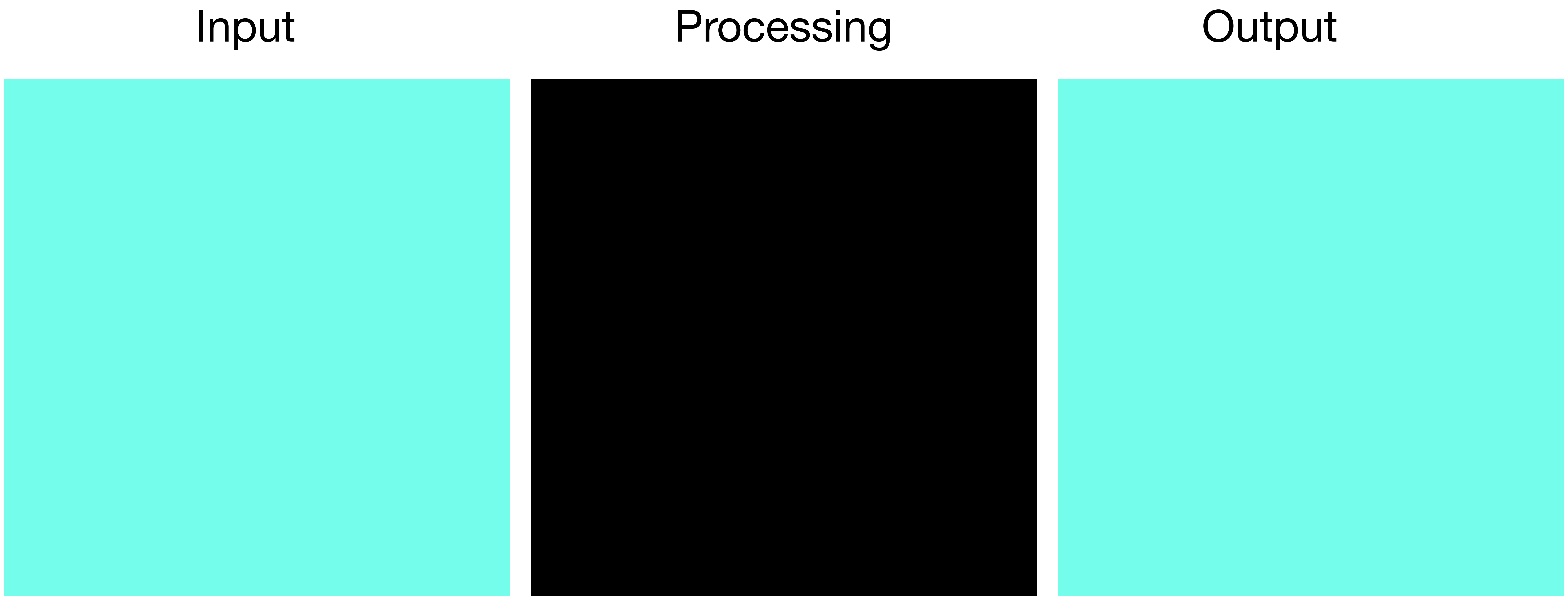
Consumers

Those who “drive” our application

- Could be human beings (User Interface)
- Could be other services (we are the provider) (Application Programming Interface)
- Could be:
    - Clock (CRON job)
    - File System Changes
    - Sensor Data
    - Messaging

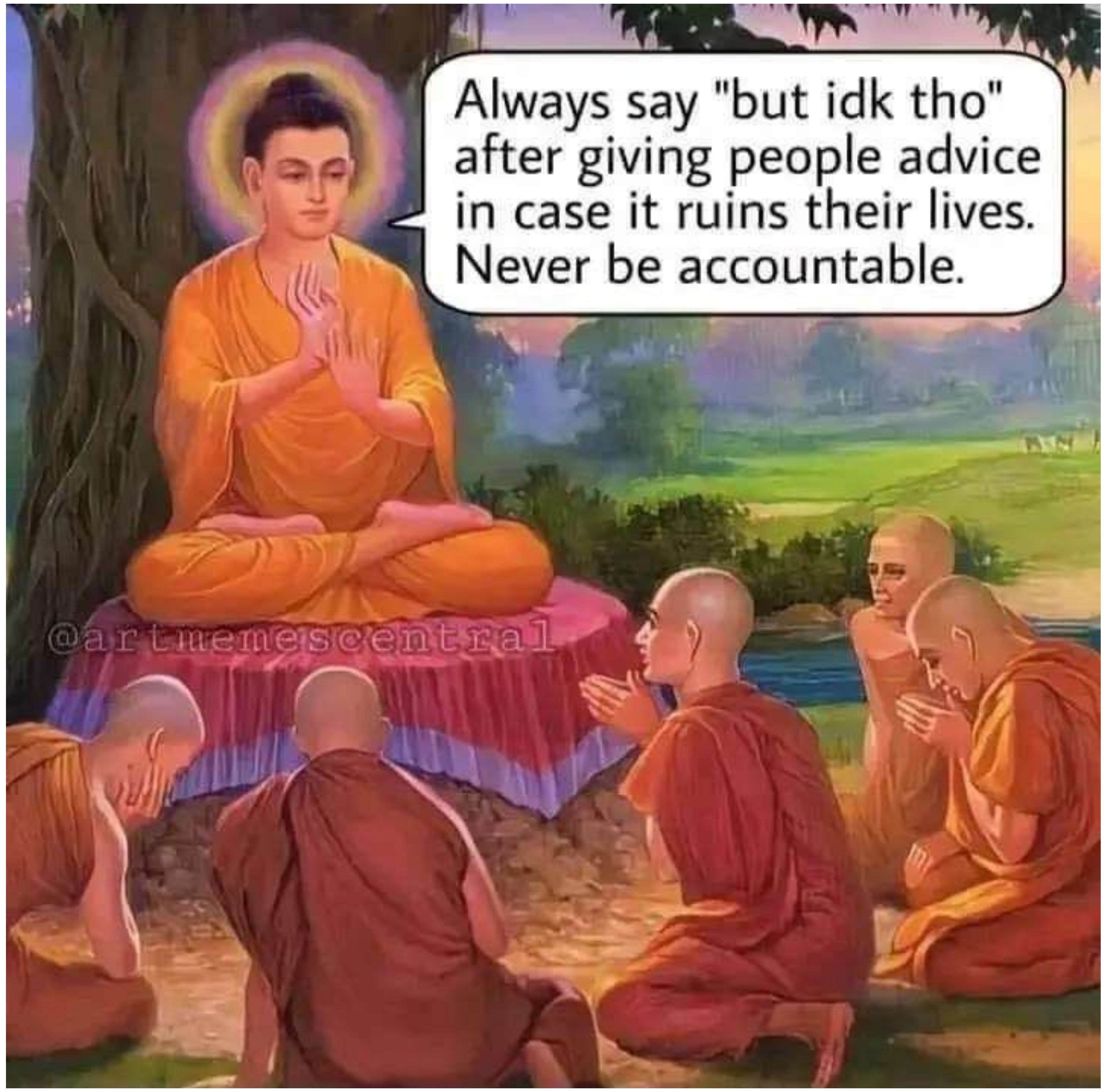
IPO Charts

Input - Processing - Output



Disclaimer

Relax. This is a conference.


Always say "but idk tho" after giving people advice in case it ruins their lives. Never be accountable.
@artmemescentral

What We’ll Cover

- Browser Apps
  - API Calls
  - Identity
- Server Apps
  - API Calls
  - Database
  - Messaging

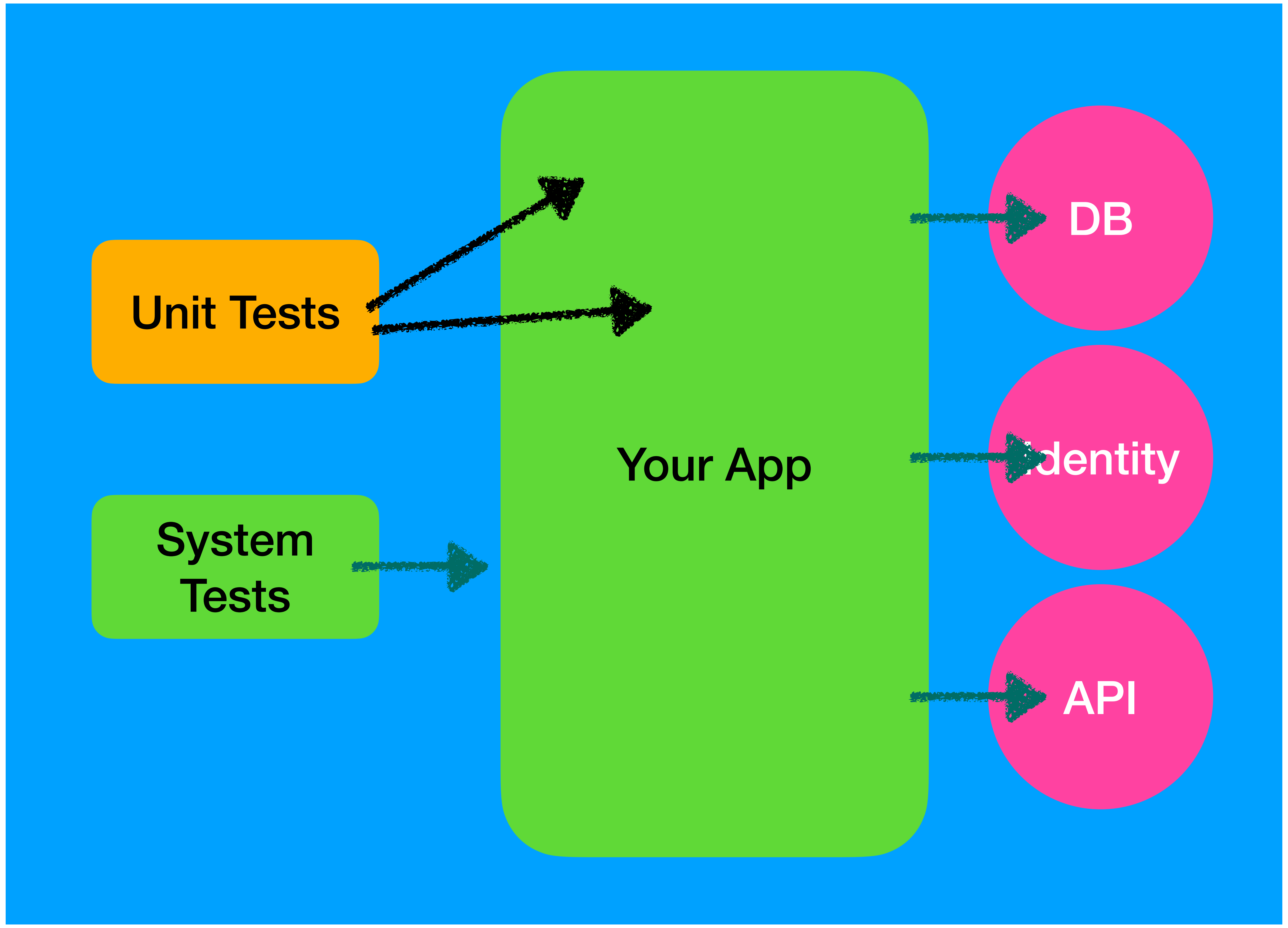
Addendum

Relationship to Automated Testing

- Using the techniques here you create *isolated* system tests.
    - Stated more strongly, you *cannot* create isolated system tests without the techniques shown here, or some version of this.
- By changing the configuration (and removing our out-of-process test doubles) you could create *connected* “end to end” or “system integration tests”
- You could (and should) add low level unit tests to increase you and your teams confidence in the portions that you feel aren’t adequately exercised by your other tests.

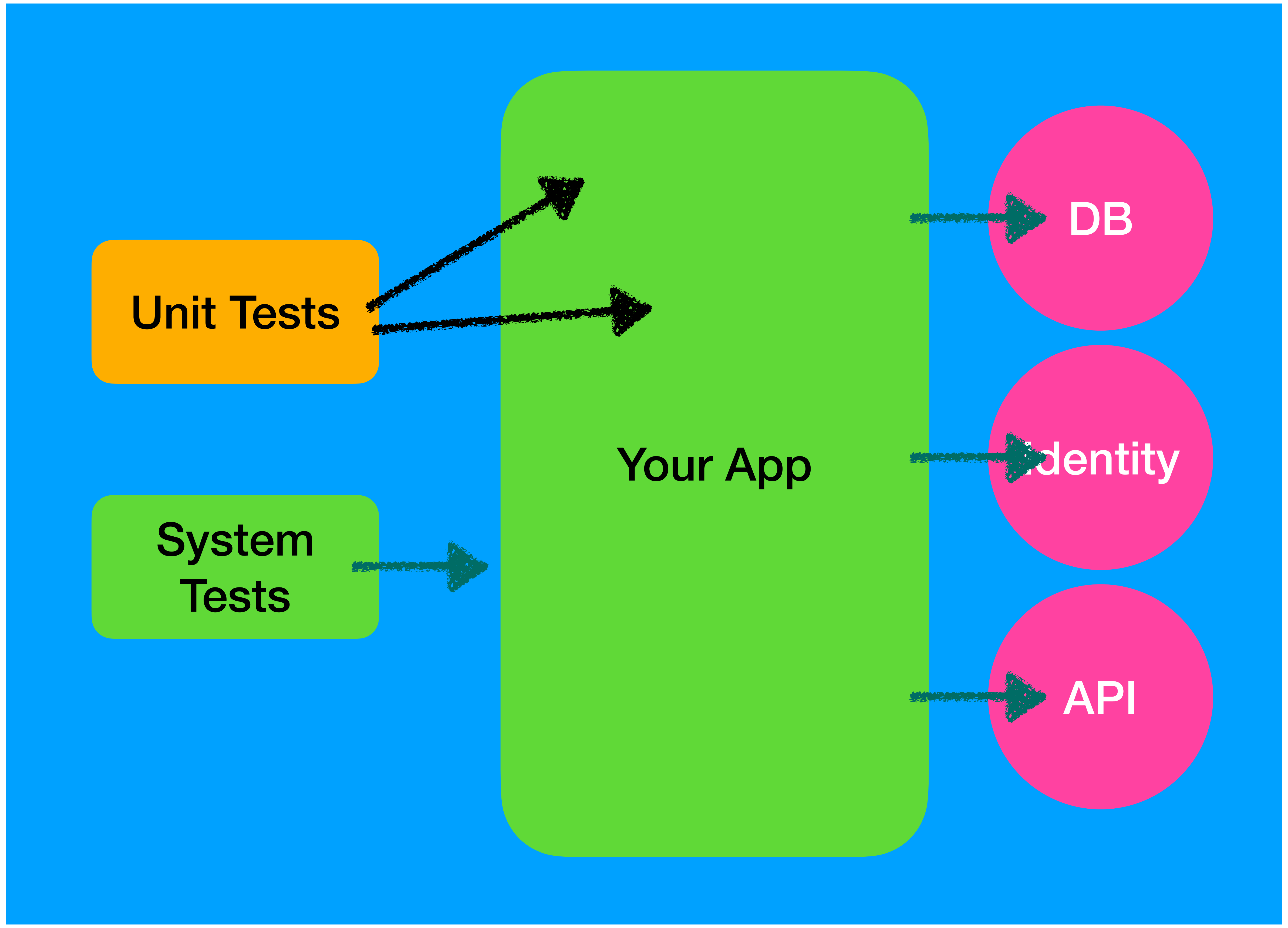
Addendum - Testing

Local Environment



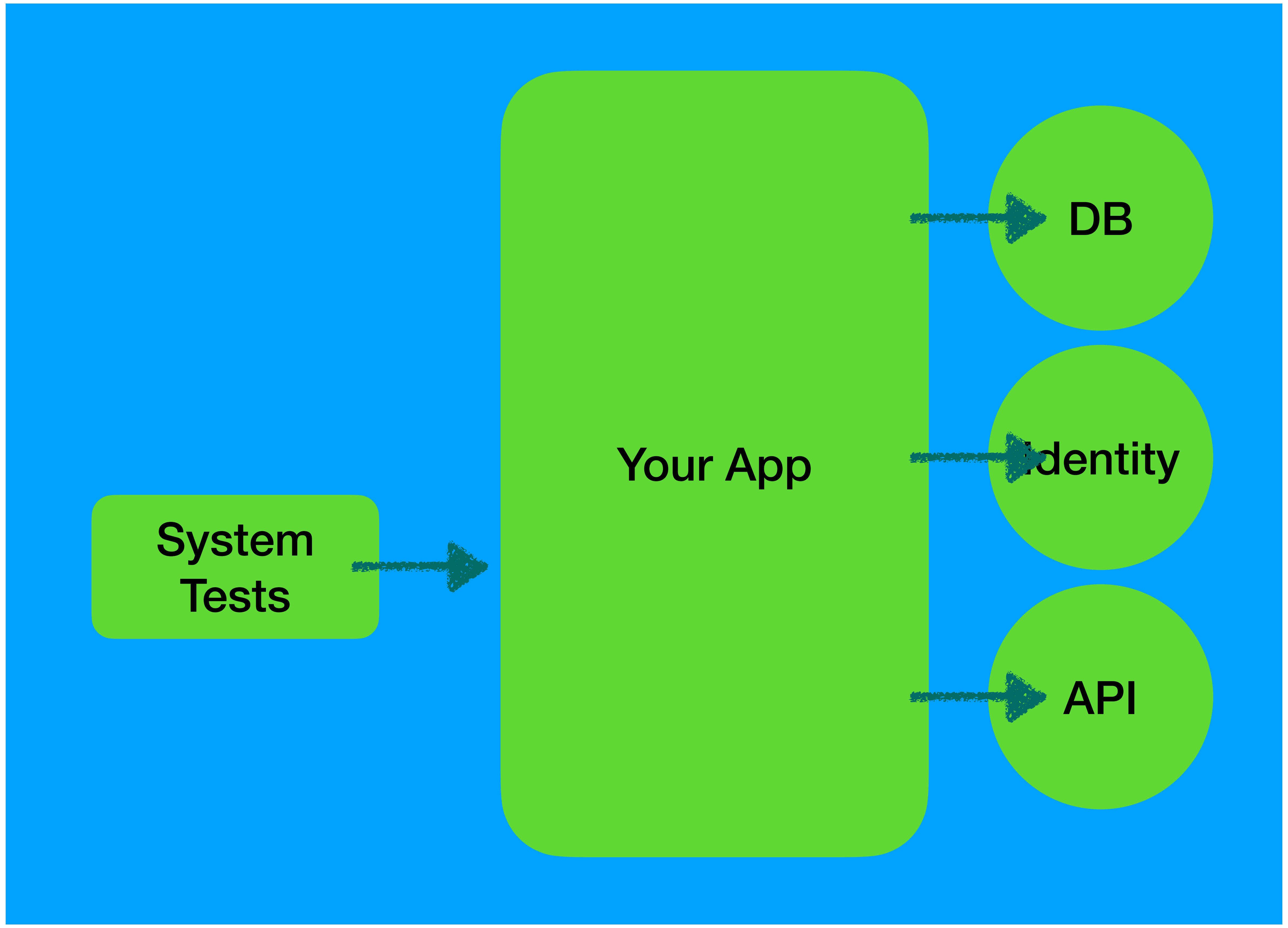
Addendum - Testing

CI Pipeline Testing



Addendum - Testing

Systems Testing (Staging, Production, Whatevs) “E2E”



Addendum to the Addendum

Consumer-Driven Contract Testing

- For backing services your company controls (especially APIs), you can use your tests to generate a specification for the team that owns that API be “bound” to.
- This is “Consumer-Driven Contract Testing”
- There are several tools - I recommend looking at https://pact.io/

I Hate Slides

- https://test-doubles.hypertheory.com