

HOLISTIC TESTING STRATEGIES FOR MOBILE APPLICATIONS

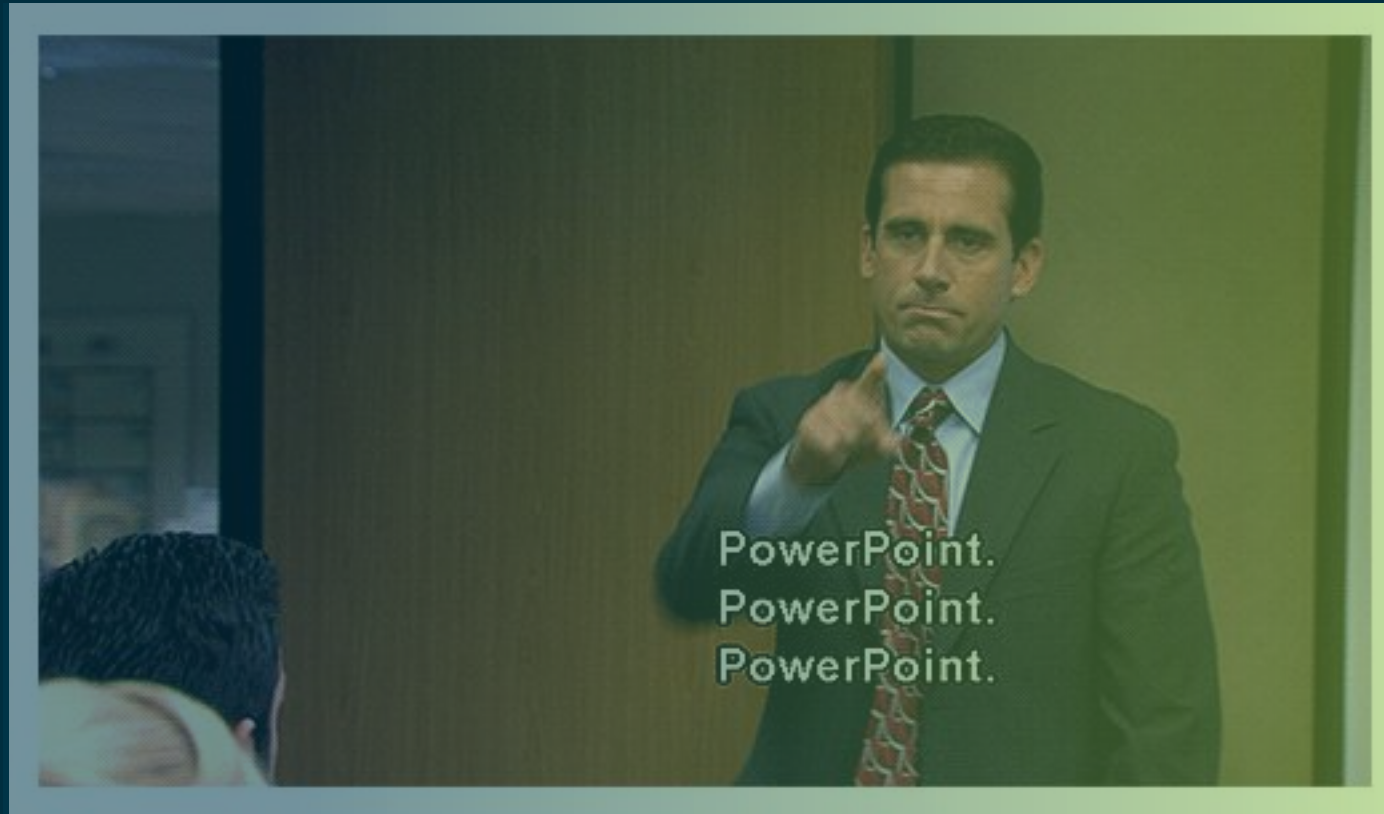
How to ensure a consistent high level of quality

Tobias Weidinger | tobias.weidinger@mhp.com

TOBS WHO?

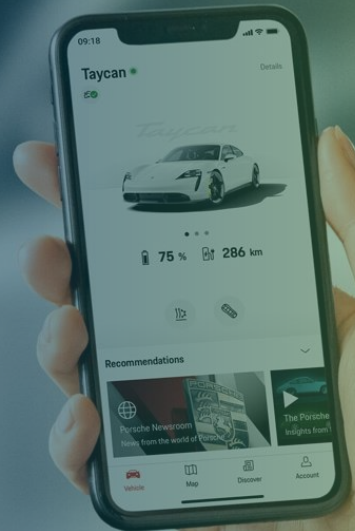
- Android Developer
 - 10+ years of Mobile App Development
 - Mostly Automotive
 - Passion for Good UX
 - (Some) Backend Expertise
- *Home Assistant*
- *Drum and Bass*
- *Sim Racing*





App People Around?





WHAT IS AN APP?

Characteristics of a mobile application

VISUAL APPEARANCE

- Layout
- Readable Texts
- Corporate Identity
- Accessibility

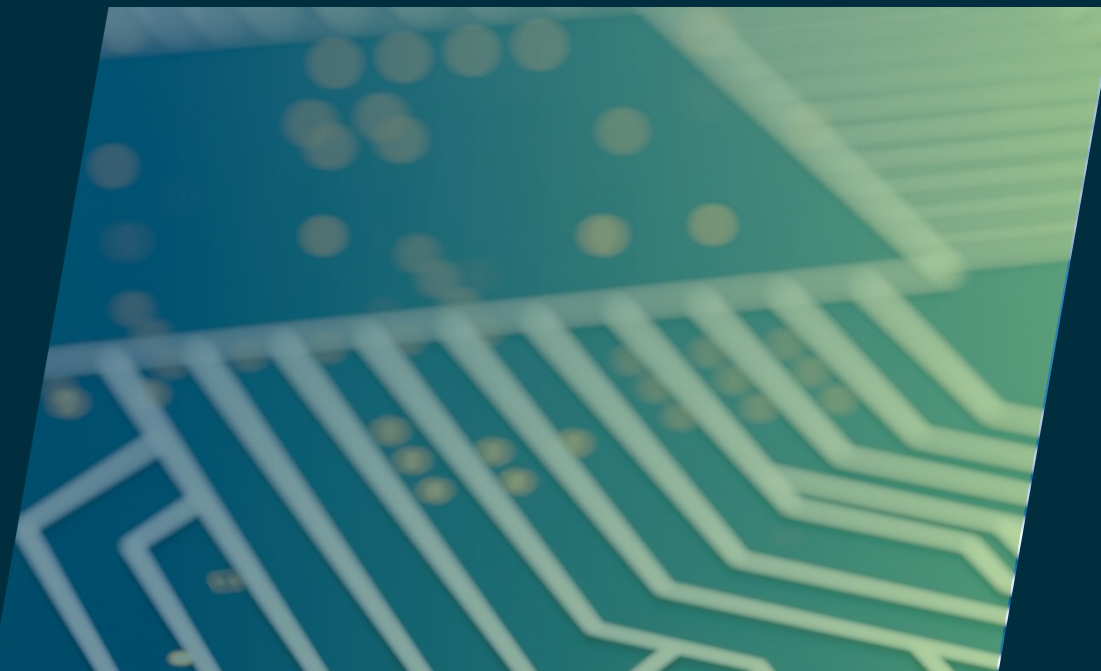


BEHAVIOUR

- Deterministic
- Correct Business Logic
- Even under weird conditions and edge cases

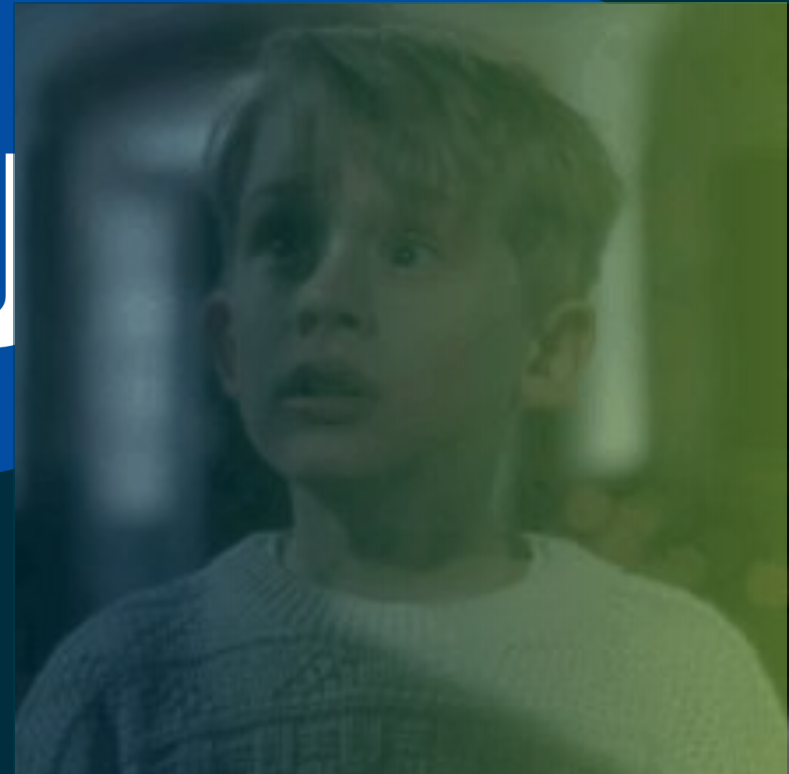
PLATFORM CHALLENGES

- Compatibility
 - Different API Levels
 - Screen Sizes
 - CPU Architecture (ARM, X86, RISC,...)

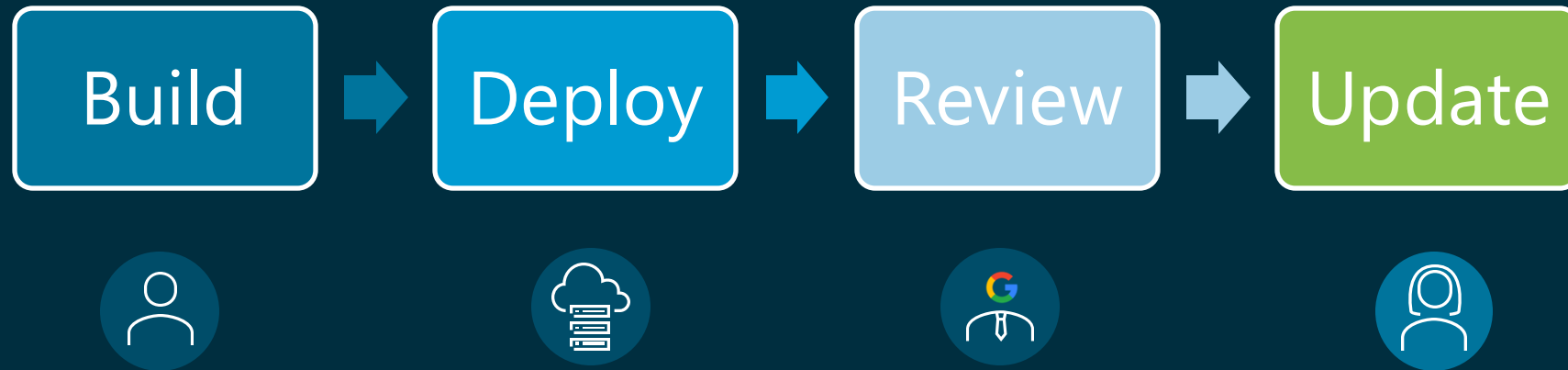


PLATFORM CHALLENGES

SAMSU



RELEASE CYCLE



MOTIVATION

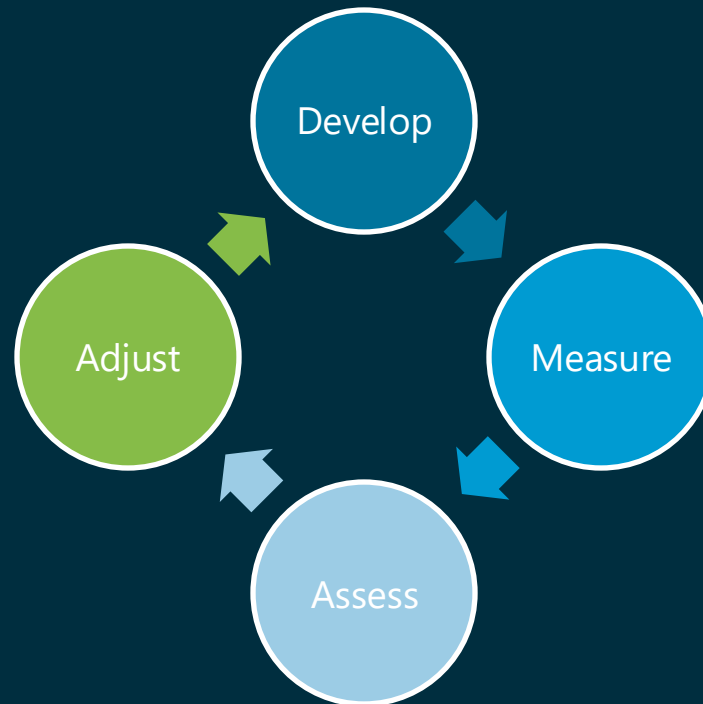
Why ~~should~~ do we care?

CONTINUOUS DEVELOPMENT

- MVP -> Product
- Add Features
- Improve Experience

RELEASE IN SHORT ITERATIONS

- Respond to feedback
- Launch new features



CUSTOMER SATISFACTION

- Growing user base
- Loyalty
- Revenue

CUSTOMER SATISFACTION

- Growing user base
 - Loyalty
 - Revenue
-
- Improving > Fixing

STABILITY

- Solid architecture
- Reducing time between first occurrence and fix of a bug

EXPECTATIONS

What's our target?

MANUAL TESTS?



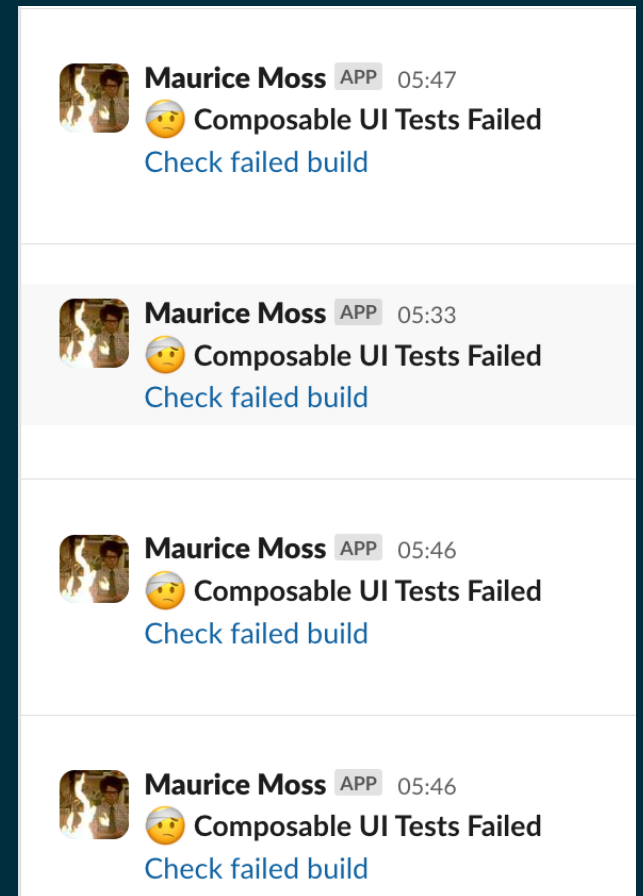
MANUAL TESTS

- Increasing Costs
- Slow Results
- Trustworthy Results?

AIM FOR AUTOMATION

- Fast Results
- Frequent Execution
- Joy to work with 🤖

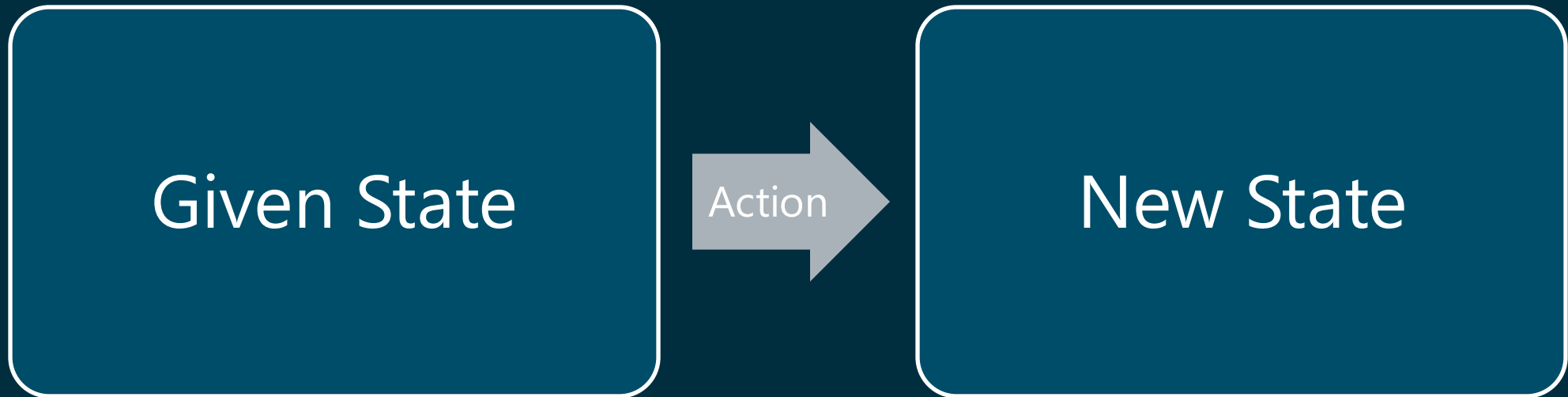
- Not Flaky
 - Don't invest time to find errors that don't exist
 - Keep attention on results
 - Reduce the noise
 - Don't ignore actual issues



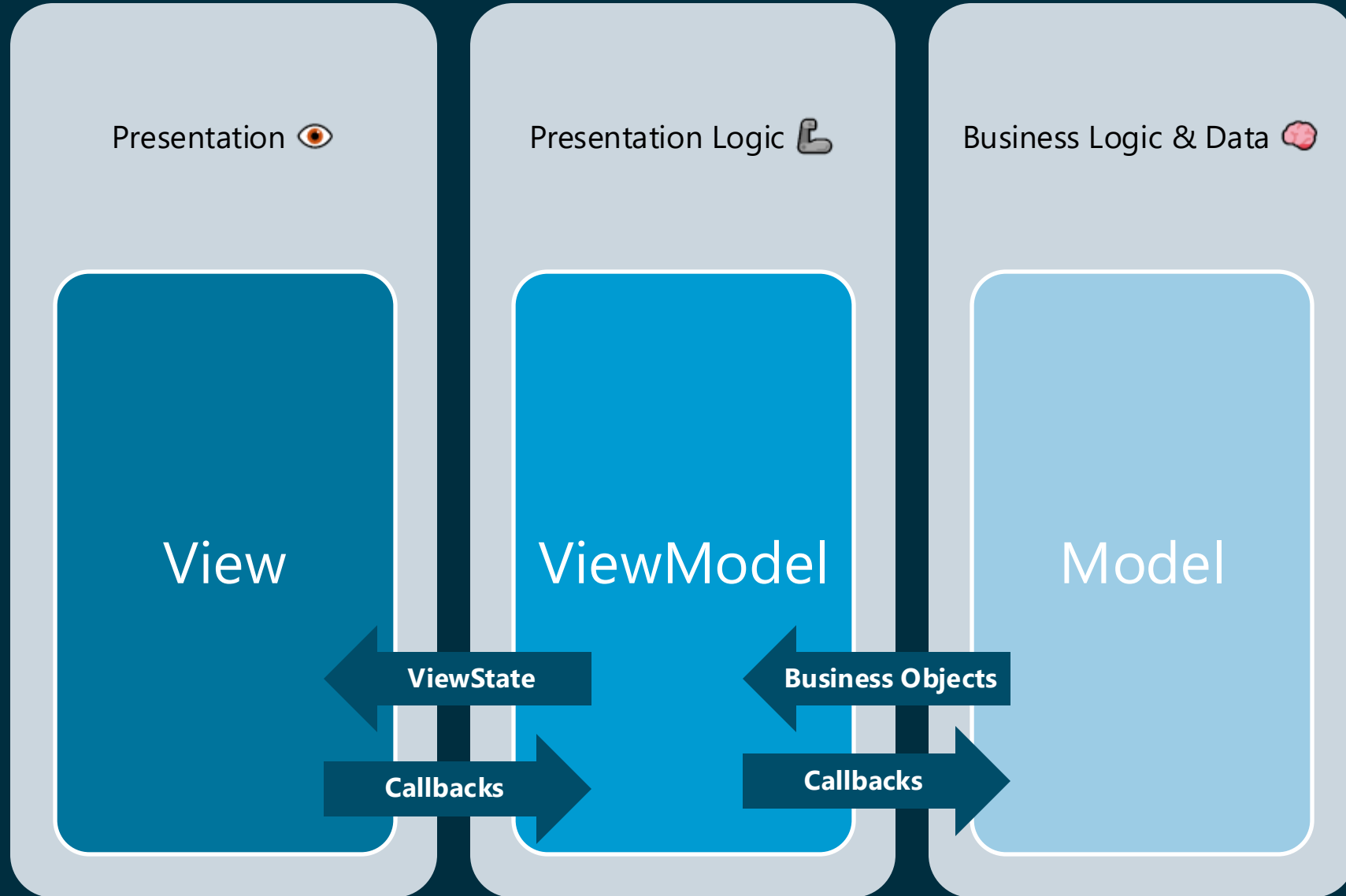
MOBILE ARCHITECTURE

How is it built?

STATE MACHINE



MVVM




MULTI MODULE

- Split per feature
- Extract reusable components
- Gradle modules



EXTERNAL DEPENDENCIES

- Multiple versions
- Compatibility?
 -  Version conflicts

TESTING STRATEGY

How do we solve this?

- Scope is the **entire** user experience
- End to end perspective
- All aspects
 - Layout
 - Business Logic
 - Compatibility
 - Availability

SUSTAINABLE

- Lean
- Avoid waste
- Write once - Benefit everyday

ONE FITS ALL?



ONE FITS ALL?

- Selenium/Appium/Browserstack
 - Flaky
 - Based on yet another language
 - Slow
 - Cumbersome

ONE FITS ALL?



- State Machine
 - **Given** defined state
 - **When** performing action
 - **Then** observe output / new state

UNIT TESTS

- Per class / per function
- Multiple combinations of input
- Super cheap
- Mocked dependencies
- High number of test cases

SCREENSHOT TESTS

- Comparing pixels
- Static layout
- Given state(s)

COMPONENT TESTS

- Multiple classes
- Higher level of abstraction

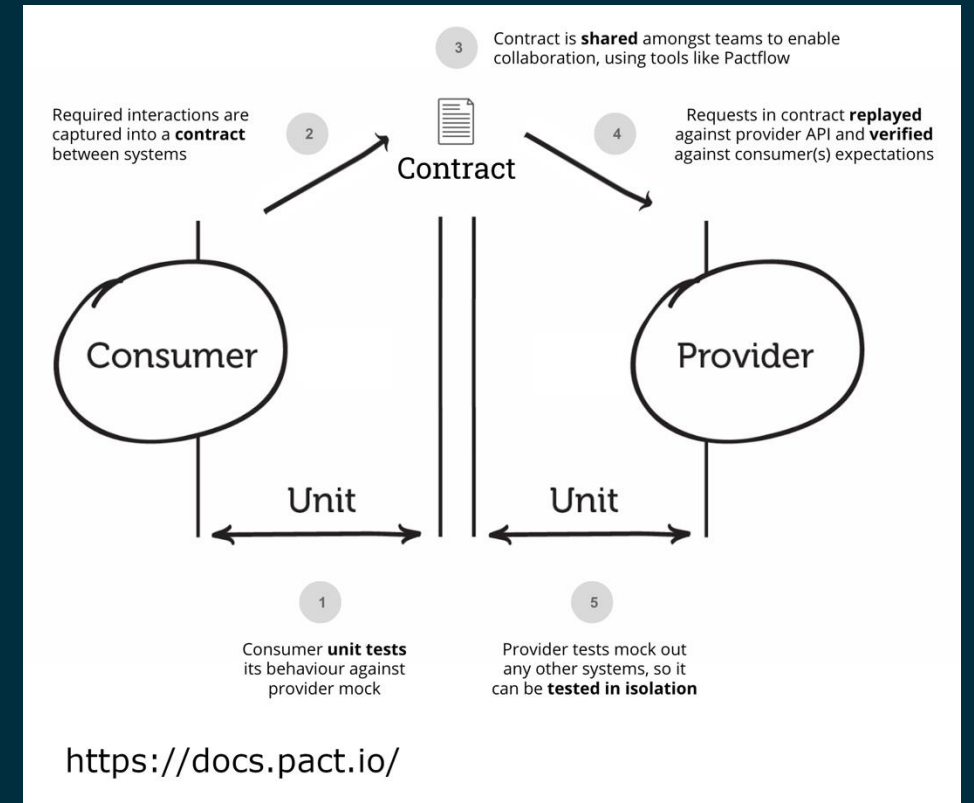
API CONTRACT TESTS

- Communication between Frontend and Backend
- Can we deploy?
 - Breaking changes?
 - Backwards compatible?

API CONTRACT TESTS

- Communication between Frontend and Backend
- Can we deploy?
 - Breaking changes?
 - Backwards compatible?

■ Example: Pact Tests



INTEGRATION TESTS

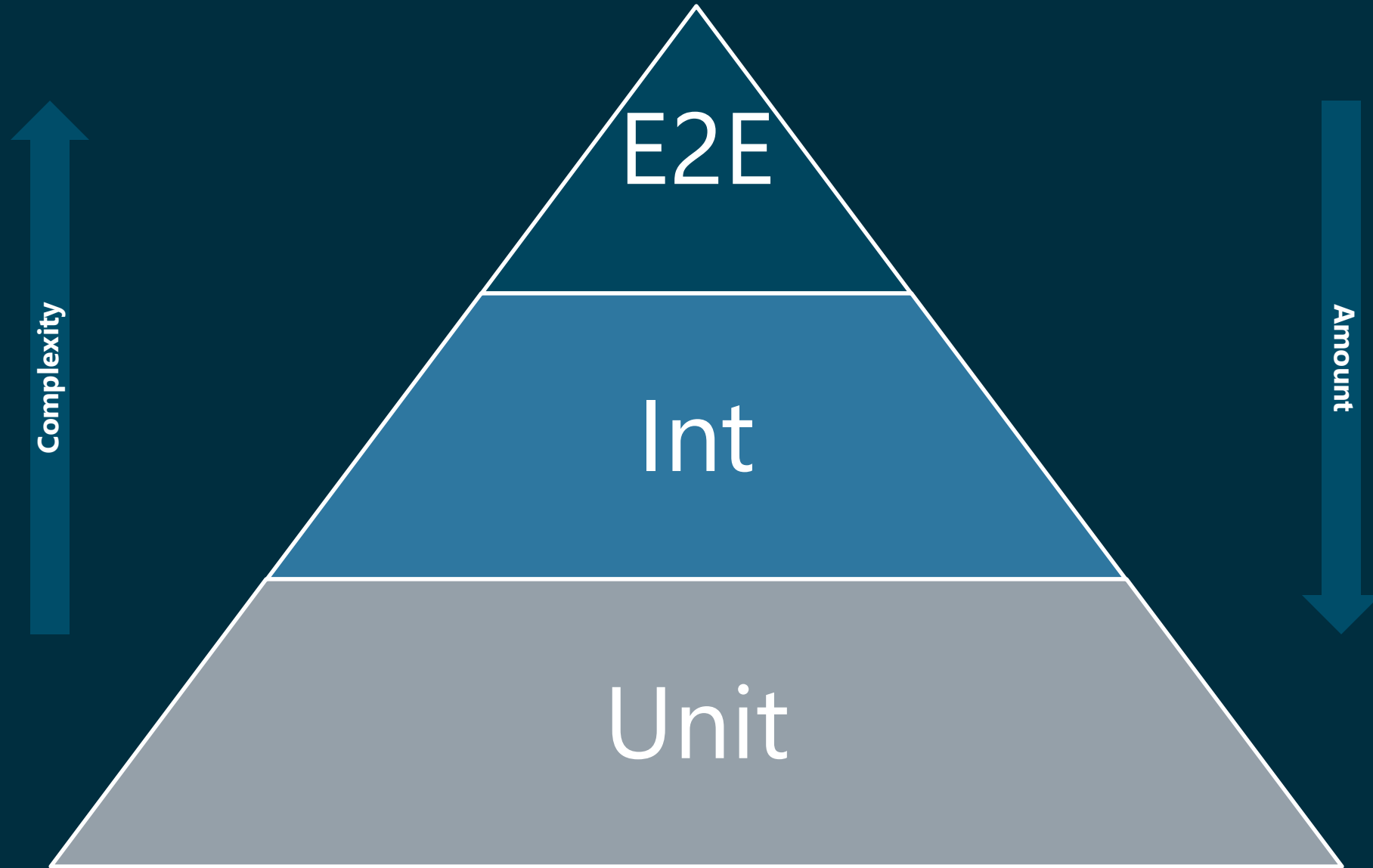
- Even more abstract
- Integrate higher level components into a more complex environment



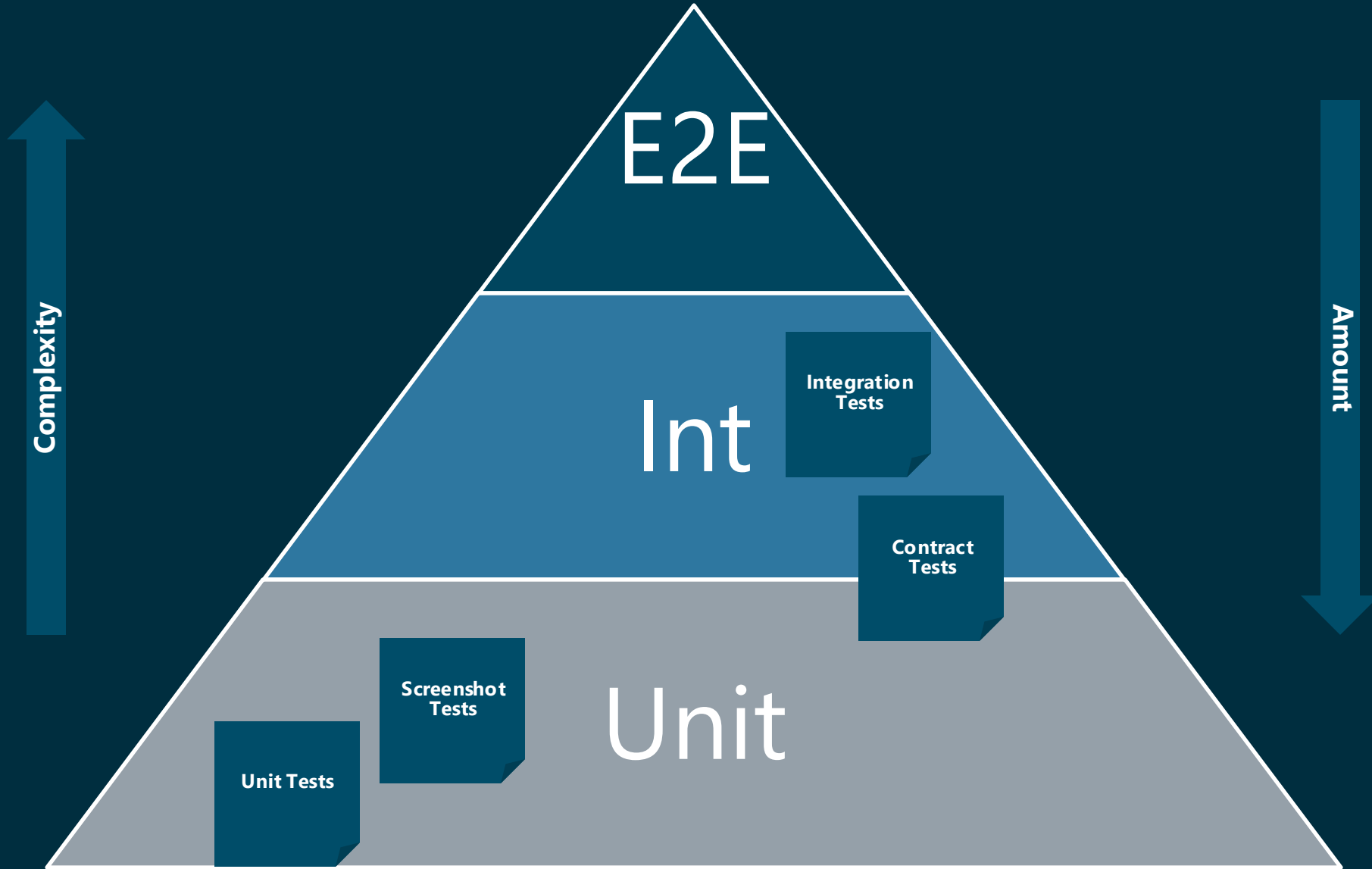
END TO END TESTS

- User perspective
- Black Box
- Hard to maintain
- Flaky by nature

THE TRUTH IS IN THE PYRAMID



THE TRUTH IS IN THE PYRAMID



HANDS ON

