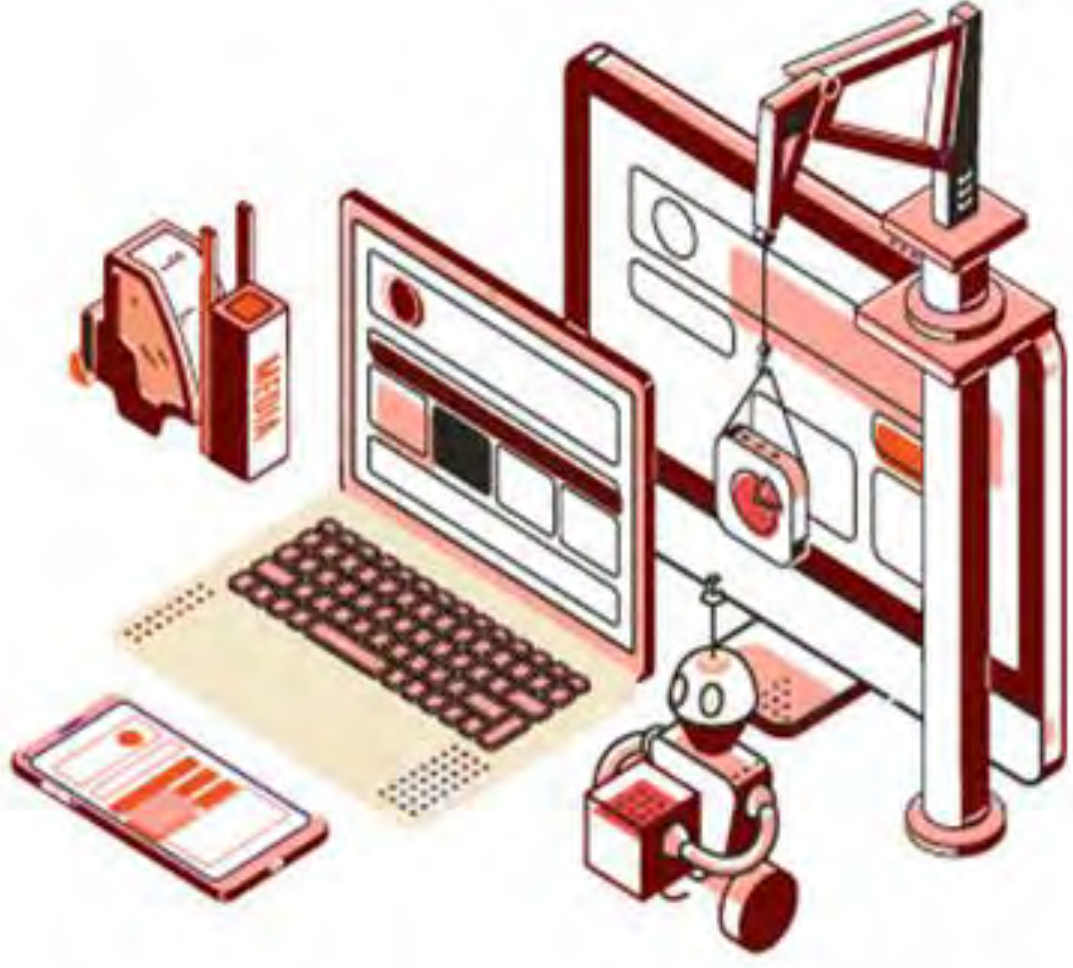
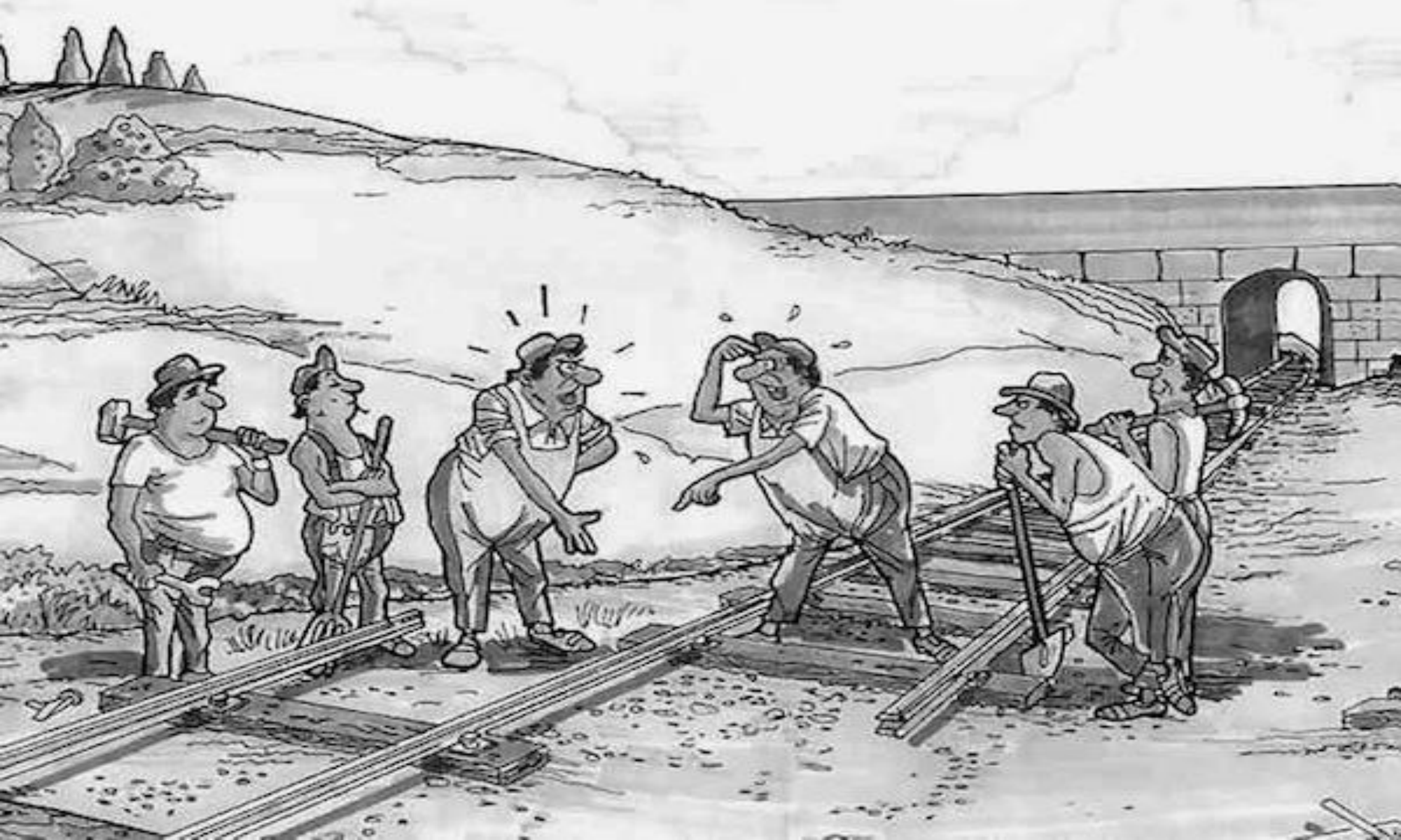
END TO END

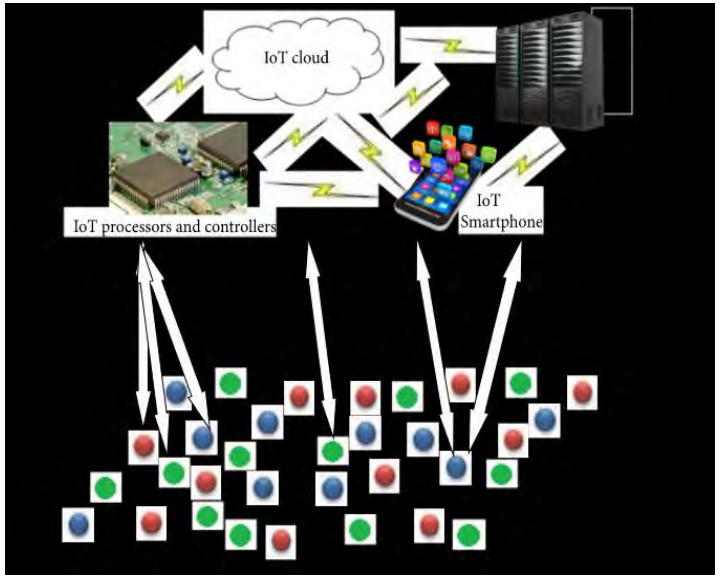


Testing Strategy for Embedded System

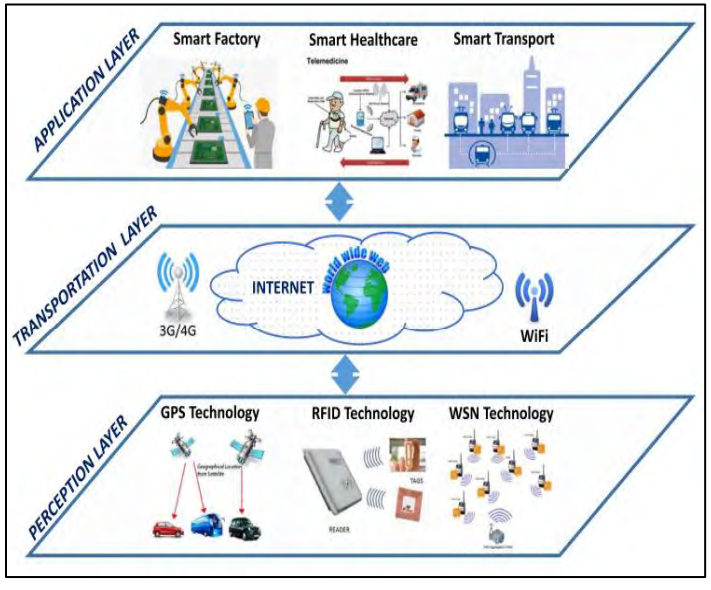
WHY DO WE NEED END TO END TESTING WHEN WE CAN TEST EACH ASPECT OF THE APPLICATION?



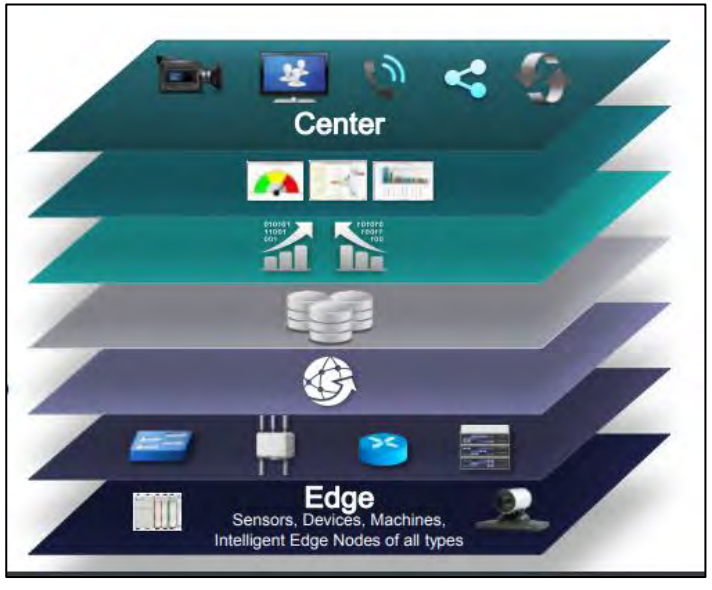
Iot layered Architecture



2 Layered Architecture



3 Layered Architecture



7 Layered Architecture

IoT World Forum Reference Model

Levels

7 **Collaboration & Processes**
(Involving People & Business Processes)

6 **Application**
(Reporting, Analytics, Control)

5 **Data Abstraction**
(Aggregation & Access)

4 **Data Accumulation**
(Storage)

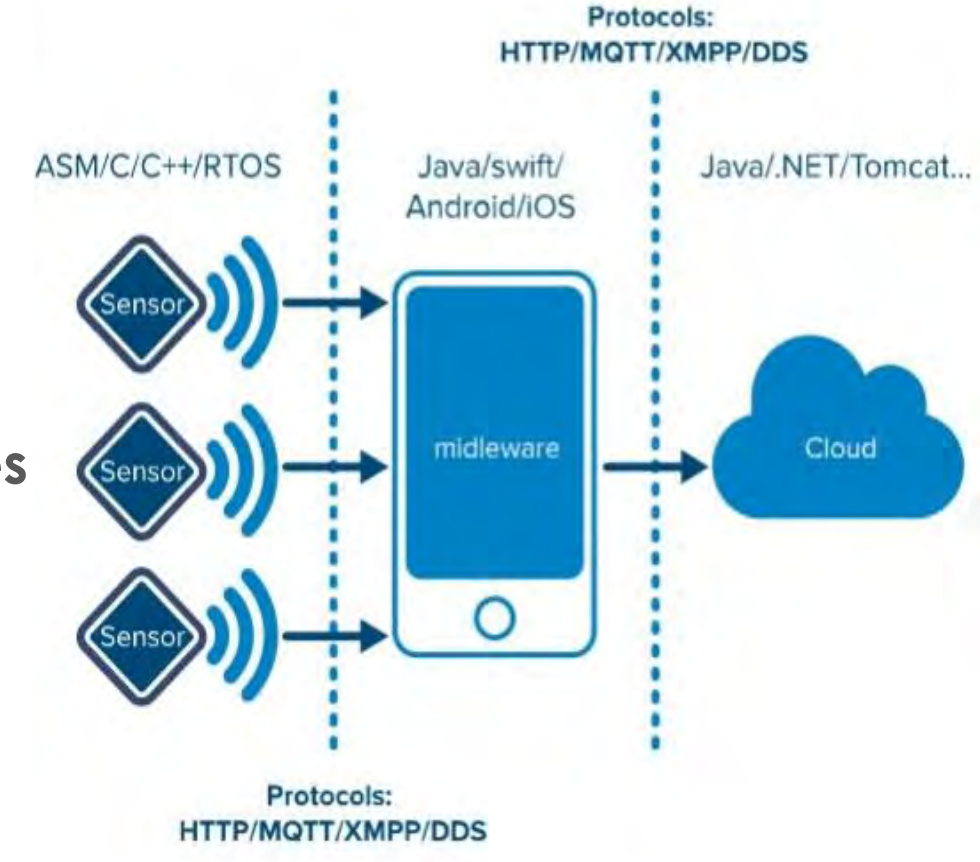
3 **Edge Computing**
(Data Element Analysis & Transformation)

2 **Connectivity**
(Communication & Processing Units)

1 **Physical Devices & Controllers**
(The "Things" in IoT)

Challenges in Iot Testing

- Multi-Layered Systems
- Dissimilar Technologies – Low level microcontrollers & high level server programming
- Functionalities spanning across multi layers
- Incompatible protocols between devices
- Functionalities developed by different teams
- Solution looks simpler due to limited interface
- Not enough code to justify testing



Testing Effectiveness

General Practice

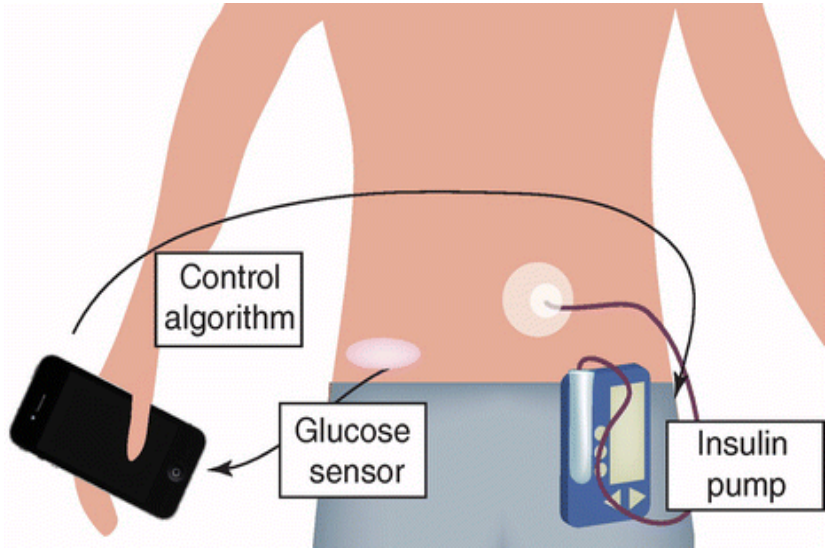
- Organizations use E to E testing at System Level
- Follow the most logical option available i.e. assemble the system fully and then test
- This gives a realistic simulation of end user experience

Issues they face:

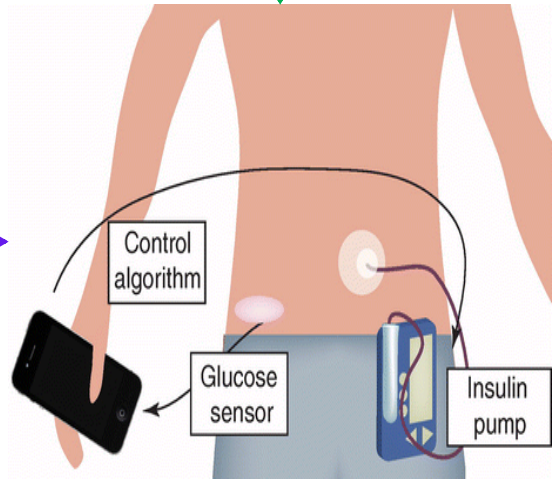
- System cannot simulate all situations
- Building such a system is very time consuming and costly
- Testing is challenging as system involves many applications working in tandem
- Late Cycle error detection
- Delay in Time to Market
- Verifying responses from all applications is difficult

Blood Glucose Tracking System

- Wearable
  - Glucose sensor
  - Insulin injector
- Smartphone - Middleware
- healthcare system in Cloud



Blood Glucose Tracking System Diagram



Data for Interface to monitor important Parameters

Measurements sent Wirelessly

Middleware Mobile Phone

Measurements Stored Locally

Warning to user

Immediate Medical help

Glucose levels and other metrics

Healthcare system

Alert to Medical staff

Comparison with historical data and analysis, find unwanted patterns

Wider Scenarios

Blood Sensor is Simulated

Data Package is selected

Data moves to cloud

Cloud generates Alerts

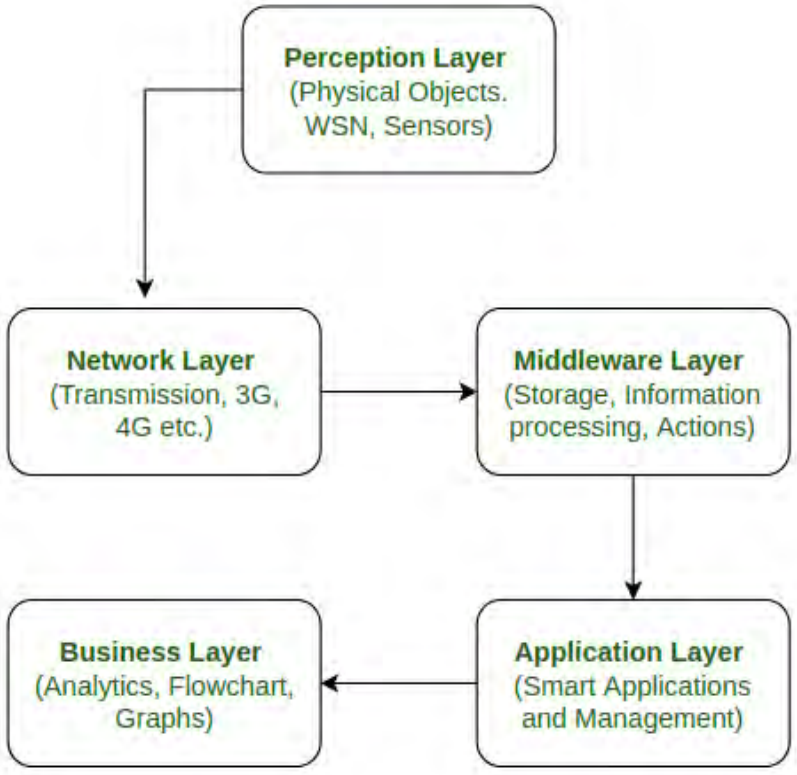
Medical staff responds to Alerts

Patient receives Alert / notification

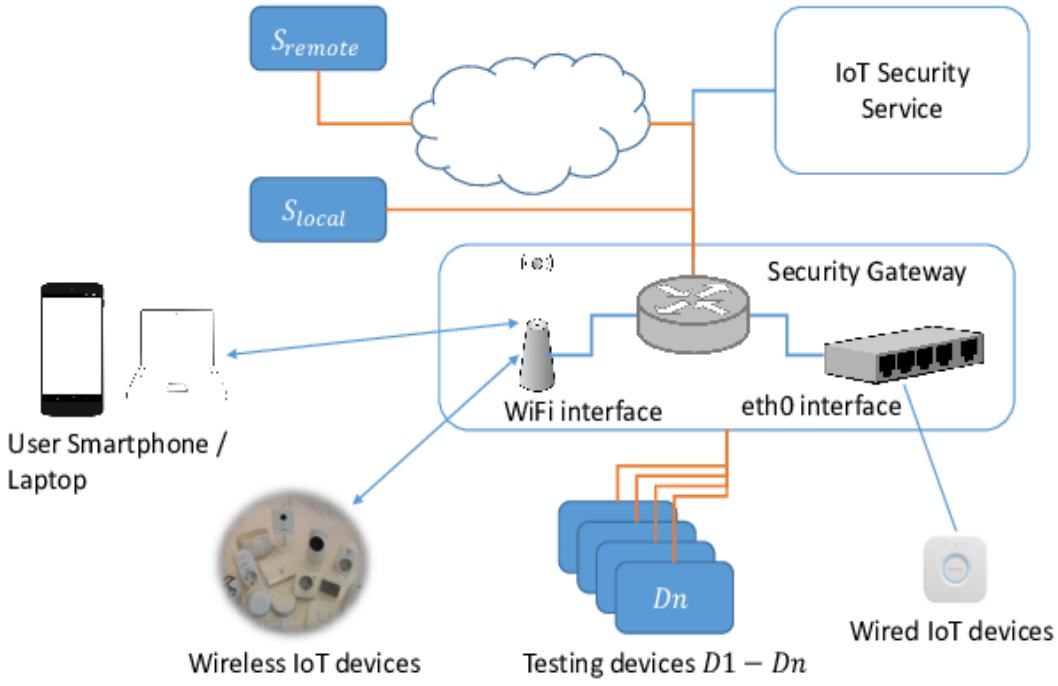
Managing Injection schedule and injecting insulin

Solution?

- Deconstruct the System into Layers for More Effective Testing



- Isolate Components and test early



Deconstructing The System Into Layers

Primary challenges

- Designing the system in a way that it can be conducive to deconstructing in smaller blocks with well defined interfaces
- Build Automation around these blocks

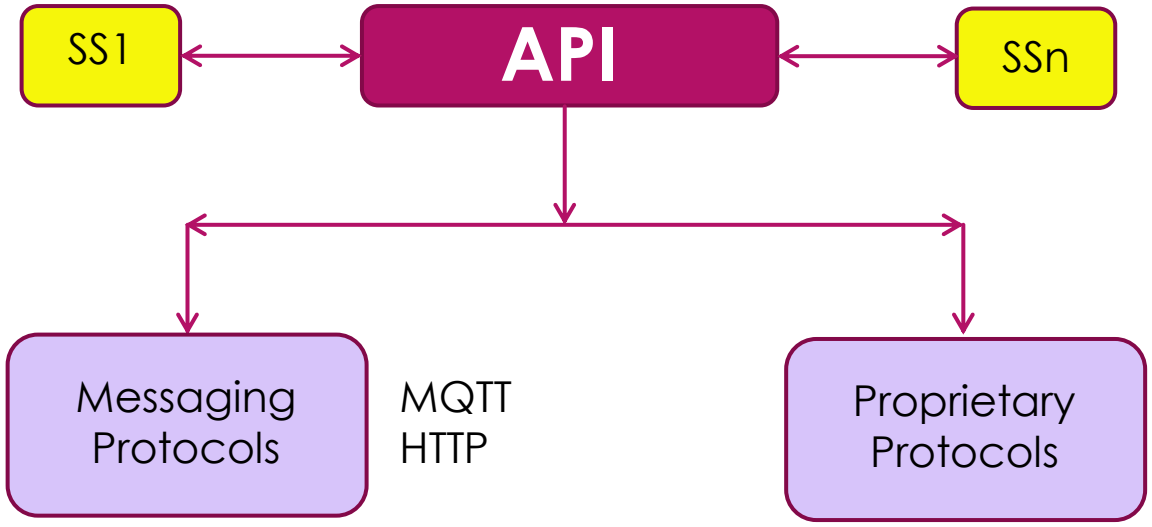
Unit tests vs functional tests?

- Unit test when system is complex
- Unit tests help finding root cause quicker
- Functional tests less prone to breakage, but hard to find system bugs

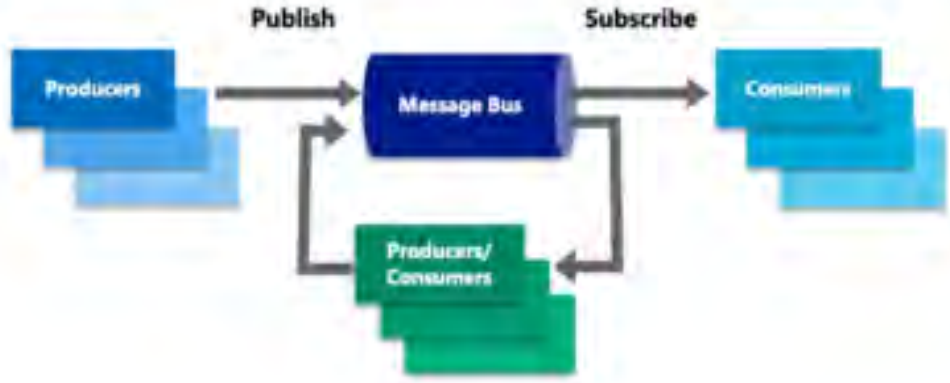
Recommendation: Use a blended approach for testing IoT

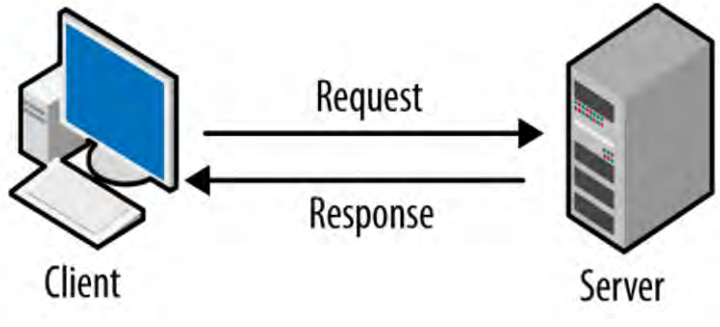
Deconstructing The System Into Layers

Wireless Communication Layer



Communication Model





Automation Is The Key

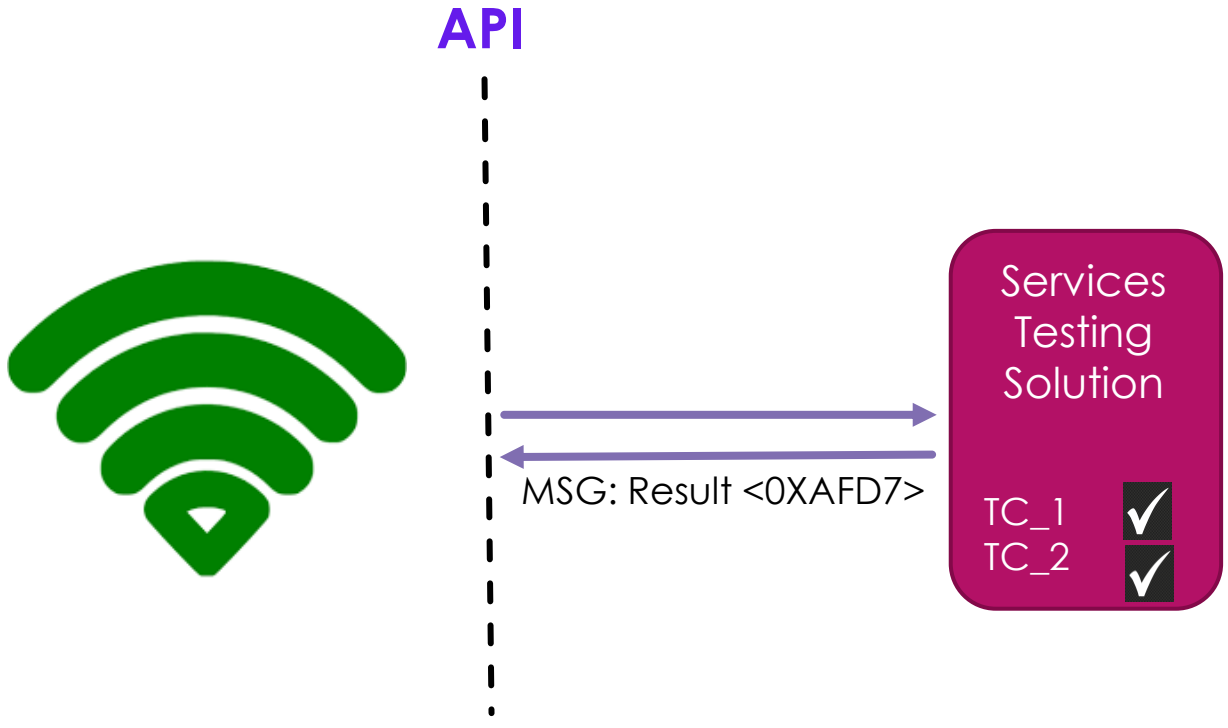
Components

- Blood Sensors
- Injectors
- Cloud app
- Mobile

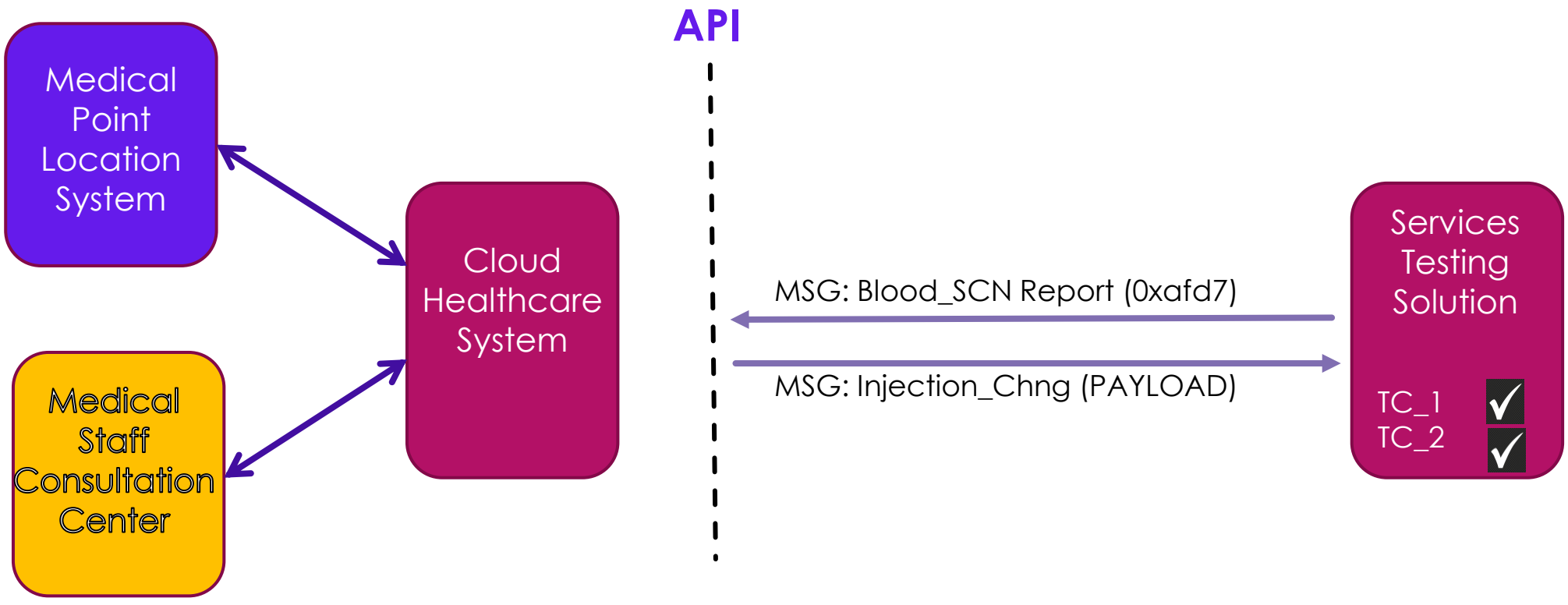
All are Wireless and interact through API

Automate the flow !

And its SIMPLE !!



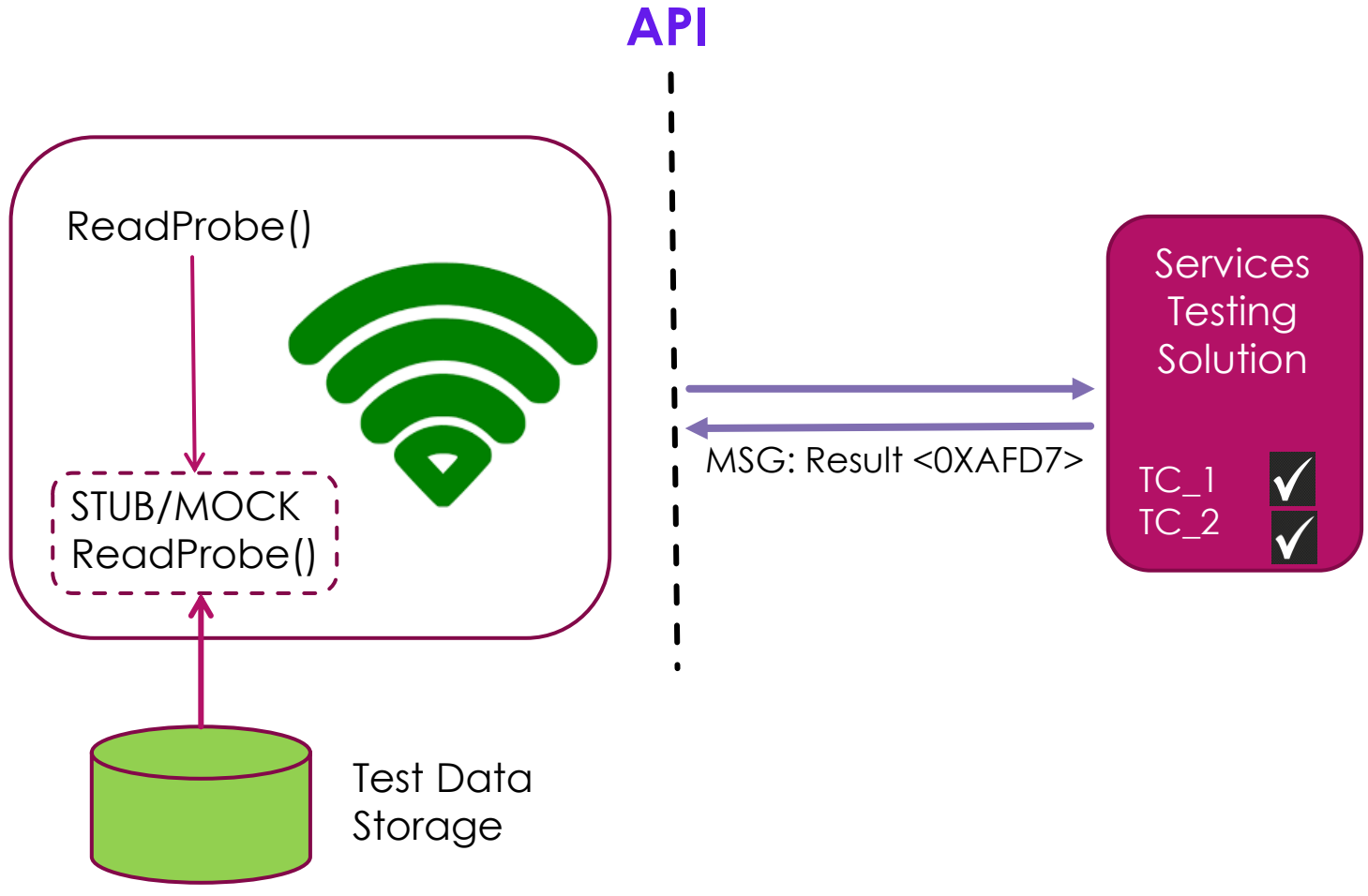
Automating Server Components



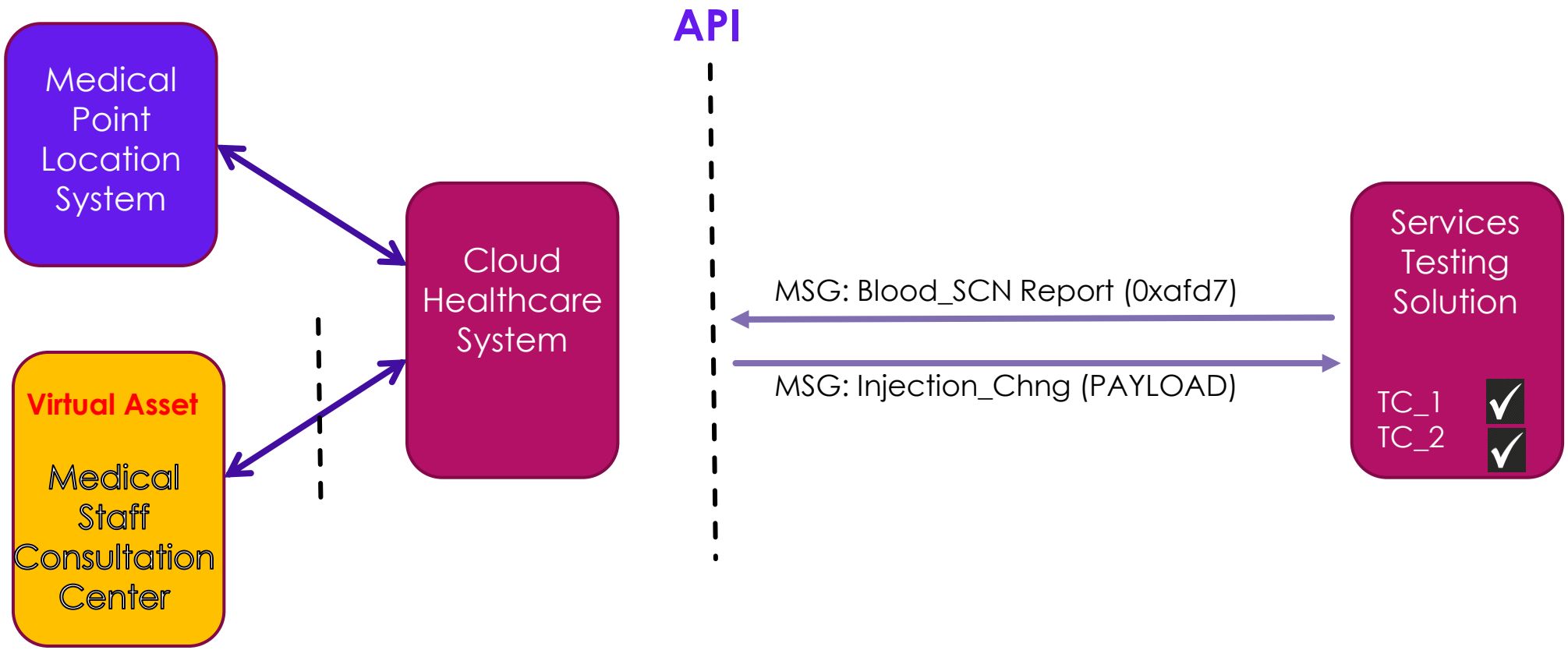
Isolate Components And Test Early

Case: Scan blood to determine Glucose level

- Intercept ReadProbe() function call
- Redirect it to a Virtual Asset
- Generate request and Simulate Response



Isolate Server Side Components And Test Early



Things To Remember

- IoT systems require thinking about software quality in a larger scope.
- IoT solutions, such as our medical device example, are different from "normal" systems because an individual feature or function may span multiple layers of the solution.
- Delivering a high-quality system requires testing capabilities at every layer: the low-level layer in C code, the API testing layer, and the hard-to-access back-end part of the solution.
- Consider the cost associated with the system because a design failure far outweighs the cost of deploying a testing solution that enables you to isolate and test components or API testing or backend testing

About Me

: vipinqalead

: @vipin_QA

: vipin.jain@metacube.com


QUESTIONS ?