Navigating the MV* Mess

$ whoami

Agenda

- System design and quality attributes
- Architecture tradeoffs analysis method
- How to perform intermediate design reviews
- Android architecture patterns
- Considerations when picking an architecture pattern
- How to navigate architecture decisions in the enterprise

Disclaimer

The opinions expressed in this presentation and on the following slides are solely those of the presenter and not the ones of any of the past and current employers of the presenter.

Every effort has been made to offer current and accurate information to our users, but errors can occur. The content has been curated to the best ability of the presenter to be inclusive and gender-neutral.

The slides deck is intentionally minimal for readability, presentability, and accessibility. For an annotated version of the slides reach out to the presenter on Twitter @giorgionatili.

Understanding Quality Attributes

Quality Attributes

A quality attribute of a system is the definition of a *characteristic* that the system must have.

A Starting Point

The most common quality attributes are *performance*, *reliability*, *availability*, *security*, *modifiability*, and *subsetability*.

Which Quality Attributes Matters for You?

Performance

The ability of a system to allocate *computational resources* to satisfy *timing* requirements.

Reliability

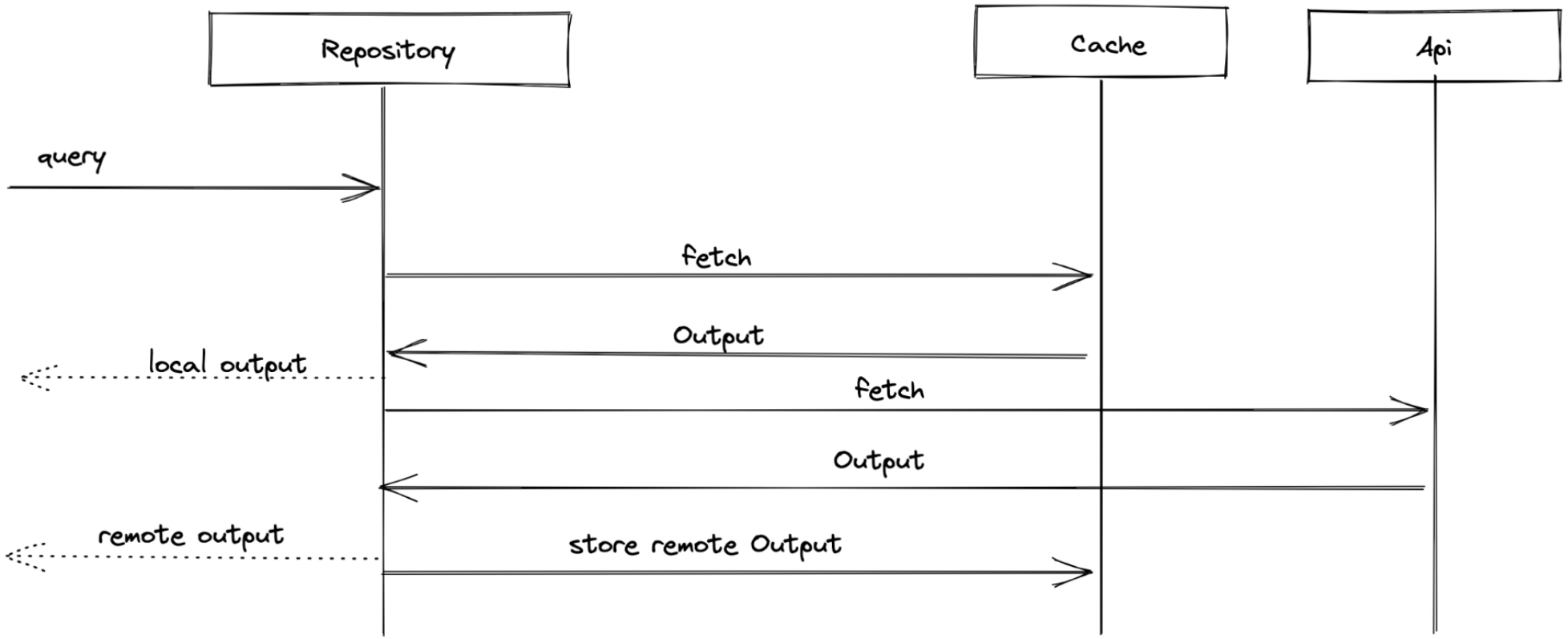
The ability of a system to keep operating overtime.

Availability

The portion of time a system is up and running and its fault recovery capability.

What About a Mobile App?

Local Cache Activity Diagram



Security

The measure of the system to resist *unauthorized attempts* of usage and denial of service.

Modifiability

It is the ability to *make changes* to a system quickly and cost-effectively.

Subsetability

The ability to support the deployment to the production environment of a *subset of the system*.

Attribute-Based Architecture

Quality attributes deeply influence architecture *decisions* and *patterns*.

Ambiguous Attributes

Quality attributes are often ambiguous and not clear,
that's why you want to invest time in clarifying them.

Architecture Tradeoff Analysis Method

Definition

The architecture tradeoff analysis method (ATAM) is a risk-mitigation process used in software development.

Architecture and Quality Attributes

ATAM assesses the *consequences* of the architectural decisions in light of quality attributes and business goals.

Participants

ATAM brings together architects, managers, stakeholders, and the evaluation team for a *structured conversation*.

ATAM Lifecycle

A cycle of ATAM includes time for *presentation*, *analysis*, *testing*, and *reporting*.

Presenting Business Drivers

Stakeholders present the *functions* of the system, the expected *outcomes*, and the *success* criteria.

PR FAQ Documents

Frequently Asked Questions:

FAQ 1: Don't we already match leading mass merchants and club stores? What is new about Prime Pantry?
Today we only match the online prices of image competitors. With Prime Pantry, we will offer selection that is not available online, and match or beat competitor in-store prices.

FAQ 2: Is the unique Prime Pantry selection entirely new, or will items previously available to all customers in Amazon.com now be available only for Prime members?
Most of the Prime Pantry selection will be new to Amazon. We may continue to sell larger packs of a given product in Amazon.com/Consumables, but only offer the single pack in the Prime Pantry. We may convert some unprofitable Add-On items to Prime Pantry, especially if we can offer them for a lower price. Finally, we may move the Retail items in some of our most profit challenged sub-categories, such as liquid laundry detergent, entirely into Prime Pantry.

FAQ 3: Doesn't Amazon's Add-on program solve the low ASP problem?
No. The Add-on program does help by incrementally lowering the ASPs we can sustainably offer for sale on Amazon, but it does not allow us to go as low as we need to go to serve the highest consumer demand. At today's product margin, the breakeven ASP for Add-on items in most Grocery categories is $TK (versus an offline average of $TK). And this breakeven would be much higher on mainstream CPG selection price-matched to offline retail. Finally, Add-on will not allow us to profitably sell heavy items

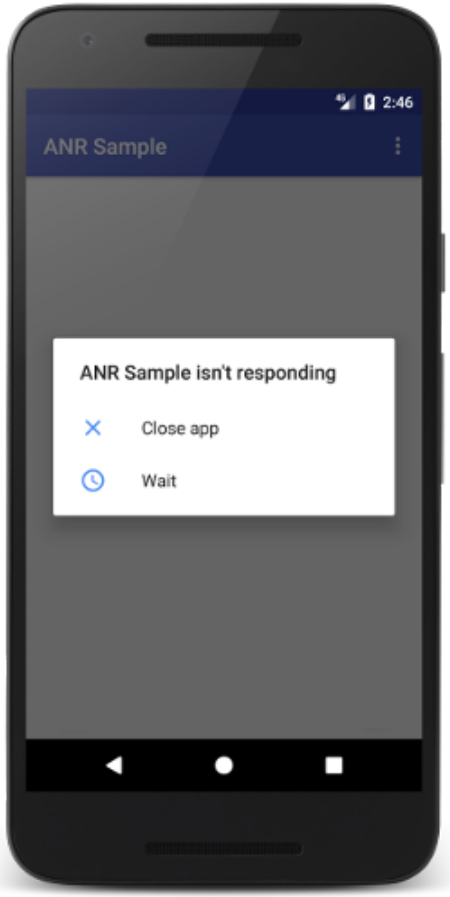
Presenting the Architecture

Architects present the candidate architecture through *functional*, *code*, *development*, *concurrency,* and *physical* views.

Analysis

It identifies architecture approaches, maps architecture decisions with quality attributes, and generates a *utility tree*.

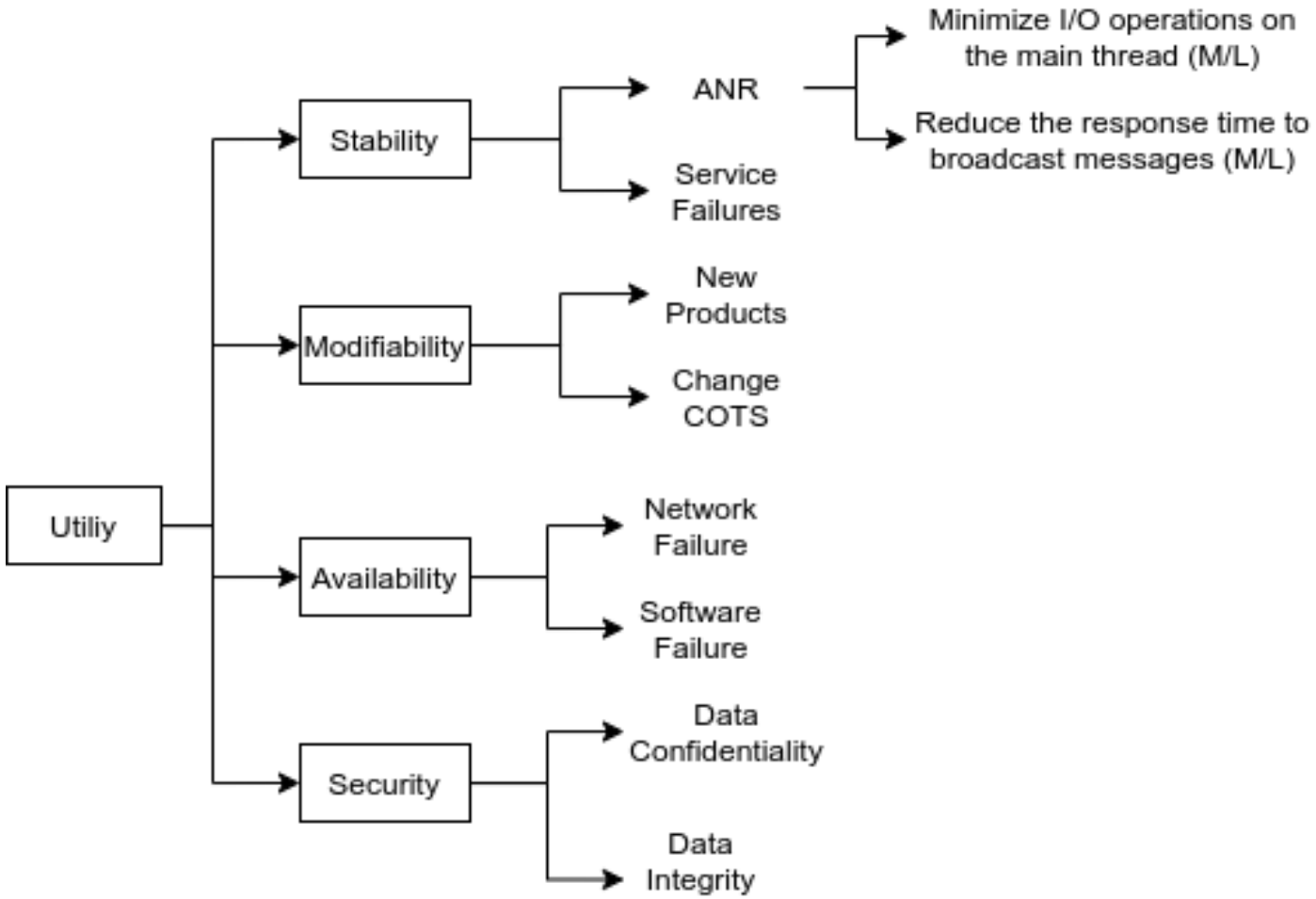
How To Reduce the ANR Rate?



Utility Tree

Starting from the most important quality attributes
generates a *prioritized list* of quality attributes requirements.

ANR Rate Attributes



Be Open Minded

The outcome of the utility tree exercise can be *surprising* and bring to de-prioritize quality attributes.

Assessing the Architecture

The process of assessing the architecture consists in *asking questions* based on the utility tree scenarios.

Testing the Architecture

The validation of the architecture consists of running it *through the scenarios* identified by the utility tree.

Use Cases Scenarios

Are scenarios that describe a user's intended interaction with the system (i.e., app + services).

Growth Scenarios

Are scenarios that validate how the system can handle more users, increased throughput, reduced latency, etc.

Exploratory Scenarios

Are scenarios that expose the *limitations* of the system architecture (extreme growth or requirement changes).

Risks and Non-Risks

Running an architecture through multiple scenarios identifies the *weaknesses* and *strengths* of a system.

Additional Outcomes

Prioritized scenarios, architectural decisions and changes, sensitivity points, and tradeoffs.

Sensitivity Points

Architectural decisions that involve one or more components that are critical to achieving a quality attribute.

Tradeoffs

An architectural decision that affects one or more quality attributes of the system.

Results

Early discussions within stakeholders, clarifications of the assumptions, decision records, and prioritized use cases.

Intermediate Architecture Reviews

Architecture Readiness

It is hard to determine if the architecture is *suitable* without building the system.

Architecture Lifecycle

The architecture design should be *iterative* and stop only when the system gets decommissioned.

Evaluating Partial Architectures

Identify stakeholders and reviewers, clarify assumptions,
and verify use cases.

Benefits

Reviewing pre-release designs can uncover important *inconsistencies* or *oversights*.

Architecture Patterns in Android

MV * (C, P, VM, I, U)

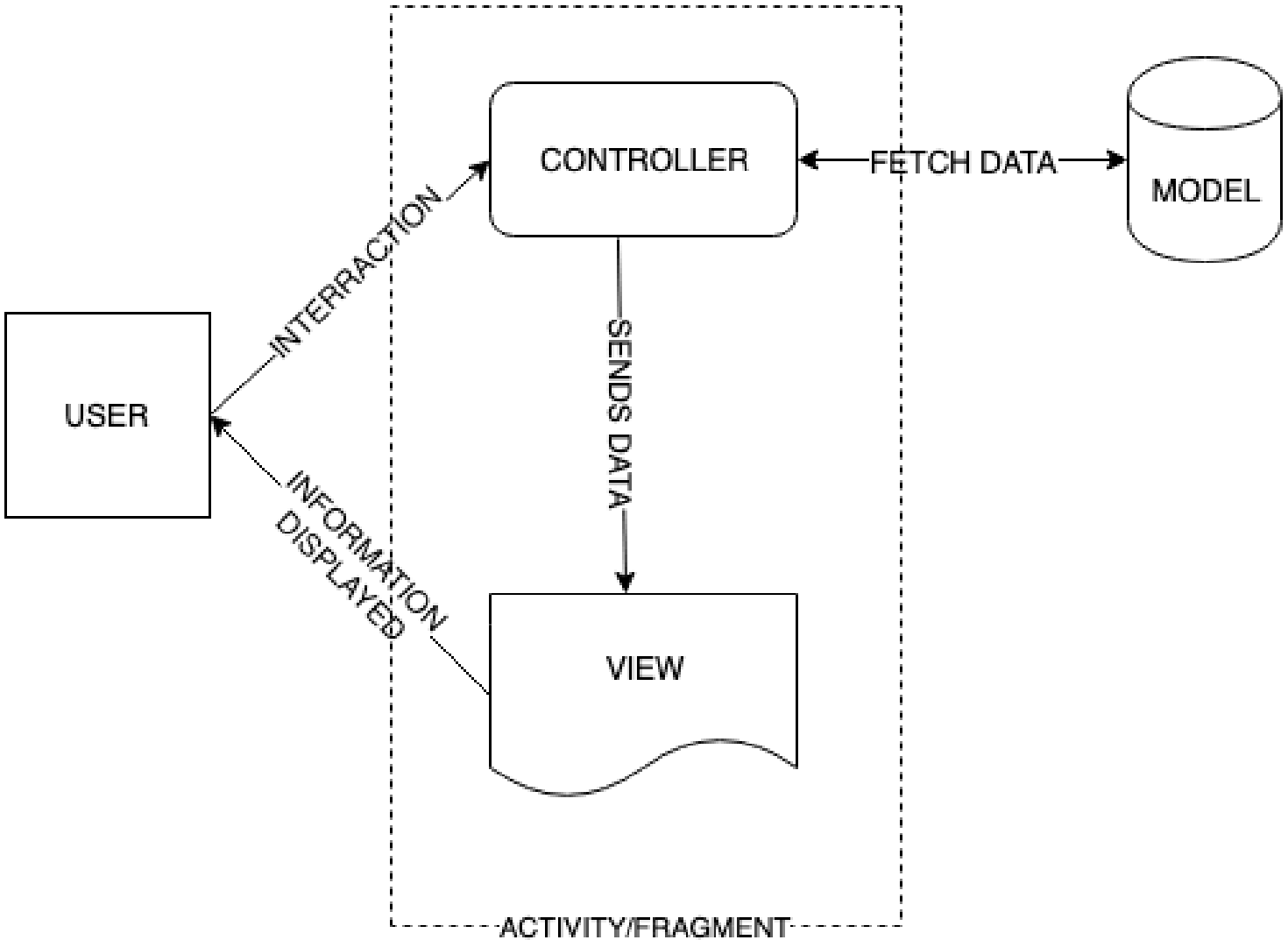
There are many variations of the MV pattern that influence the architecture of Android applications.

Historical Challenges

Implement an app that is easy to maintain, testable, and easy to modify.

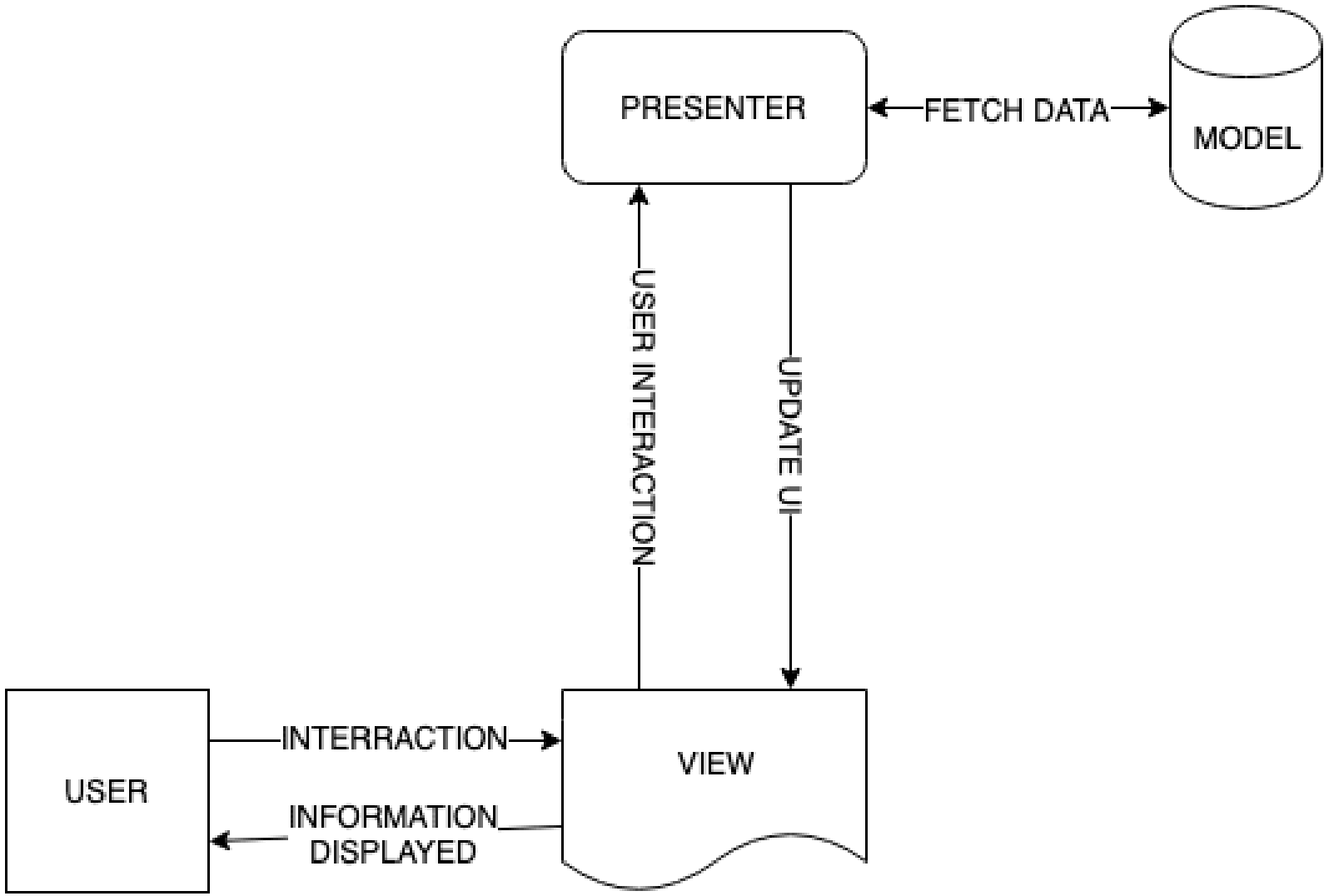
MVC

The model handles the domain logic, the view the UI components, and the controller establishes the relationship between view and model.


CONTROLLER
FETCH DATA
MODEL
INTERRACTION
USER
SENDS DATA
INFORMATION
DISPLAYED
VIEW
ACTIVITY/FRAGMENT

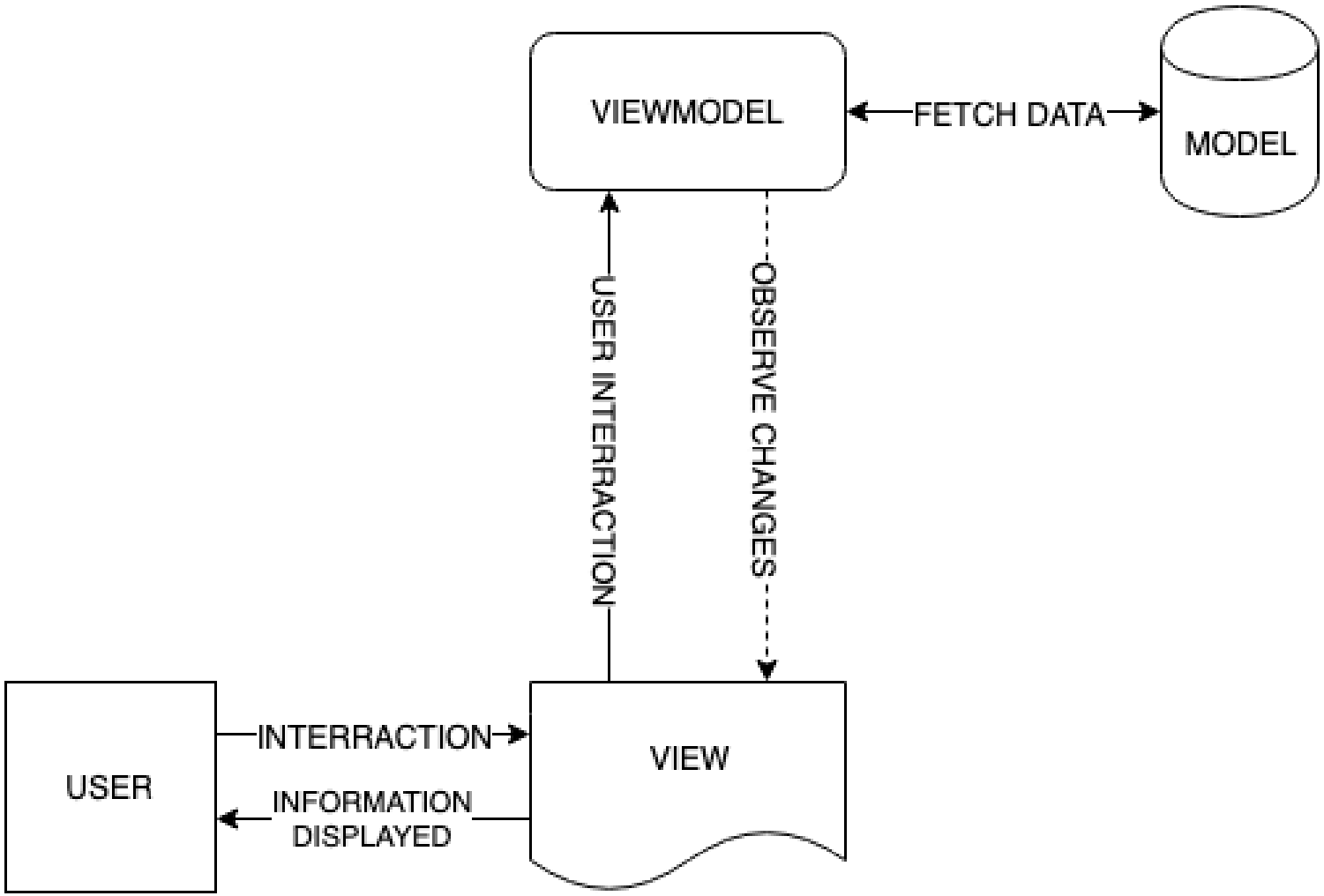
MVP

Similar to MVC, but with a more decoupled view that is exposed through an interface.


PRESENTER
FETCH DATA
MODEL
USER INTERACTION
UPDATE UI
USER
INTERRACTION
VIEW
INFORMATION
DISPLAYED

MVVM

It removes the tight coupling between each component,
the children only have a reference to the observables.


VIEWMODEL
FETCH DATA
MODEL
USER INTERRACTION
OBSERVE CHANGES
USER
INTERRACTION
INFORMATION
DISPLAYED
VIEW

MVI

It leverages the strengths of MVP and MVVM while using reactive streams for unidirectional data flows.

How to Pick a Pattern?

Lessons Learned

There is not a perfect choice and, if there is, it's valid until a new layer is not introduced into your app.

Anectodes and Use Cases

Centralized Teams

A team owns an entire app and its architecture.

Federated Teams (Single Architecture)

App modules and screens are split between teams and
there is a single dominant architecture.

Federated Teams (Multiple Architecture)

App modules and screens are split between teams and
every team owns the architecture of a module/screen.

Conclusions

Thanks!

@giorgionatili