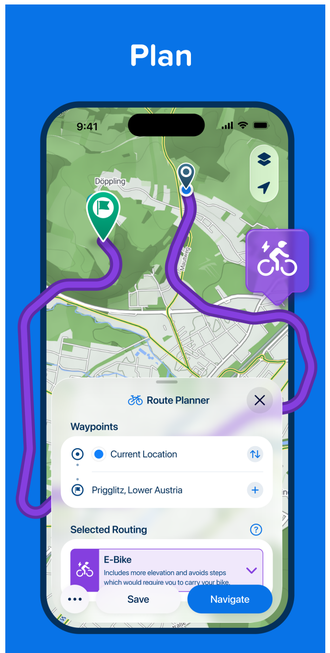
Defense Against Analytics Regressions

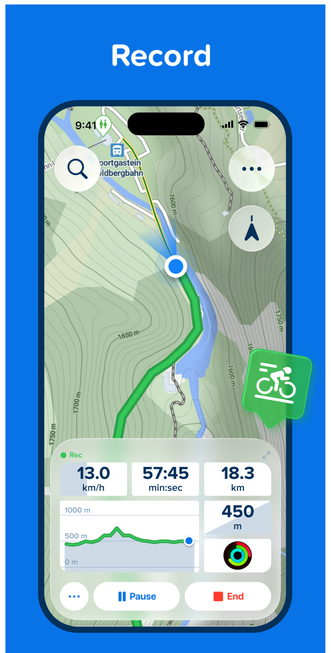
Oleksii Kirizii • # Pragma Conference 2025

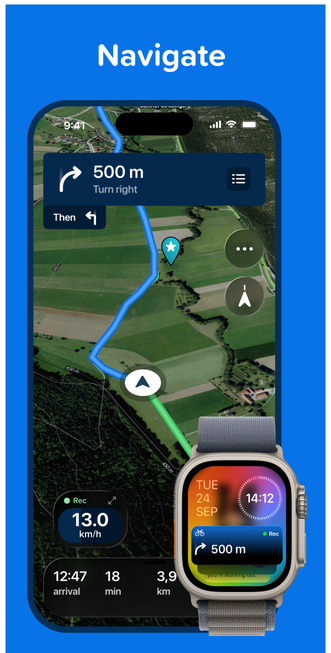
Oleksii Kirizii

Head of iOS @ Bikemap

Bikemap develops apps and services which help cyclists all around the world find safe and enjoyable routes and provides navigation to follow them.



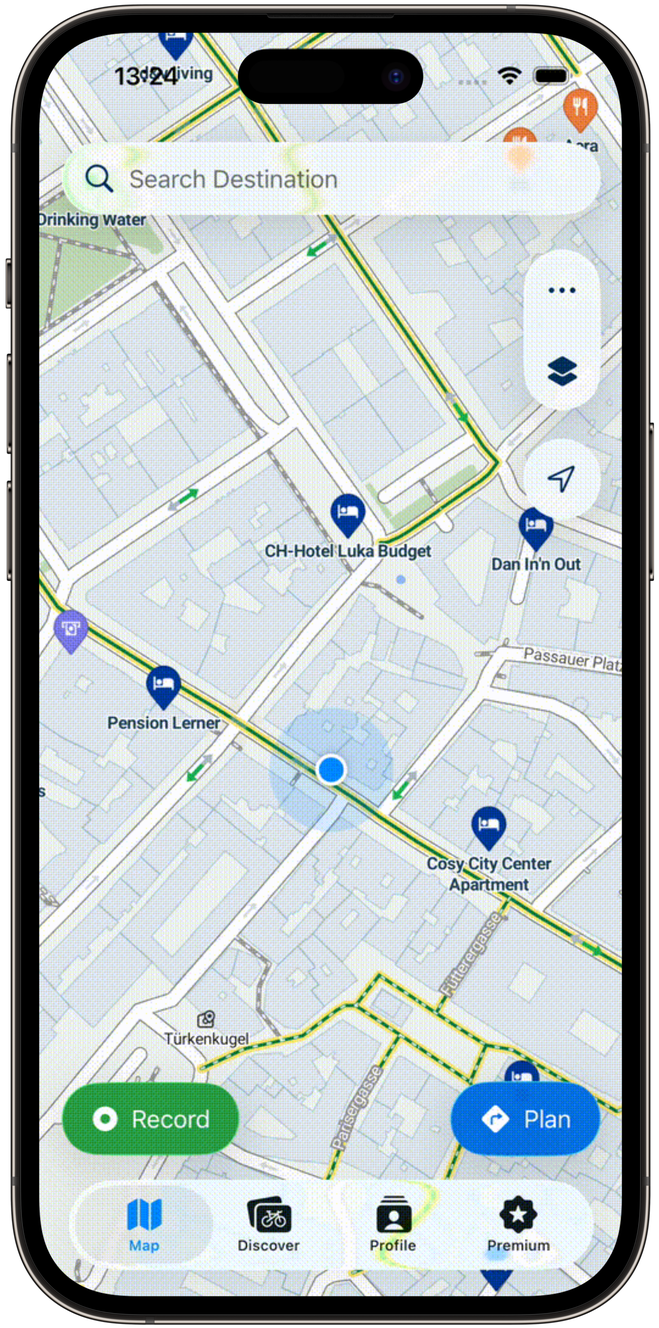




Analytics

Tracking of user’s activities

We would like to know most popular features and how users navigate between them.



Regression

Unintended loss of existing functionality after changes

Verschlimmbesserung

[fɛɐ̯ˈʃlɪmˌbɛsəʁʊŋ]

When the intent to improve actually makes things worse.

Regressions of analytics

One of the worst kind

Hard to spot
app is still running
without any visual signs

Long lived
often detected only
when data are requested

Irreversible
can’t be fixed by a hotfix,
data are lost

How to fight regressions?

Tests. More tests! And maybe double-check manually.

Unit Tests @Test, XCTest

UI Tests XCUITest

Manual Tests Simulator, Device

Will any of these help us with analytics regressions?

What about analytics regressions?

Well, it's complicated…

Unit Tests @Test, XCTest

UI Tests XCUITest

Manual Tests Simulator, Device

Snapshot Tests

Recording of baseline state and comparing with it to identify unexpected changes

What do we want?

Requirements

We want to ensure that:

✅ A specific set of actions triggers **all expected events**

✅ Events have **specified names and parameters**

✅ **No extra**, unexpected **events** are triggered

The basic idea 💡

UI Tests XCUITest + **Snapshot Test** of events

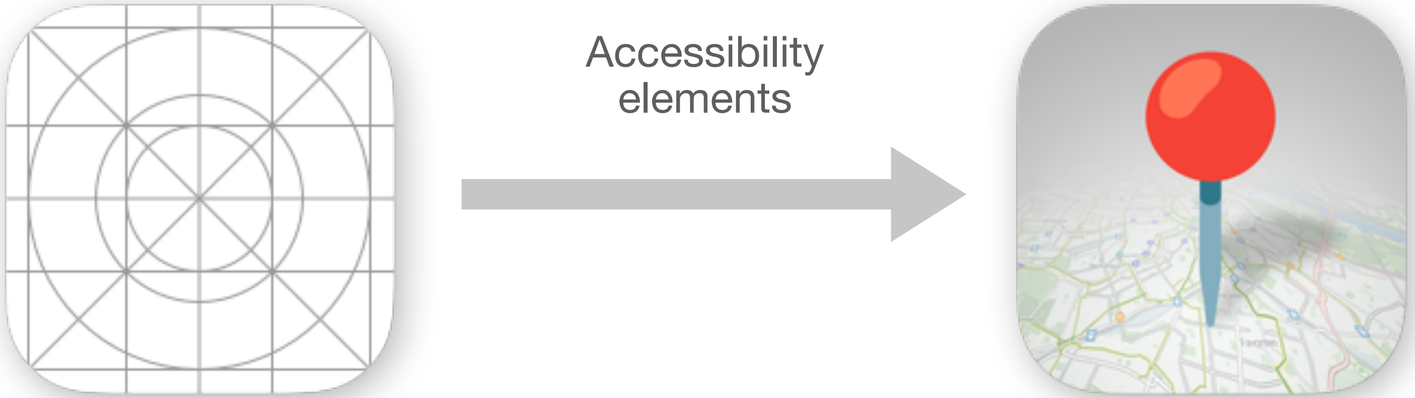
1. We will run a UI test with defined set of actions

2. All events triggered during this test will be recorded and stored locally

3. Next time when we run the same UI test we compare that at the end we have got exactly the same set of events with the same parameters

Anatomy of XCUITest

Two apps



POIListUITests-Runner

POIList

Communication methods

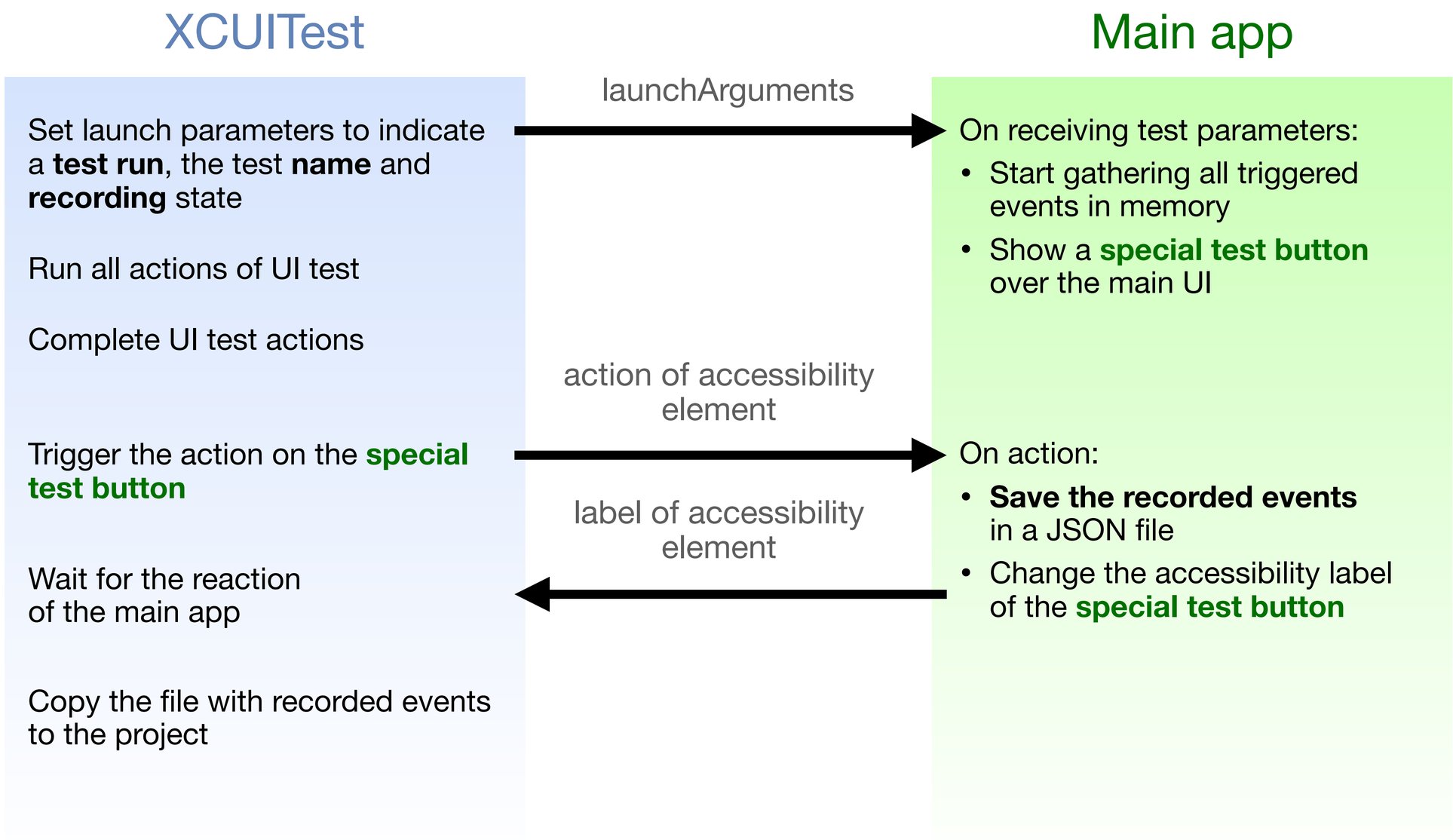
From UI test to the main app

1. Setting environment variables and launch arguments
2. Reading accessibility identifiers and invoking actions of UI elements
3. Writing and reading from clipboard

From main app to UI test

1. Assigning accessibility identifiers to UI elements
2. Writing and reading from clipboard

Creating snapshot



Validating against snapshot

XCUITest

Set launch parameters to indicate a **test run** and the test **name**

launchArguments

Run all actions of UI test

Complete UI test actions

action of accessibility element

Trigger the action on the **special test button**

label of accessibility element

Wait for the reaction of the main app

Depending on the reaction either **mark the test as ✅ or ⛔**

Main app

On receiving test parameters:

- Start gathering all triggered events in memory
- Show a **special test button** over the main UI

On action:

- **Compare gathered events** to snapshot's events
- Change the accessibility label of the **special test button**

Engineering challenges

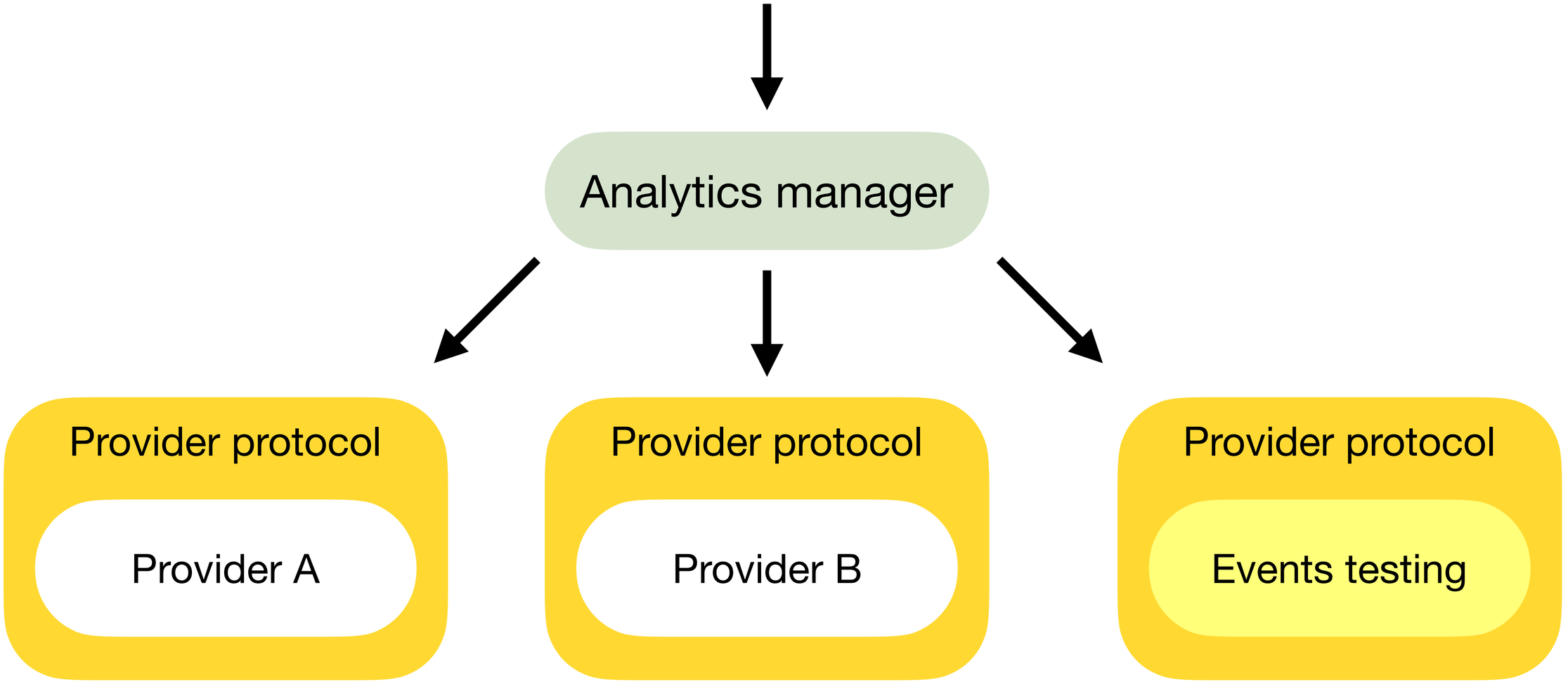
📄 Gathering all events

🛠️ Definition of analytics event

📝 Creating snapshot

Gathering all events

Let’s do some protocols!



Gathering all events

Let’s do some protocols!

```
protocol AnalyticsService {

  func initialize()

  func trackEvent(_ event: AnalyticsEvent)

}

@Observable
final class AnalyticsManager {

  private var services: [AnalyticsService] = []

  init() {

    services = [
      ProviderAService(),
```

Gathering all events

Let’s do some protocols!

```
@Observable
final class AnalyticsManager {

  private var services: [AnalyticsService] = []

  init() {

    if isUITesting {
      services = [TestAnalyticsService()]
    } else {
      services = [
        ProviderAService(),
        ProviderBService()
      ]
    }

    services.forEach {
      $0.initialize()
    }
  }
```

Gathering all events

Let’s do some protocols!

```
    services.forEach {
      $0.initialize()
    }
  }

  func trackEvent(_ event: any AnalyticsEvent) {
    services.forEach {
      $0.trackEvent(event)
    }
  }
```

Definition of analytics event

Each event is a struct

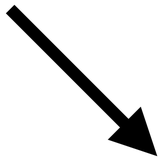
```swift
/// Protocol for analytics event data model
protocol AnalyticsEvent: Codable {

  var defaultEventName: String { get }

  var excludeFromUITestEvaluation: Bool { get }

}
```

```swift
enum ScreenName: String {
  case poiList
  case poiDetail
}

struct OpenScreenEvent: AnalyticsEvent {

  private(set) var defaultEventName = "open_screen"
  private let name: String

  init(screen: ScreenName) {
    name = screen.rawValue
  }
}
```

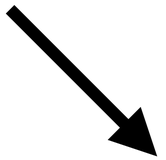
Storing events to JSON

In universal way

```
struct OpenScreenEvent: AnalyticsEvent {

  private(set) var defaultEventName = "open_screen"
  private let name: String

  init(screen: ScreenName) {
    name = screen.rawValue
  }
}
```



```
struct AnalyticsEventData: Codable, Equatable {
  let eventName: String
  let eventParameters: [String: String]
}
```



```
{
 "eventName" : "open_screen",
 "eventParameters" : {
   "name" : "poiList"
 }
```

Creating snapshot

Let’s follow event’s life cycle

1

```
let openScreenEvent = OpenScreenEvent(screen: .poiList)
```

2

```
@Environment(AnalyticsManager.self) private var analytics

analytics.trackEvent(openScreenEvent)
```

3

```
func trackEvent(_ event: AnalyticsEvent) {
  guard event.excludeFromUITestEvaluation == false else {
    return
  }
  self.currentEvents.append(event)
}

private var currentEvents: [AnalyticsEvent]
```

```
struct AnalyticsEventData: Codable, Equatable {
```

Creating snapshot

Let’s follow event’s life cycle

1

```
let openScreenEvent = OpenScreenEvent(screen: .poiList)
```

2

```
@Environment(AnalyticsManager.self) private var analytics

analytics.trackEvent(openScreenEvent)
```

3

```
func trackEvent(_ event: AnalyticsEvent) {
  guard event.excludeFromUITestEvaluation == false else {
    return
  }
  self.currentEvents.append(event)
}

private var currentEvents: [AnalyticsEvent]
```

4

```
struct AnalyticsEventData: Codable, Equatable {
  let eventName: String
  let eventParameters: [String: String]
}
```

Creating snapshot

Let’s follow event’s life cycle

3

```
func trackEvent(_ event: AnalyticsEvent) {
  guard event.excludeFromUITestEvaluation == false else {
    return
  }
  self.currentEvents.append(event)
}

private var currentEvents: [AnalyticsEvent]
```

4

```
struct AnalyticsEventData: Codable, Equatable {
  let eventName: String
  let eventParameters: [String: String]
}
```

5

```
var normalisedParameters: [String: String] = [:]
for (parameter, value) in self.parameters {
  normalisedParameters[parameter] = "\(value)"
```

Creating snapshot

Let’s follow event’s life cycle

4

```
struct AnalyticsEventData: Codable, Equatable {
  let eventName: String
  let eventParameters: [String: String]
}
```

5

```
var normalisedParameters: [String: String] = [:]
for (parameter, value) in self.parameters {
  normalisedParameters[parameter] = "\(value)"
}
let eventData = AnalyticsEventData(eventName: self.name(for: .uiTest),
                                   eventParameters: normalisedParameters)
```

6

```
let encoder = JSONEncoder()
encoder.outputFormatting = .prettyPrinted
let jsonString = try? encoder.encode(eventData)
```

Creating snapshot

Let’s follow event’s life cycle

5

```
var normalisedParameters: [String: String] = [:]
for (parameter, value) in self.parameters {
  normalisedParameters[parameter] = "\(value)"
}
let eventData = AnalyticsEventData(eventName: self.name(for: .uiTest),
                                   eventParameters: normalisedParameters)
```

6

```
let encoder = JSONEncoder()
encoder.outputFormatting = .prettyPrinted
let jsonString = try? encoder.encode(eventData)
```

7

```
{
 "eventName" : "open_screen",
 "eventParameters" : {
   "name" : "poiList"
 }
```

Creating snapshot

Let’s follow event’s life cycle

6

```
let encoder = JSONEncoder()
encoder.outputFormatting = .prettyPrinted
let jsonString = try? encoder.encode(eventData)
```

7

```
{
 "eventName" : "open_screen",
 "eventParameters" : {
   "name" : "poiList"
 }
```

Creating snapshot

Let’s follow event’s life cycle

7

```
{
 "eventName" : "open_screen",
 "eventParameters" : {
   "name" : "poiList"
 }
```

Normalising events parameters

```swift
extension AnalyticsEvent {

  var parameters: [String: Any] {

    let mirror = Mirror(reflecting: self)

    var allParameters: [String: Any] = [:]
    for property in mirror.children {

      // Don't report parameter in case when it's optional and contains nil
      if case Optional<Any>.none = property.value { continue }

      guard let propertyLabel = property.label else { continue }

      // Event name itself should excluded from parameters
      guard propertyLabel != "defaultEventName" else { continue }

      allParameters[propertyLabel] = property.value
    }
    return allParameters
  }
```

Normalising events parameters

```
extension AnalyticsEvent {

  var parameters: [String: Any] {

    let mirror = Mirror(reflecting: self)

    var allParameters: [String: Any] = [:]
    for property in mirror.children {

      // Don't report parameter in case when it's optional and contains nil
      if case Optional<Any>.none = property.value { continue }

      guard let propertyLabel = property.label else { continue }

      // Event name itself should excluded from parameters
      guard propertyLabel != "defaultEventName" else { continue }

      allParameters[propertyLabel] = property.value
    }
    return allParameters
  }
```

Checking equality

Neither less, nor more

Snapshot	Test events	Check outcome
Event #1	Event #3 (y)	✅
Event #2	Event #1	⛔
Event #3 (x)	Event #4	⛔
Event #4	Event #5	✅
		⛔

Checking equality

```
var snapshotEventsData = events
while snapshotEventsData.isEmpty == false {
  guard let checkEvent = snapshotEventsData.first else {
    return .failed("No events in recorded events list")
  }
  var eventFound = false
  var eventNotFoundError: String?

  for i in 0..<gatheredEventsData.count {
    let event = gatheredEventsData[i]

    if event == checkEvent {
      eventFound = true
      gatheredEventsData.remove(at: i)
      break
    } else if event.eventName == checkEvent.eventName {
      eventNotFoundError = self.compareEventsParameters(recordedEvent: checkEvent,
                                                        currentEvent: event)

    }
  }
```

Checking equality

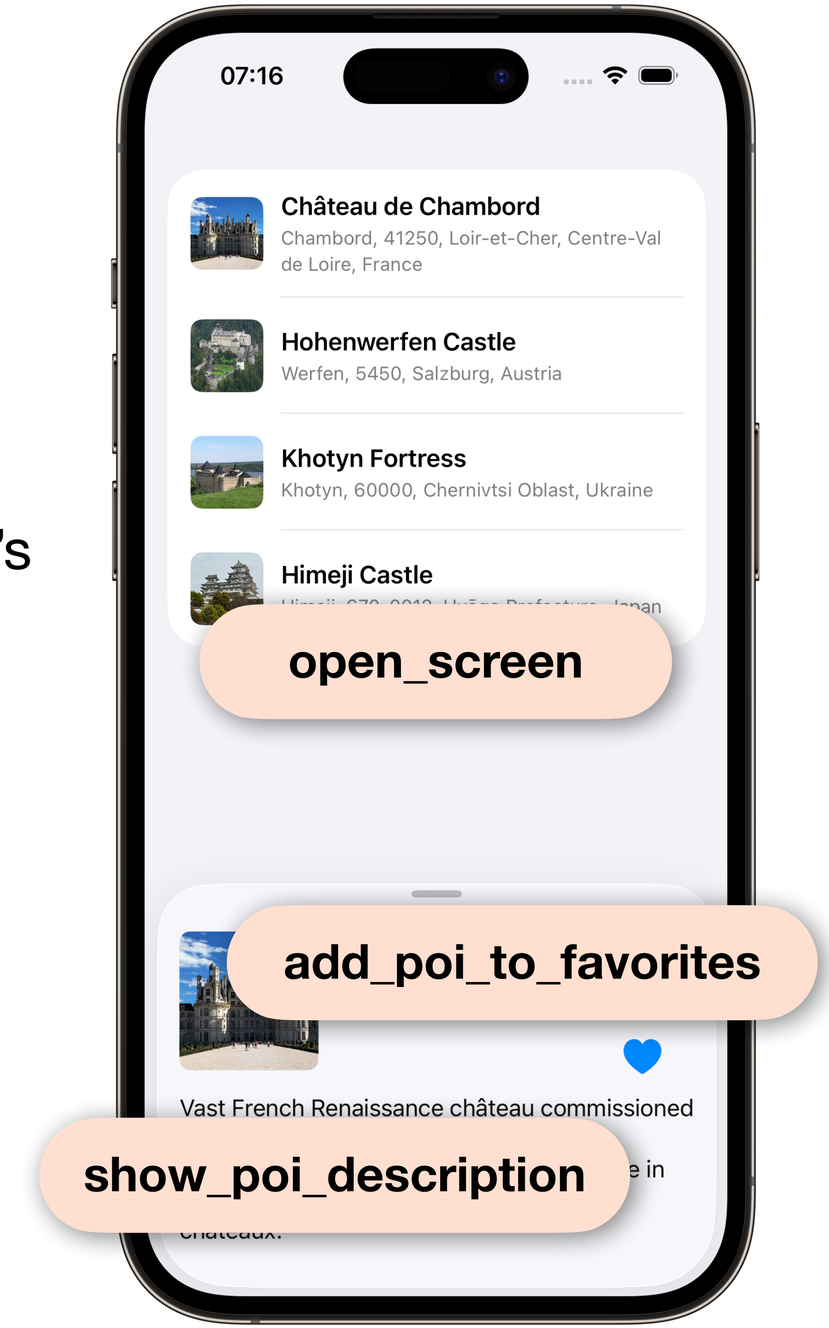
```
    if event == checkEvent {
      eventFound = true
      gatheredEventsData.remove(at: i)
      break
    } else if event.eventName == checkEvent.eventName {
      eventNotFoundError = self.compareEventsParameters(recordedEvent: checkEvent,
                                                        currentEvent: event)

    }
  }

  if eventFound == false {
    if let errorMessage = eventNotFoundError {
      return .failed("In test \(testName) the event \(checkEvent.eventName) has a difference in
          parameters: \(errorMessage)")
    }

    return .failed("In test '\(testName)' the event '\(checkEvent.eventName)' was not emitted,
        even though it is in the recorded events json file")
  }

  snapshotEventsData.removeFirst()
}
```

And now all parts in action!

In POIList demo app

The demo app shows a simple list of attractions.
We can open details for each one and add it to favorites.

In this flow, three events are emitted and we will ensure that it’s true and stays like this.

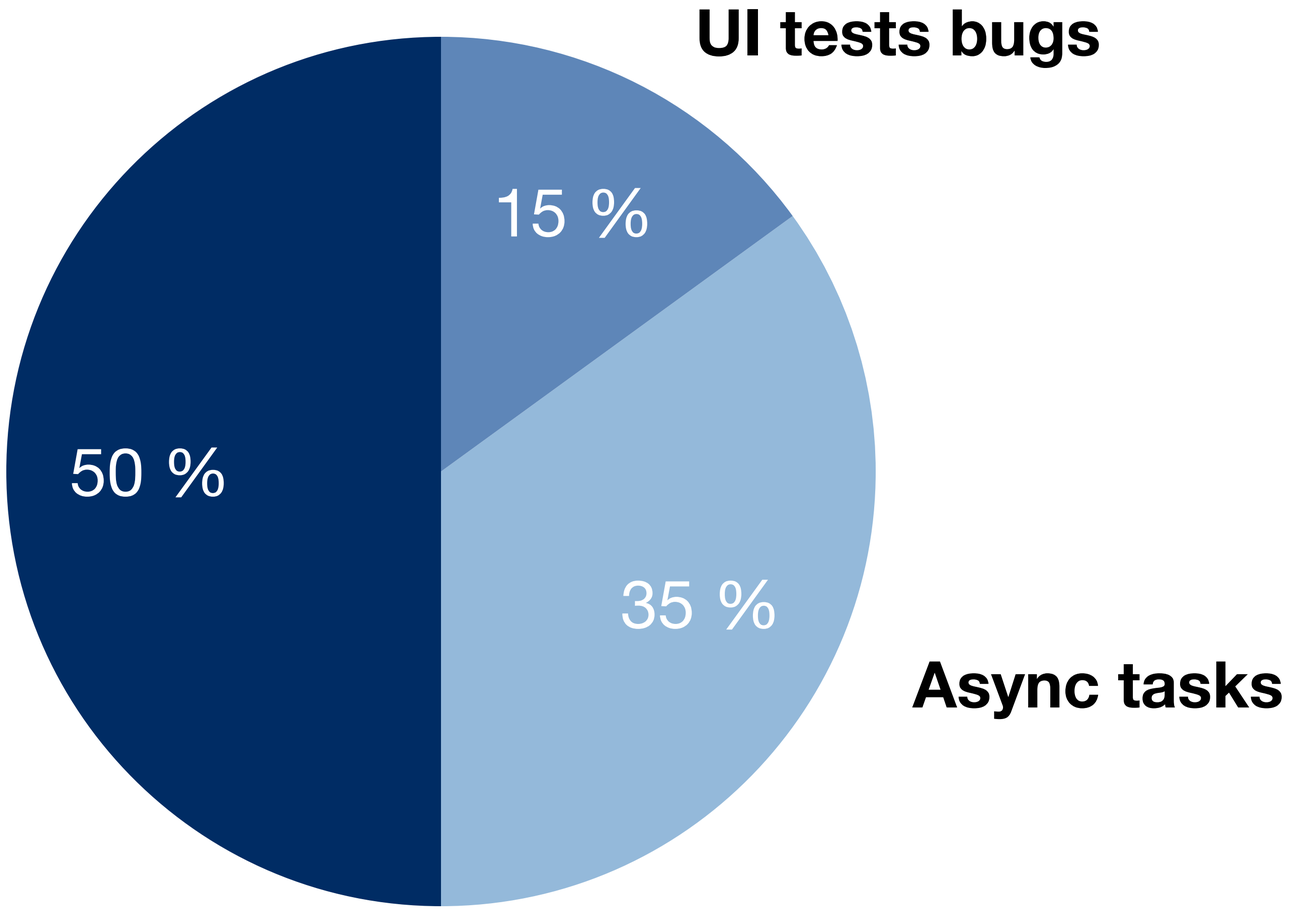


Why not UIPasteboard?

- 🛑 Also used for other operations, like text input.
- 🛑 Requires manual consent

Stability of UI Tests

Subjective opinion



Additional PRO-tip

Speed your tests up!

```
if ProcessInfo.processInfo.isUITesting {
  UIApplication.currentWindow?.layer.speed = 100
}
```

Pro: UI tests are executed much faster. This approach can save minutes in each run.
And help to identify rare race conditions.

Contra: this increases probability of race conditions, which is normally mitigated by animation duration.

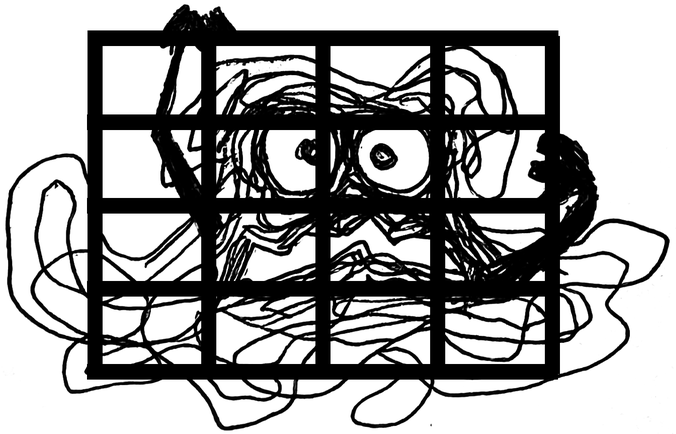
Conclusion

✅ Snapshot testing is a powerful, universal technique.

✅ There are ways to organize bidirectional communication between XCUITests and the app.

✅ Analytics architecture could be type-strict but still flexible enough to cover any events.

✅ Regressions are unavoidable but detectable and can be kept under control.



Thank you for watching!

Code example

https://github.com/OlKir/defense-against-analytics-regressions.git

E-mail
oleksii.kirizii@gmail.com

Mastodon
mastodon.social/@olkir

X/Twitter
OKirizii