bumble inc.
kind connections

Adopting SwiftUI at scale

Alexis Santos

Bumble Inc. is the parent company of Bumble, Badoo, and Fruitz.

The Bumble platform enables people to connect and build healthy and equitable relationships.

Our apps are available in more than **190 countries**, translated in more than **50 languages**.

We have 3 offices location (London, Barcelona and Austin) and more than 900 employees

SwiftUI

“SwiftUI helps you build great-looking apps across all Apple platforms with the power of Swift — and surprisingly little code.
You can bring even better experiences to everyone, on any Apple device, using just one set of tools and APIs.”

SwiftUI

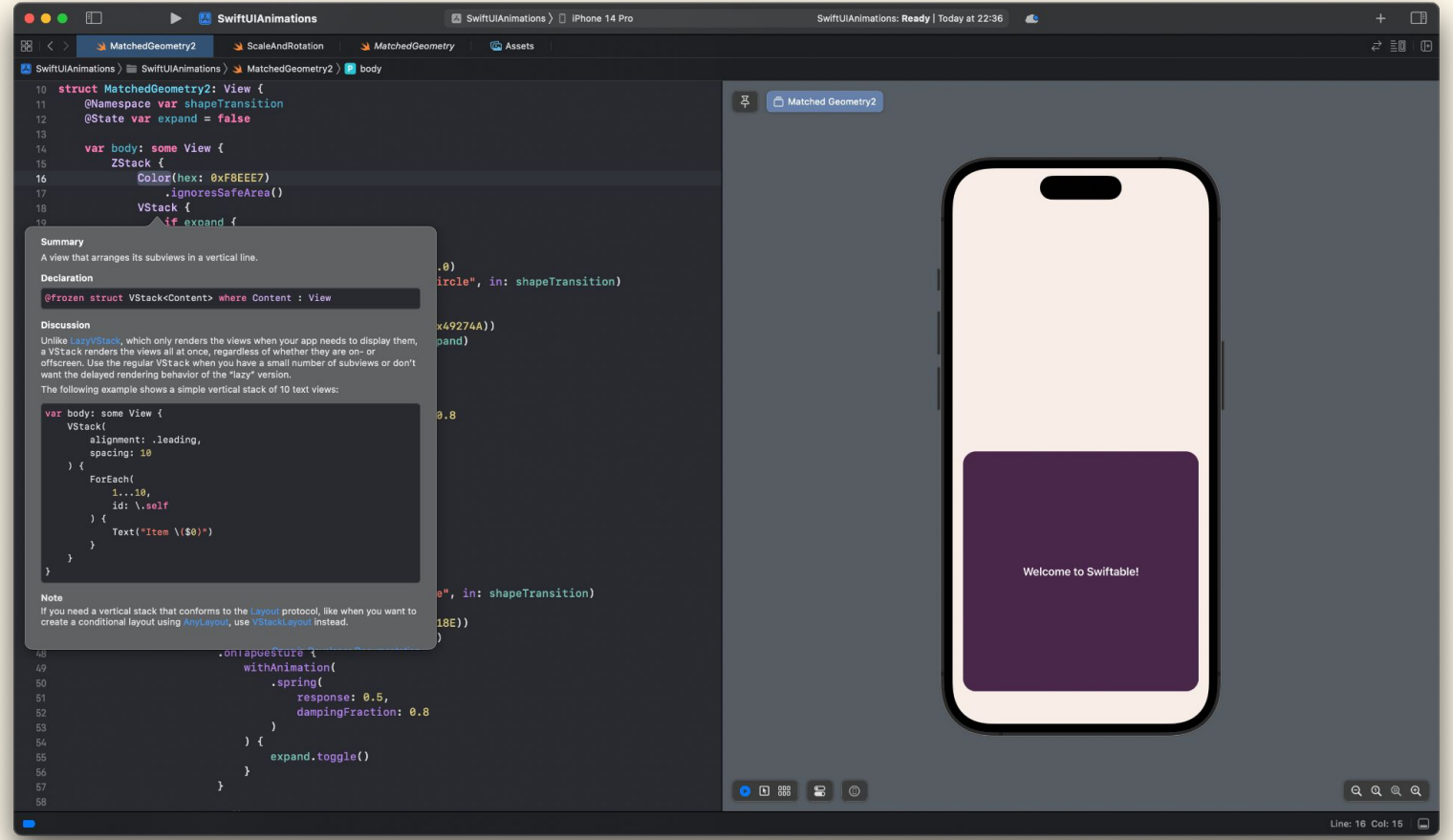
“SwiftUI helps you build great-looking apps across all Apple platforms with the power of Swift — and surprisingly little code.
You can bring even better experiences to everyone, on any Apple device, using just one set of tools and APIs.”

SwiftUI

“SwiftUI helps you build great-looking apps across all Apple platforms with the power of Swift — and surprisingly little code.
You can bring even better experiences to everyone, on any Apple device, using just one set of tools and APIs.”

SwiftUI

“SwiftUI helps you build great-looking apps across all Apple platforms with the power of Swift — and surprisingly little code.
You can bring even better experiences to everyone, on any Apple device, using just one set of tools and APIs.”


SwiftUIAnimations
SwiftUIAnimations 〉 iPhone 14 Pro
SwiftUIAnimations: Ready | Today at 22:36
MatchedGeometry2
ScaleAndRotation
MatchedGeometry
Assets
SwiftUIAnimations 〉 SwiftUIAnimations 〉 MatchedGeometry2 〉 body
struct MatchedGeometry2: View {
@Namespace var shapeTransition
@State var expand = false
var body: some View {
ZStack {
Color(hex: 0xF8EEE7)
.ignoresSafeArea()
VStack {
if expand {
Summary
A view that arranges its subviews in a vertical line.
Declaration
@frozen struct VStack<Content> where Content : View
Discussion
Unlike LazyVStack, which only renders the views when your app needs to display them, a VStack renders the views all at once, regardless of whether they are on- or offscreen. Use the regular VStack when you have a small number of subviews or don't want the delayed rendering behavior of the "lazy" version.
The following example shows a simple vertical stack of 10 text views:
var body: some View {
VStack(
alignment: .leading,
spacing: 10
) {
ForEach(
1...10,
id: \.self
) {
Text("Item \($0)")
}
}
}
Note
If you need a vertical stack that conforms to the Layout protocol, like when you want to create a conditional layout using AnyLayout, use VStackLayout instead.
.onTapGesture {
withAnimation(
.spring(
response: 0.5,
dampingFraction: 0.8
)
) {
expand.toggle()
}
}
Matched Geometry2
Welcome to Swiftable!
Line: 16 Col: 15

What about the code syntax?

```
import SwiftUI

struct AlbumDetail: View {

    var album: Album

    var body: some View {
        List(album.songs) { song in
            HStack {
                Image(album.cover)
                VStack(alignment: .leading) {
                    Text(song.title)
                    Text(song.artist)
                        .font(.title)
                }
            }
        }
    }
}
```

… and the animations?

SwiftUI is amazing!

This talk is about being strategical

Almost 3 years ago...

We need to be faster!

SwiftUI to the rescue!

SwiftUI is amazing!

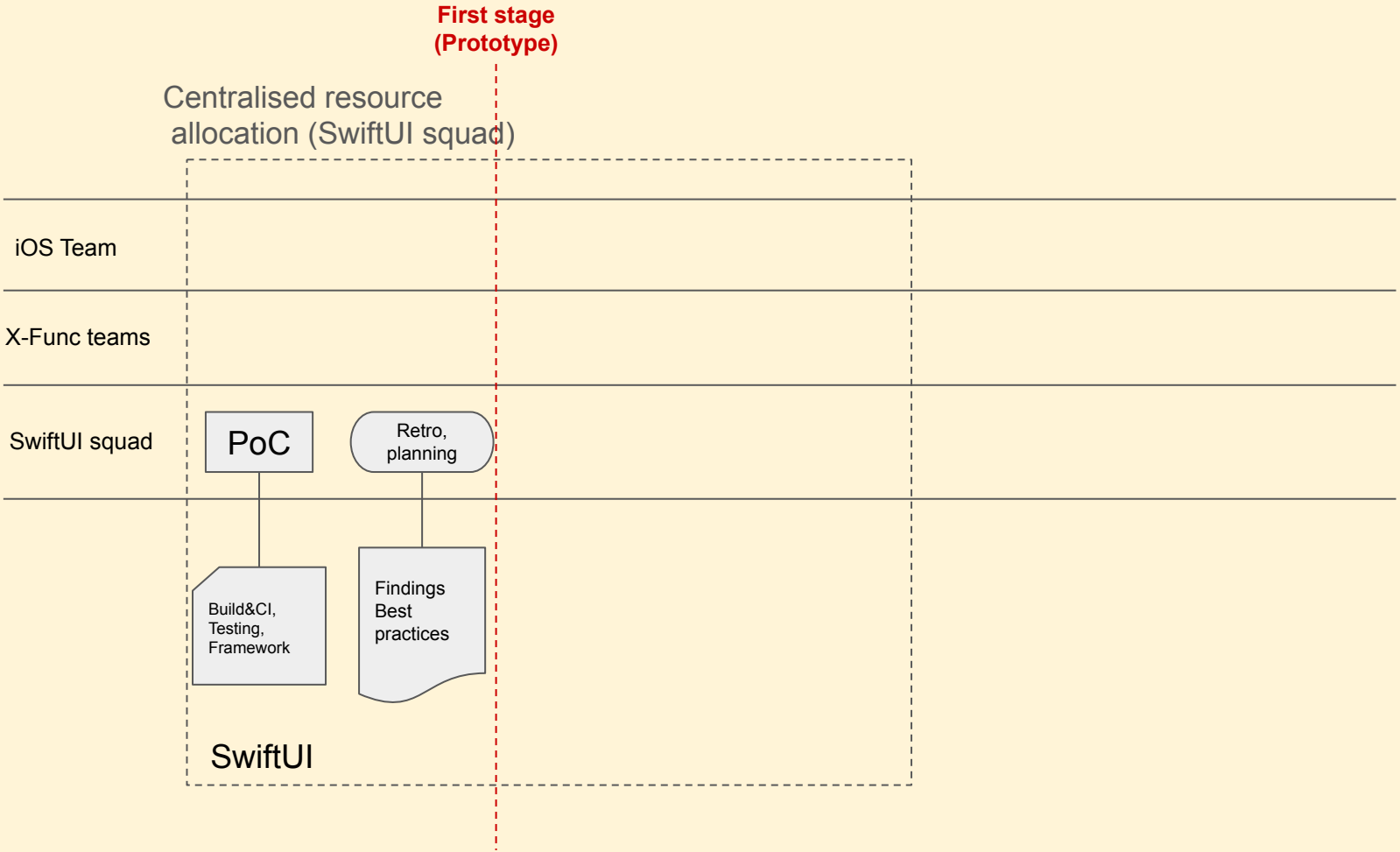
Is SwiftUI the right choice!?

SwiftUI is…
uhmm
let me check again!

Our challenges

- Older iOS versions support
- SwiftUI 1.0 stability
- 330+ internal modules
- Multiple apps sharing code
- A rapidly growing team
- Lack of expertise in SwiftUI

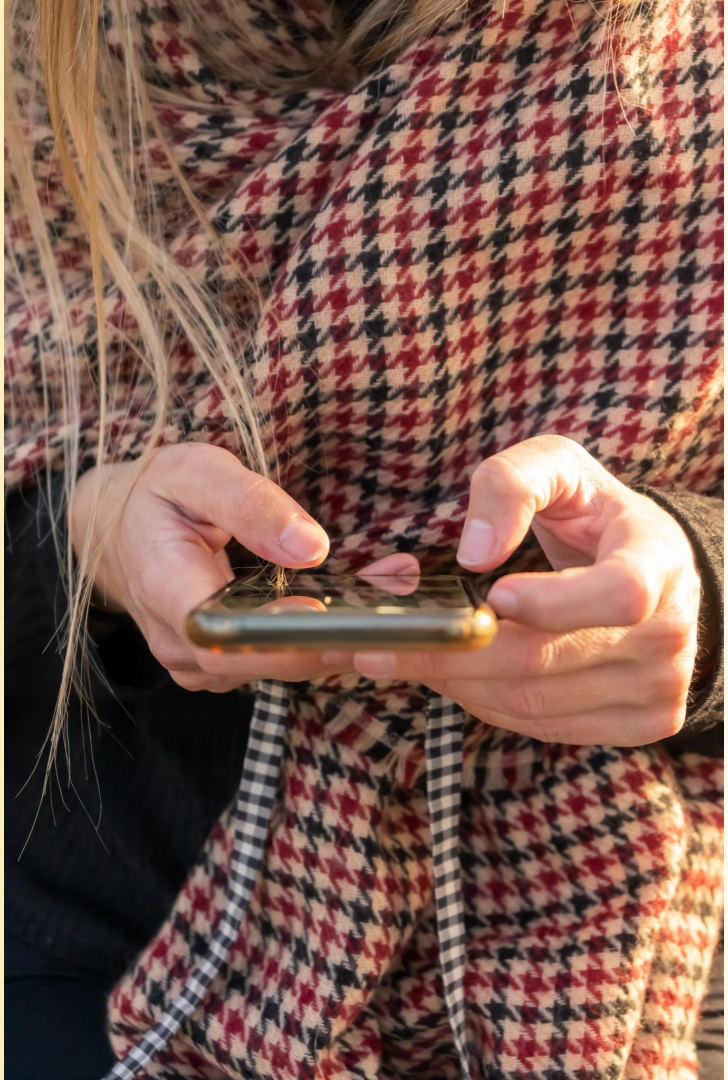
The plan


First stage
(Prototype)
Centralised resource
allocation (SwiftUI squad)
iOS Team
X-Func teams
SwiftUI squad
PoC
Retro,
planning
Build&CI,
Testing,
Framework
Findings
Best
practices
SwiftUI

Start small

- Initial production-ready prototype
- Low traffic part of the app
- Under A/B test with remote control

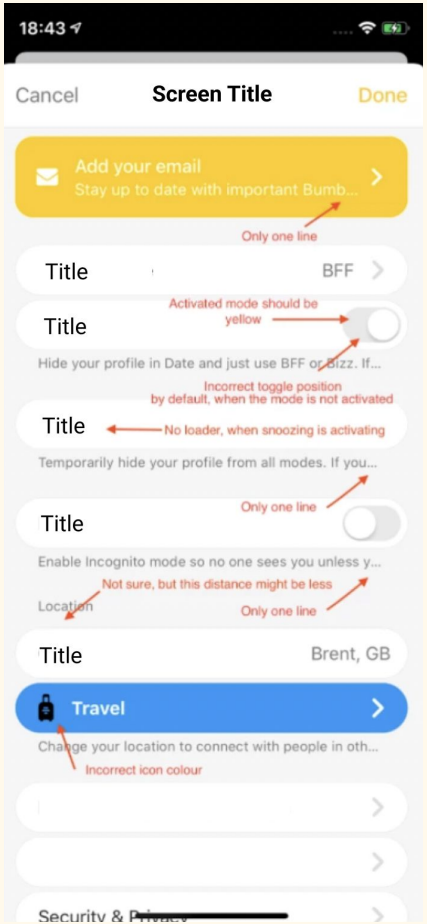
Tip: Keep a close eye on the potential crashes and be ready to react quickly



Involve QA and Automation early

- Adapt automation tools for SwiftUI
- Involve QA in the process

Tip: iOS 13.0 contains many difficult bugs, prefer iOS 13.1+



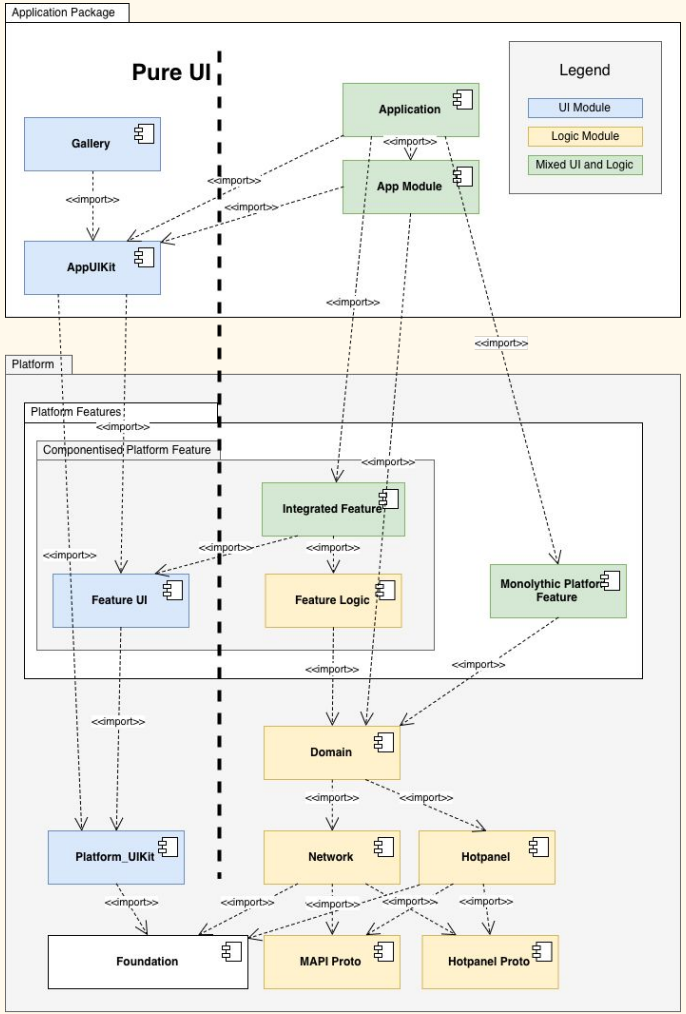
Minimum required target

- Found many hard issues for SwiftUI in iOS 13.0
- Moved Bumble to 13.1+
- We kept Badoo in iOS 12.0 a bit longer (not recommended)

Tip: Prefer SwiftUI 2.0 or higher, move to iOS 14.0+

Multiple modules support

- We support hundreds of modules
- They help us scale and reduce compilation time
- Many of them contains UI Logic (we call them Workspaces)



bumble inc.

Multiple modules support

- Conditional imports
- Conditional features for specific iOS
- Provisional changes with deprecation plans

Tip: We found this approach mentally hard to maintain, document your decisions

```
#if canImport(YourFramework)
import YourFramework
#endif
```

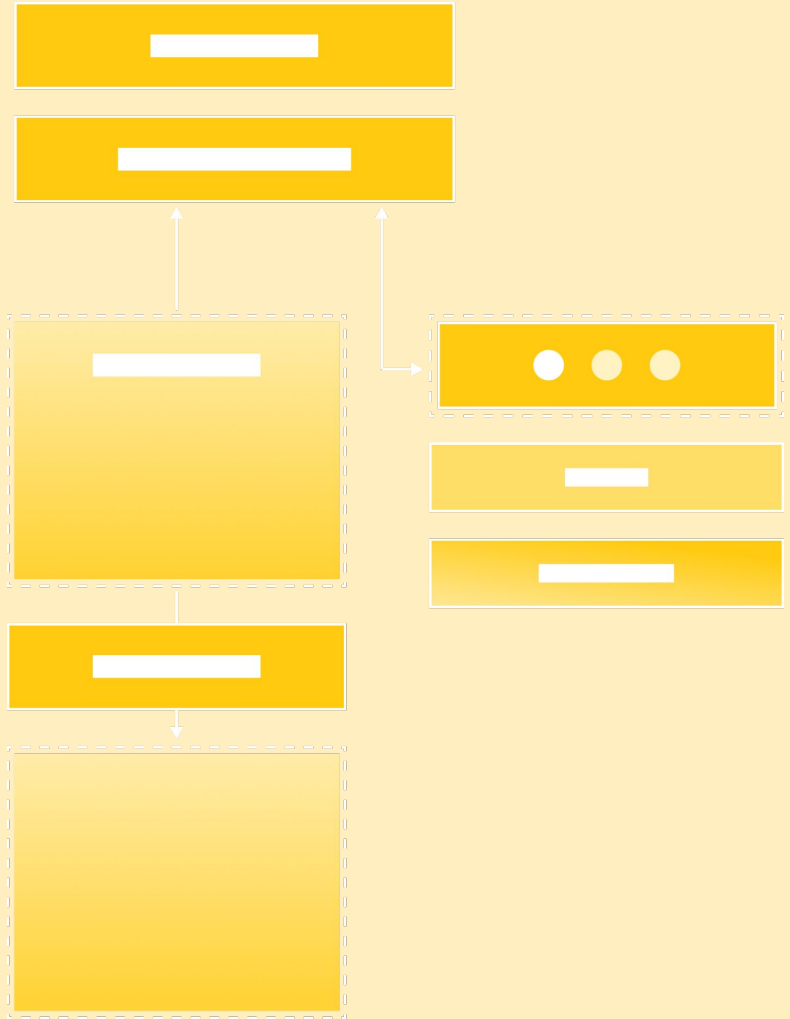
```
if #available(iOS 14.0, *) {
    print("your code goes here")
}
```

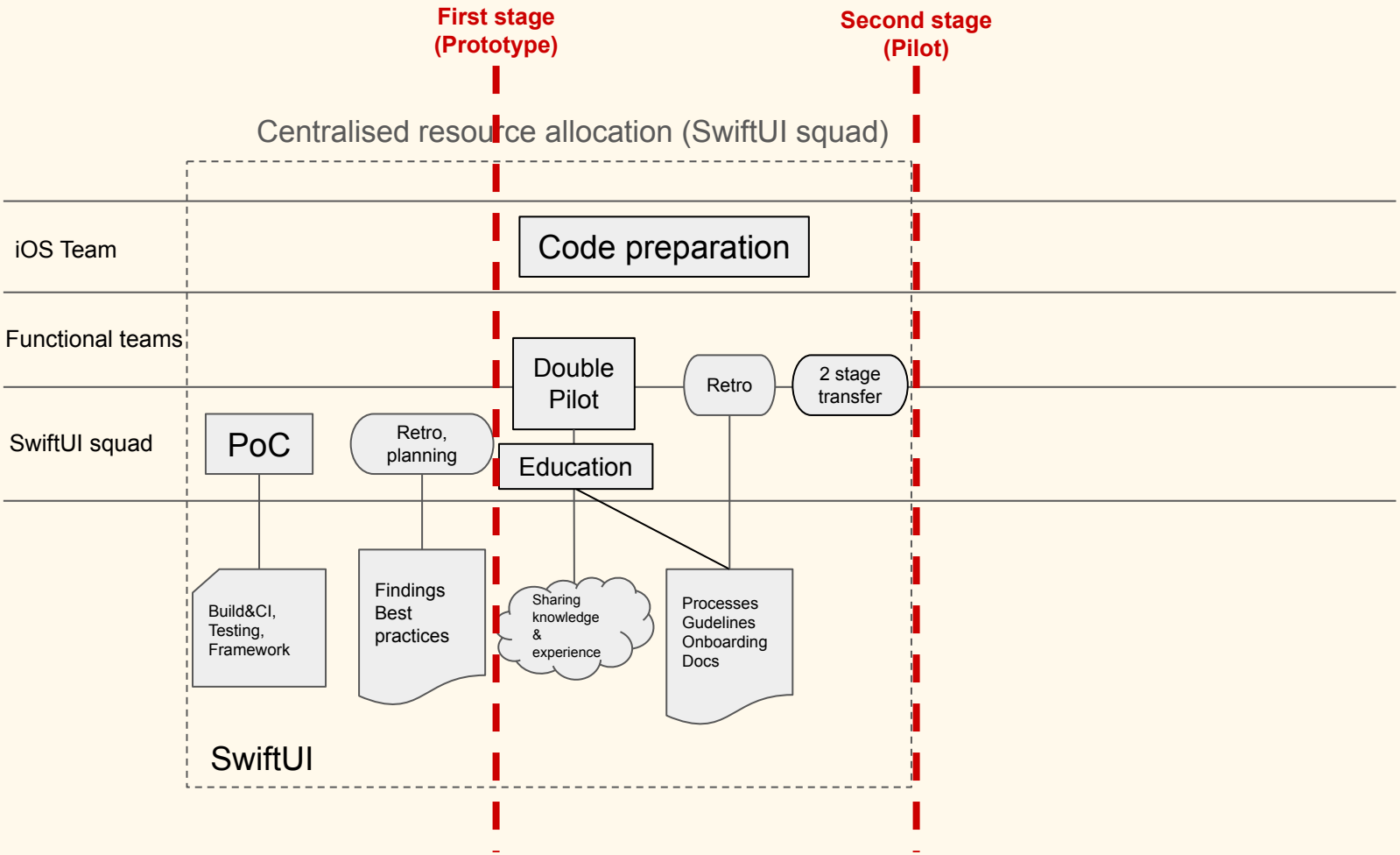
```
@available(iOS,
           introduced: 13.0,
           deprecated: 14.0)
```

Interop SwiftUI and UIKit

- ***UIHostingController*** is your friend
- ***UIViewRepresentable*** is difficult to master
- Avoid embed UIKit views into SwiftUI

Tip: Easier to rewrite the component to SwiftUI than using UIViewRepresentable approach (in most cases)




First stage
(Prototype)
Second stage
(Pilot)
Centralised resource allocation (SwiftUI squad)
iOS Team
Code preparation
Functional teams
Double
Pilot
Retro
2 stage
transfer
SwiftUI squad
PoC
Retro,
planning
Education
Build&CI,
Testing,
Framework
Findings
Best
practices
Sharing
knowledge
&
experience
Processes
Gudelines
Onboarding
Docs
SwiftUI

Increased complexity

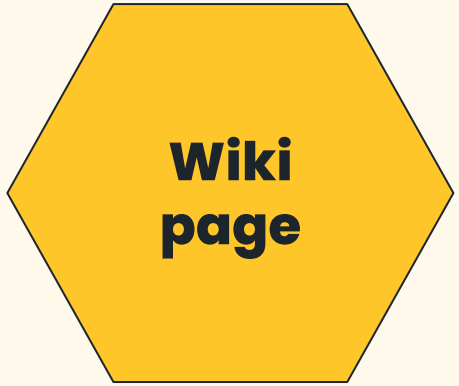
- Plan knowledge sharing
- Involve other teams and provide them support
- Validate your assumptions
- Stage the rollout to the rest of teams

Tip: Communicate adoption goals and timelines with your team

Education and training

- Be transparent with the team
- Provide constant support

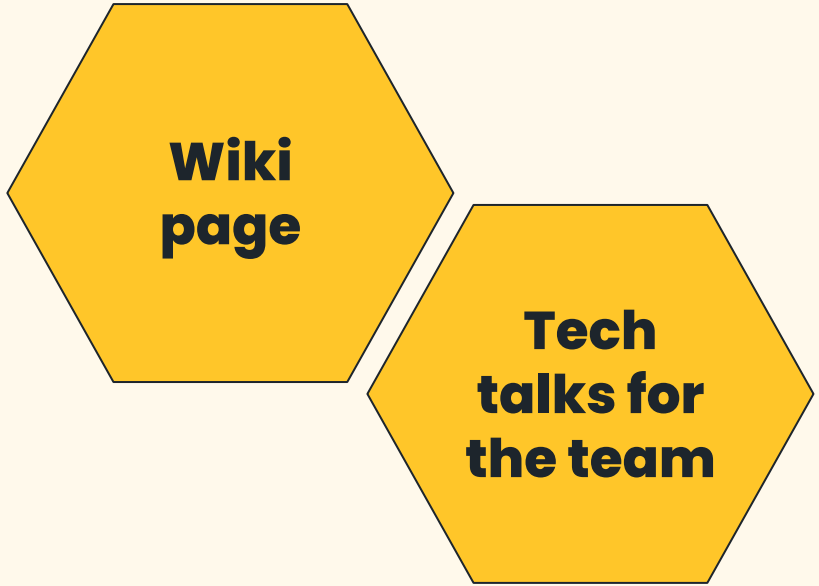
Tip: Do not overlook this part of the project, may be the most important one



Education and training

- Be transparent with the team
- Provide constant support

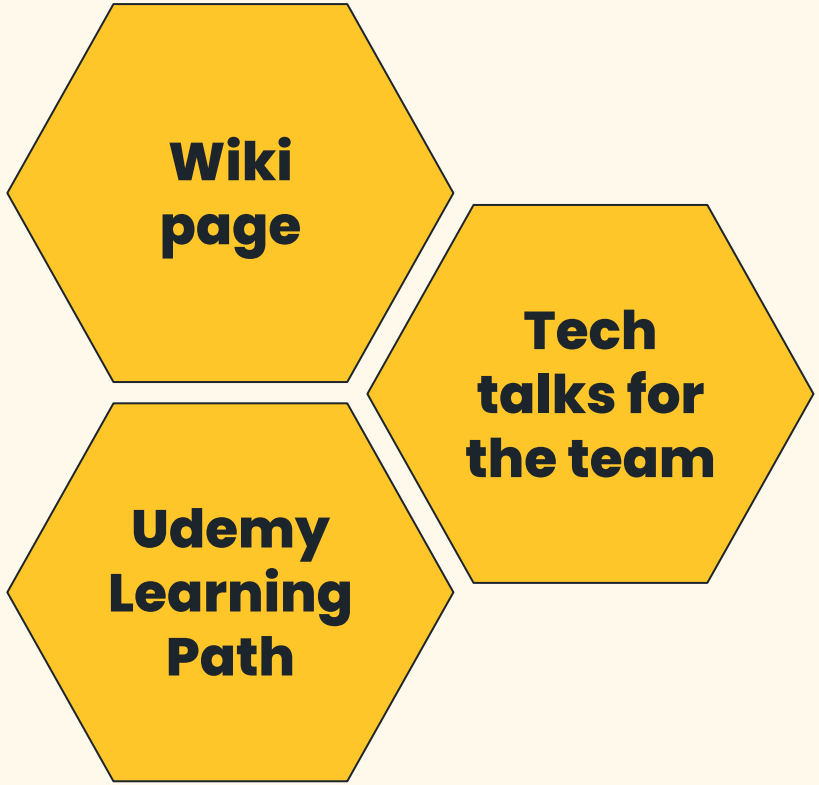
Tip: Do not overlook this part of the project, may be the most important one



Education and training

- Be transparent with the team
- Provide constant support

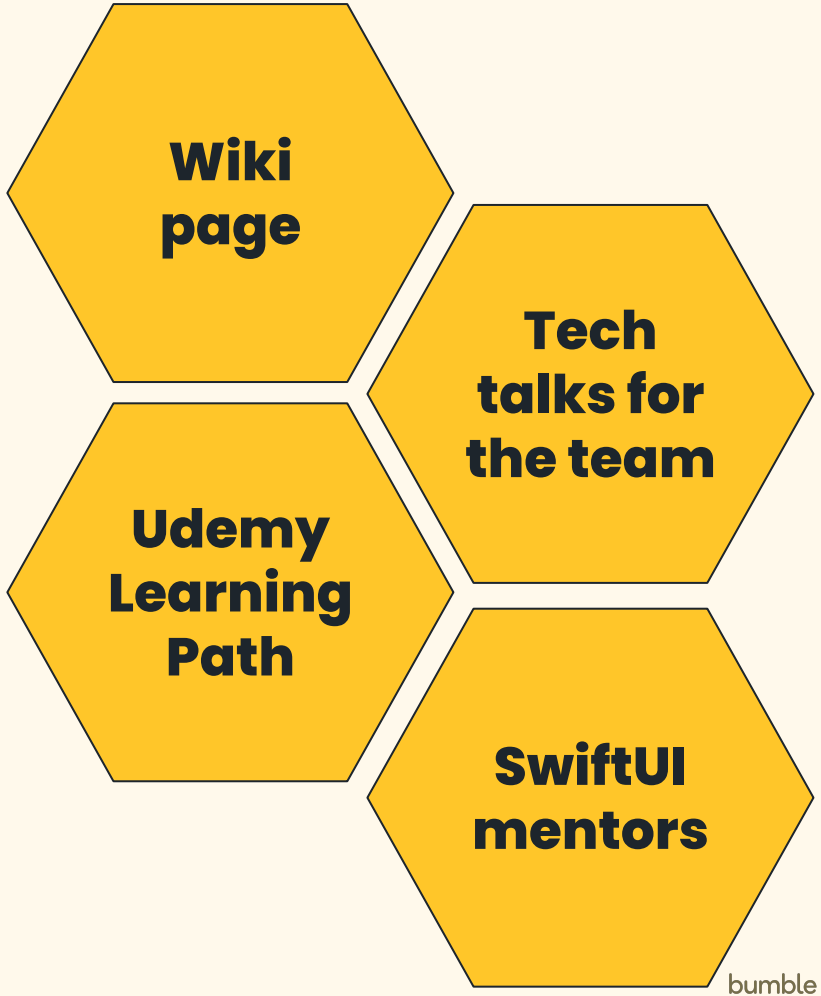
Tip: Do not overlook this part of the project, may be the most important one



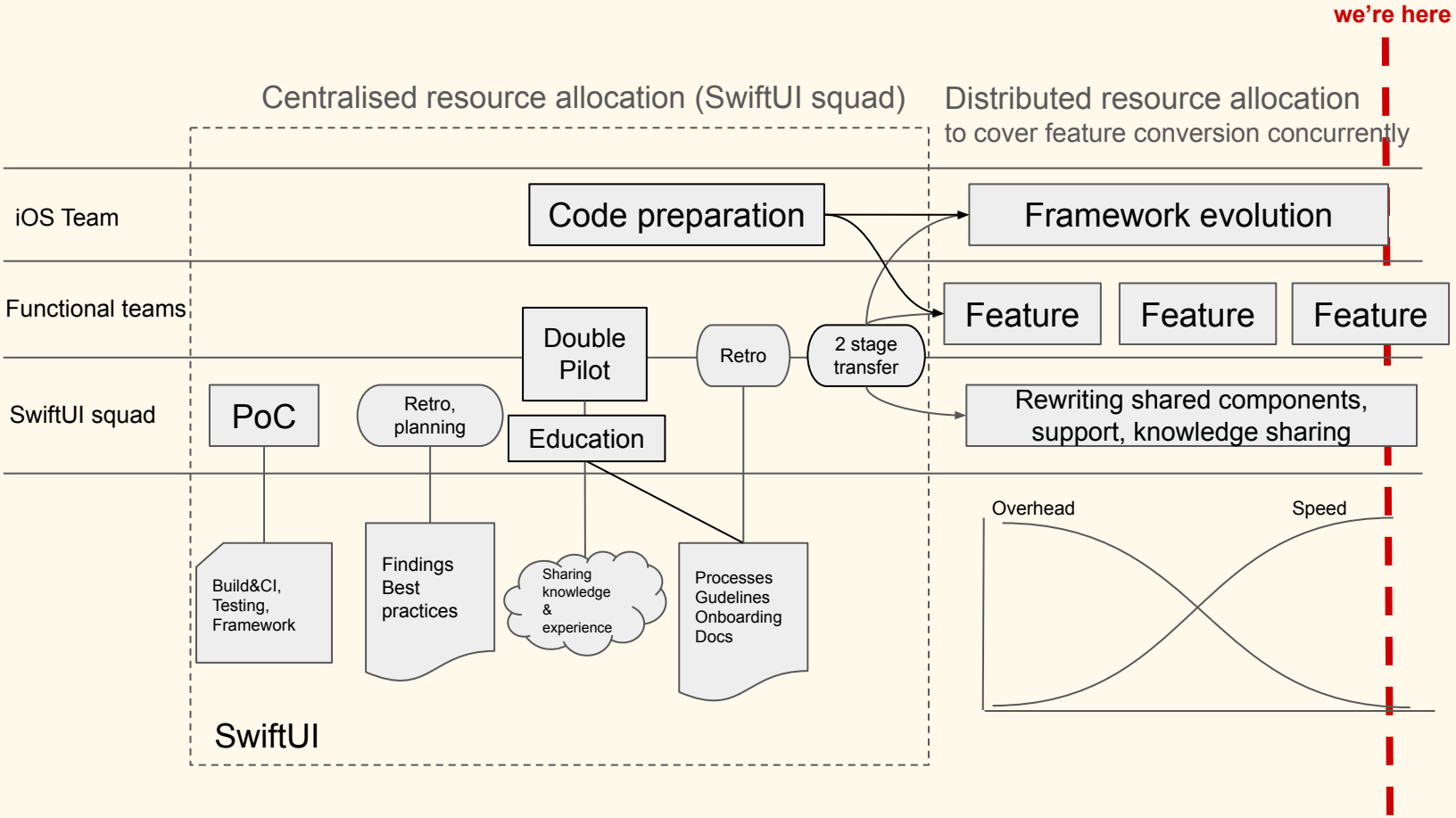
Education and training

- Be transparent with the team
- Provide constant support

Tip: Do not overlook this part of the project, may be the most important one



Current State


we're here
Centralised resource allocation (SwiftUI squad)
Distributed resource allocation
to cover feature conversion concurrently
iOS Team
Functional teams
SwiftUI squad
Code preparation
Framework evolution
Feature
Feature
Feature
Double Pilot
Retro
2 stage transfer
Rewriting shared components, support, knowledge sharing
PoC
Retro, planning
Education
Build&CI, Testing, Framework
Findings Best practices
Sharing knowledge & experience
Processes Gudelines Onboarding Docs
Overhead
Speed
SwiftUI

Current State – 2023

22%

Codebase Adoption

Current State - 2023

22%

Codebase Adoption

90%

Developers transition to SwiftUI

Current State - 2023

22%

Codebase Adoption

90%

Developers transition to SwiftUI

70%

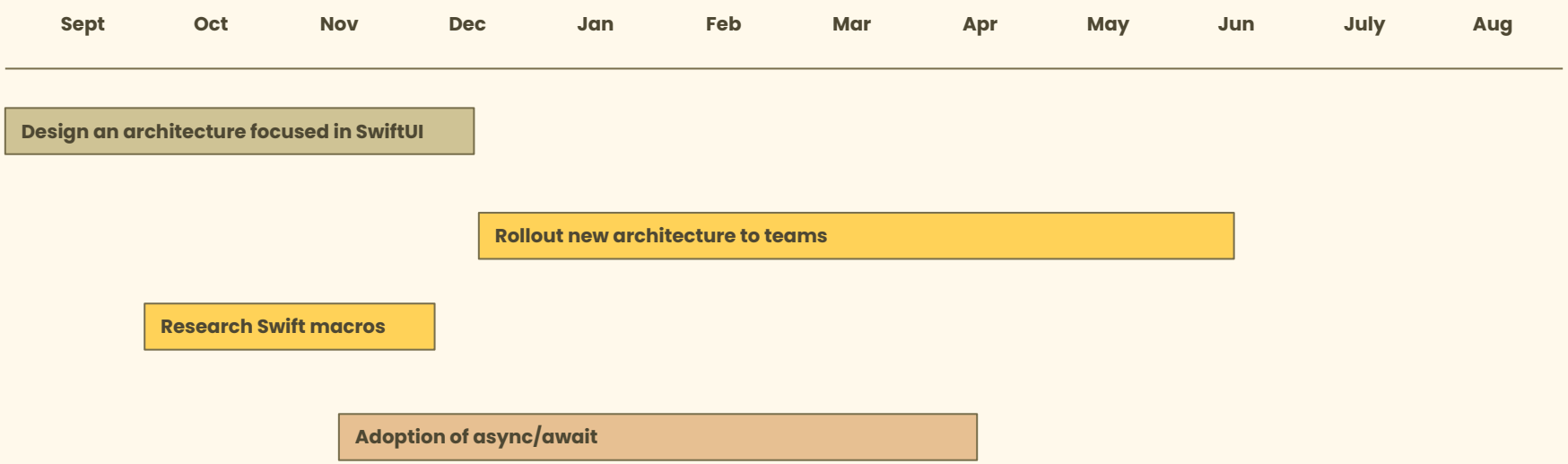
Migration of the design system

SwiftUI by default!

- Make sure the tools and the team are ready before moving by default

What is next?

What is next?


Sept
Oct
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
July
Aug
Design an architecture focused in SwiftUI
Rollout new architecture to teams
Research Swift macros
Adoption of async/await

Back to
the original question

Is SwiftUI the right choice!?

We think it is!!

Thank you.

bumble inc.