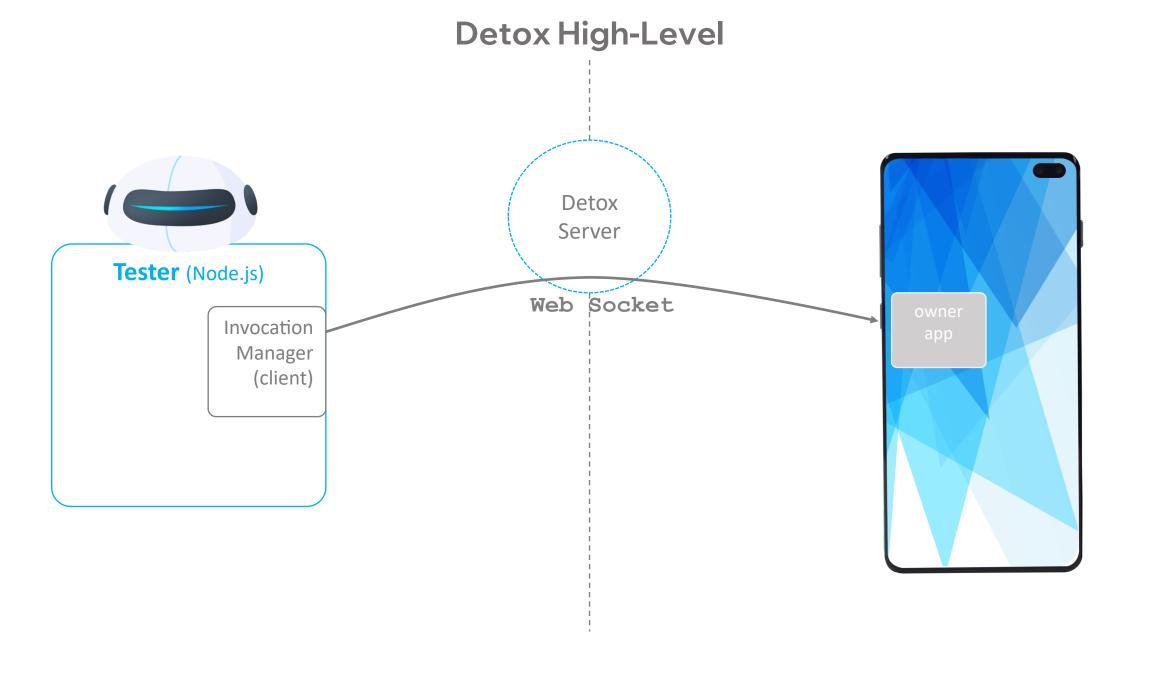
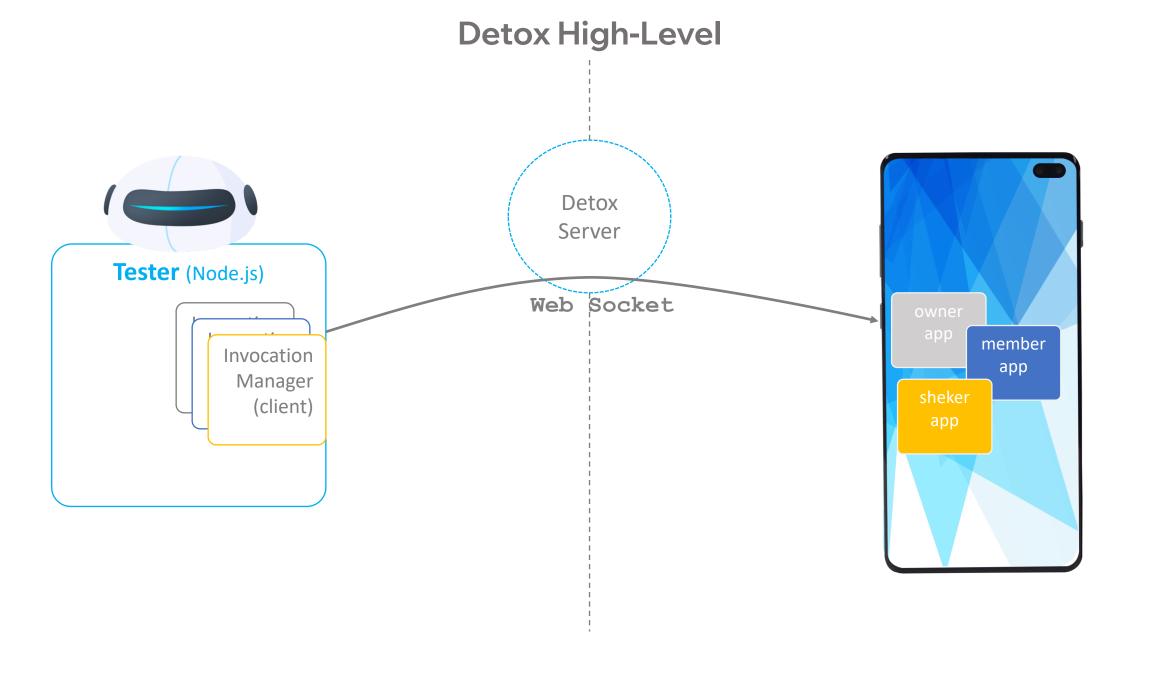
Complexity

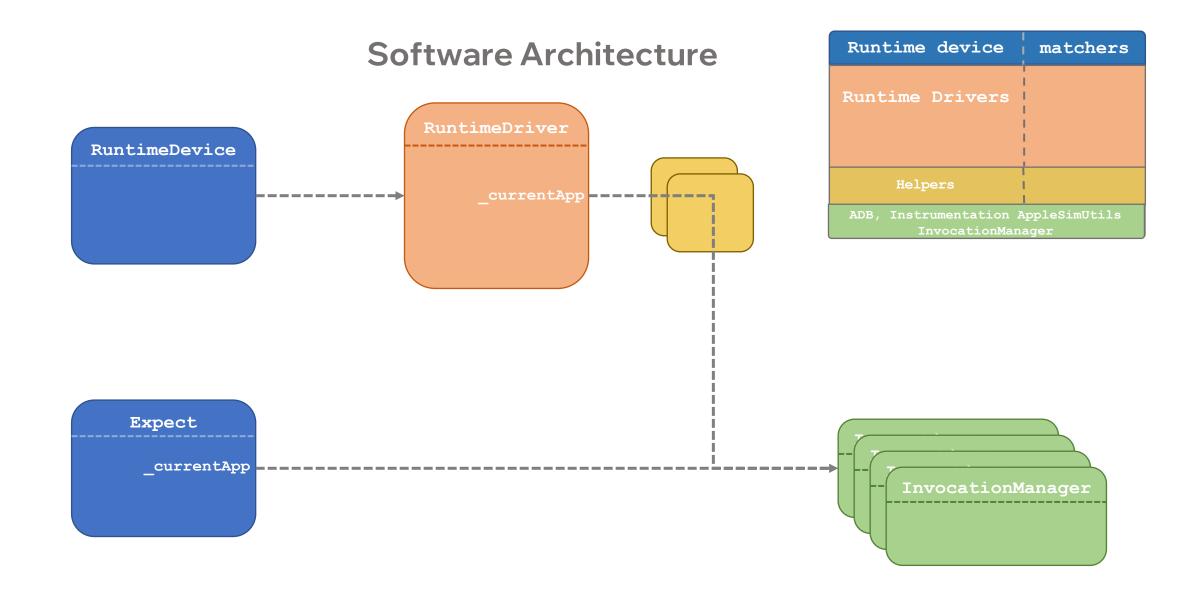
Multi-apps state management





Software Architecture

Runtime device			matchers
Runtime Drivers			
DeviceDriverBase			
Emulator	 Genycloud	 Simulator	
Driver	Driver	Driver	
AppInstallHelper FileXfer 	AppInstallHelper FileXfer		
ADB, Instrumentation		AppleSimUtils	InvocationManager

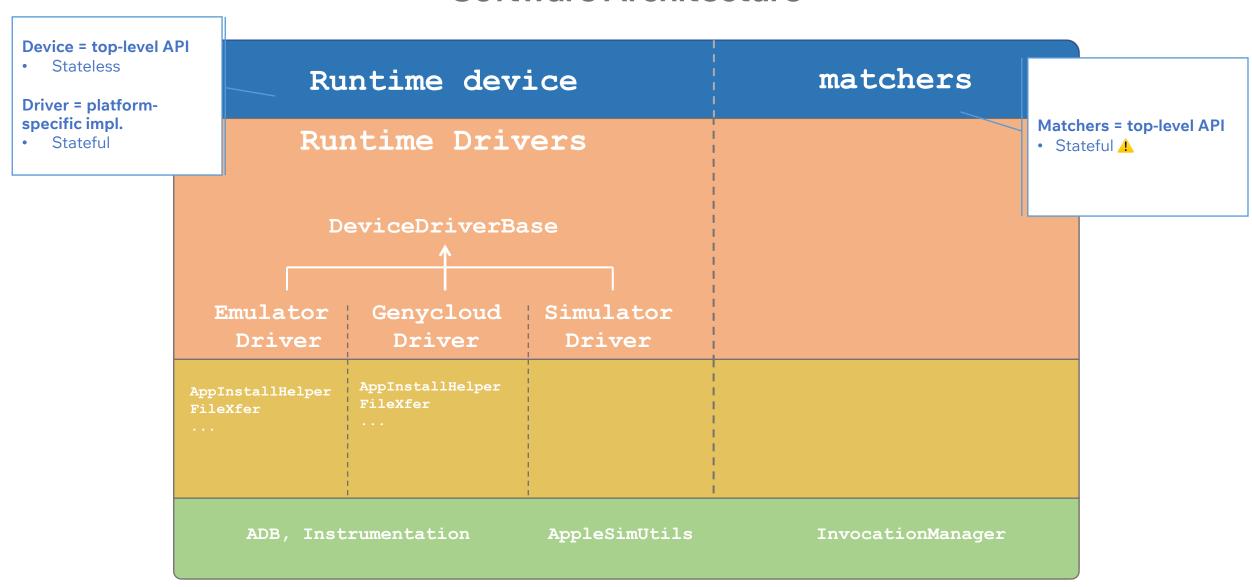


App Toggling

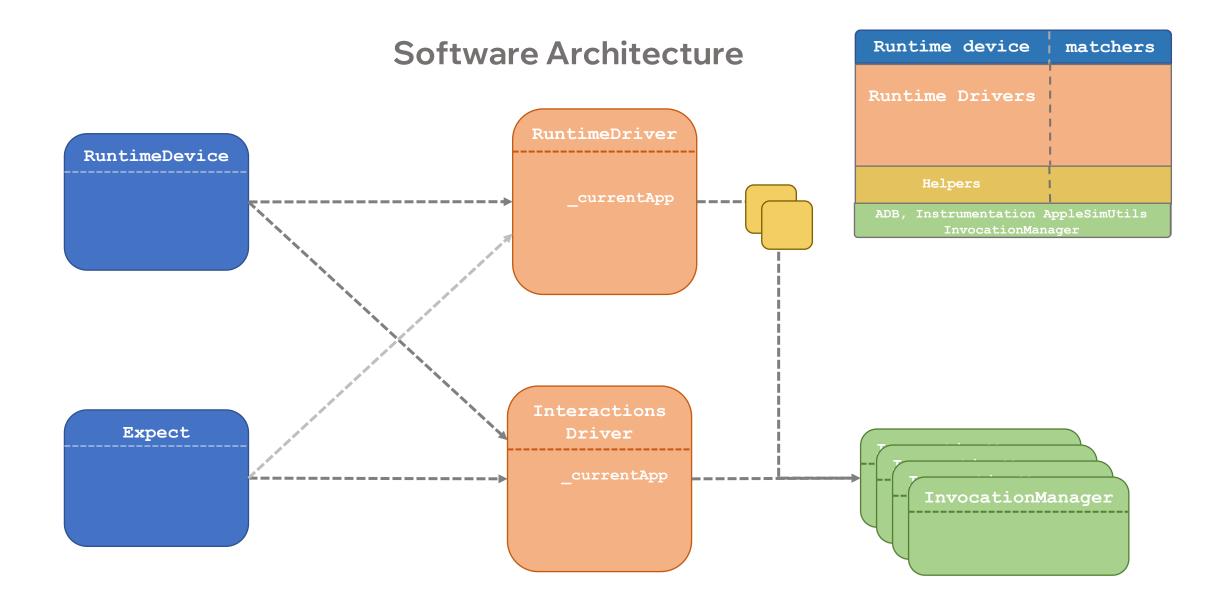
```
selectApp(alias) {
  device.selectApp(alias);
  matchers.selectApp(alias);
```

Solutions

Software Architecture



oftware Architecture element(...).tap() { interactionsDriver.executeTap(matcher, value); matchers levice InteractionsDriver.executeTap(matcher, value) { rivers Interactions Driver invocationManager.execute(interaction); Matchers = top-level API erBase Stateless **Driver = platform-specific** Stateful 1oud Simulator Driver rer device.selectApp(alias) { runtimeDriver.selectApp(alias); interactionsDriver.selectApp(alias); AppleSimUtils ADB, Instrumentation InvocationManager



Profit

- Pushes state down
 - Decouples external API from internal architecture
 - Solves multi-app case
- Matchers: Inter-platform code reuse via inheritance
- Eliminates existing matchers > device relation

Other Solutions

- Connections Abstraction
 - selectApp() → connections.selectActive()
- Add App to main objects model
 - Semantic
- Flux-like
 - Centrally manage apps as a state
 - Entities to observe mutations