

## Comparison of Threading Async Models

## RxPatterns Overview

## RxPatterns Design Methodology

## RxDevice and RxFabric

## More Information

[RxPatterns Whitepaper](#)

[RxPatterns iOS Eval Kit](#)

## Welcome to the talk

This talk centres on asynchronous software design, primarily for the unix based platforms: OS/X, iOS and Linux.

The RxPatterns design methodology is introduced to visualise async operation and so design-for micro and macro async behaviour.

Source examples are in Swift 2.2

## Why be concerned about async design?

Public expectation on software is increasing:

- App software is being asked to interact with more services, (other) apps and devices (both internal and external).
- Content is the king and that content (online services) needs to be accessed and managed asynchronously.
- We are on the cusp of the co-intercation age where software needs to co-integrate with what is available around it (e.g. IOT building, appliance and car devices and services). [My kettle has detected a fault, it knows it is still in warranty and it wants inform the store for replacement but it needs my authorisation].
- Async data may have inter-dependencies and may arrive in any temporal order.

Apps have tradionally operated in the open range but in the future they may be asked to be more responsible and more visible:

- Support discoverability and interoperability with other apps.
- Be responsible for the resources they use, to internally schedule/throttle to conform to energy-resource usage profiles.
- Support service redundancy (and be more generic in their service interaction).

This is avalanching software complexity!