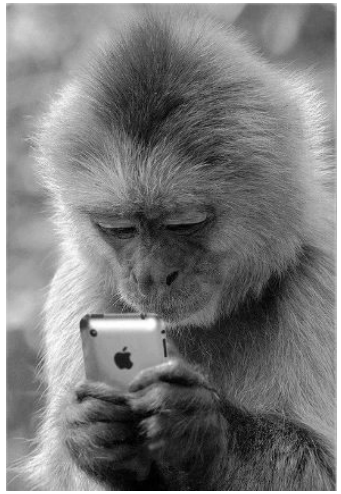
SwiftMonkey



Framework for generating randomised user input in iOS Apps

lyst

Nihal Alfred

Monkey Testing?

- A software testing **technique**
- User tests the application by providing **random inputs** or **trys to crash** the application.
- This technique **does not follow any predefined** test cases or strategy
- Mostly works on **tester's mood and gut feeling**.
- Works very well when doing **load/stress testing** when you try to break your application by proving non-stop random inputs.

Monkey testing is all about "do what you want; automatically".

Monkey Vs Horse

- *Bridle :*
  - Used to **direct & control** the horse
  - Helps the horse to keep **focus and concentrate**
- *Horse in testing:*
  - **Directed & driven** by test cases/plans & strategies
  - Controlled by the **quality metrics**
  - Don’t want to **divert our focus**
  - Strictly **concentrate** on the set of test cases & **obediently** execute them.

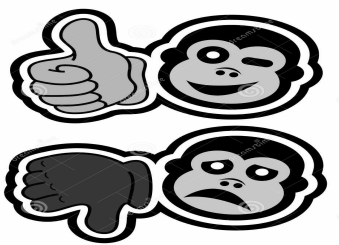
It’s perfectly fine to be a horse, but sometimes don’t you enjoy being a Monkey?

Why?

- In UI testing, it's **easy** to think about **how to test how things *should* work**, but we **struggle** to figure out **what kind of things might *not* work?**
- Ever showed your app to someone **who proceeded to bang away at the screen** & immediately **crashed it by doing something** you had **never thought of?**
- Do you want to feel a bit **more confident** about your **app's stability?**
- Do you have **rare crashes** that you just **can't reproduce?**
- Do you have **memory leaks** that take a **long time to manifest** themselves, and **require lots of UI actions?**

Randomised testing will help you with all of these!

Advantages & Disadvantages



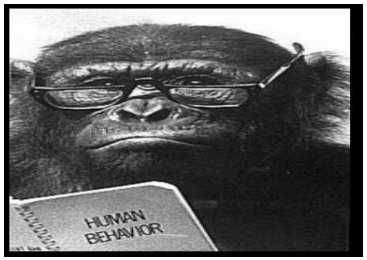
- ✓ Can **identify bugs** which might have a **higher impact**
- ✓ Can **identify** some **out of box errors**
- ✓ **Easy** to **set up and execute**
- ✓ Helps **test** the **reliability of the software**
- ✓ **Not costly**
- x Number of **bugs are less**
- x Can **go on for days** till a bug is not discovered
- x **Reproducing** the bugs (if occurs) becomes **a challenge**
- x Some **“not expected”** output scenarios are **difficult to analyse** and is **time consuming.**

Conclusion

- Though **chaotic**, recommended to **plan & assign time** for it
- Can **discover** some really good bugs: **memory leaks** or **hardware crashing**
- Normally we **ignore many cases** thinking that **“this scenario”** will **never happen**, however **if it happens**, can **lead to a serious impact** (low priority & high severity bug)

“Horse” & “Monkey” testing together can contribute to achieve more quality & confidence in the software.

High level overview



- Inspired by & has similar goals to [UI AutoMonkey]
- Integrated into the Xcode UI testing framework, providing better opportunities to debug.
- Add **`SwiftMonkey.framework`** to your UI test target.
- Then add a test that creates a **`Monkey`** object and uses it to generate events.
- Optionally, you also add the **`SwiftMonkeyPaws.framework`** to your main app, and create a **`MonkeyPaws`** object to enable visualisation. (This greatly increases the usefulness of your randomised testing, as you can see what touches caused any crash you may encounter)

Demo