Agile iOS Development

Illya Mutschnik und Simon Hartmann

Xcode

Continuous Integration

Swift

Scrum

AppCode

Agile iOS Development

Objective-C

Testing

Code Metriken

TDD

Clean Code

Nutzer Feedback

Agile iOS Entwicklung

1. Der App-Markt
2. Warum sollten wir agil entwickeln?
3. Die Entwicklungsumgebung und die Sprache
4. Code Metriken und Continuous Integration
5. Die Teststrategie
6. Kundeneinbeziehung
7. User Experience: Developer und Designer
8. Collective Code Ownership und Frameworks
9. Zusammenfassung

Der App-Markt oder „There's an app for that!“

Handy-Markt: Kein Ende des Wachstums

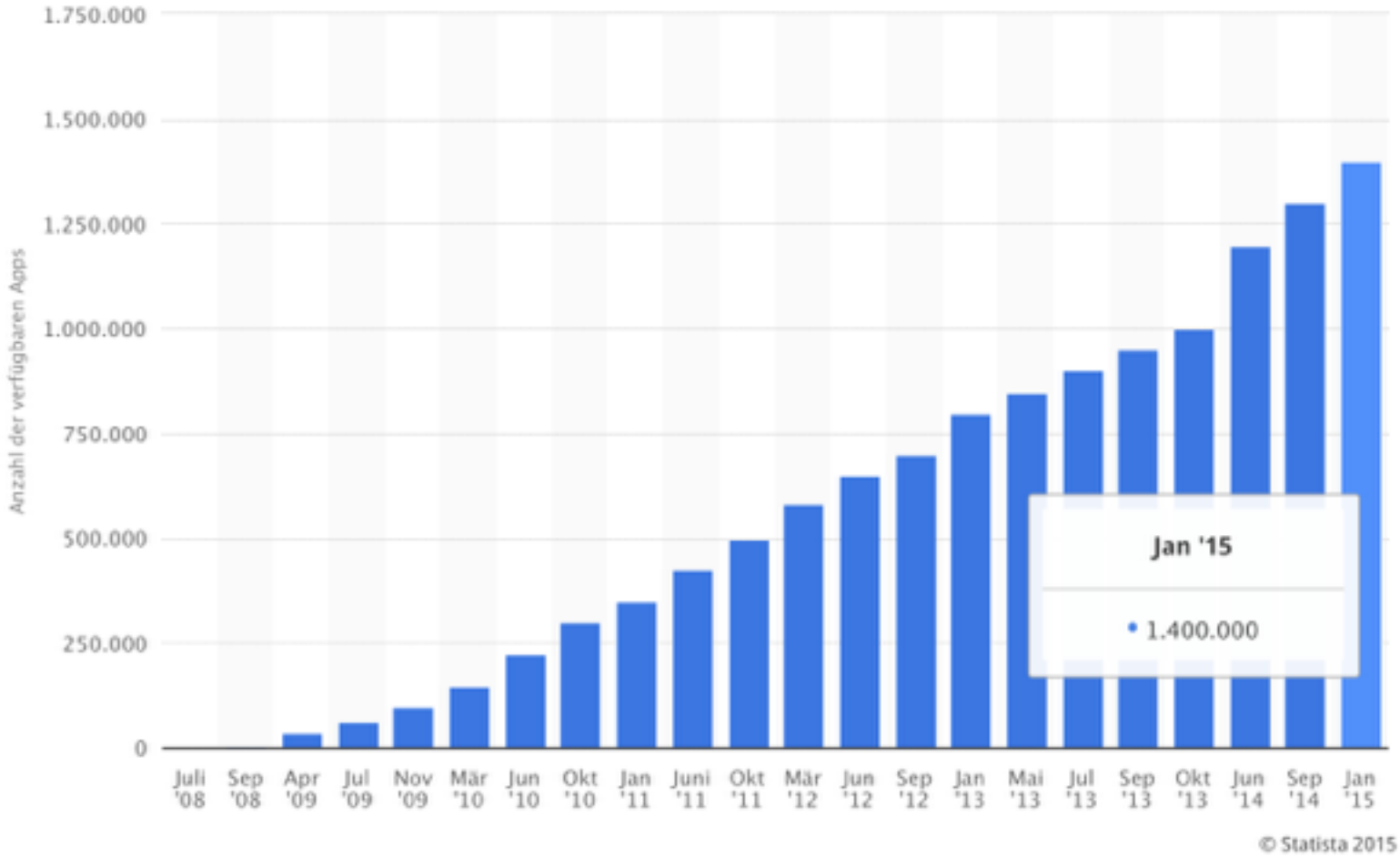
Worldwide Smartphone Sales to End Users by Operating System in 1Q15 (Thousands of Units)

Operating System	1Q15 Units	1Q15 Market Share (%)	1Q14 Units	1Q14 Market Share (%)
Android	265,012	78.9	227,549	80.8
iOS	60,177	17.9	43,062	15.3
Windows	8,271	2.5	7,580	2.7
Blackberry	1,325	0.4	1,714	0.6
Other OS	1,268.7	0.4	1,731.0	0.6
Total	**336,054.4**	**100.0**	**281,636.9**	**100.0**

Source: Gartner (May 2015)

+ 19,3%

App-Markt: Kein Ende des Wachstums



Wie lange dauert die Entwicklung?

http://www.kinvey.com/blog/2086/how-long-does-it-take-to-build-a-mobile-app

Also: Hauptsache Geschwindigkeit? Nein!

"I've seen very talented teams crank out high-quality apps in just a few weeks. However, **the demand for higher production quality in apps has certainly risen** in recent years. Accordingly, **app dev cycles have extended** and we're seeing folks spend anywhere **from 6 to 12 months** on more complex projects."

Quelle: http://readwrite.com/2013/01/09/how-long-does-it-take-to-build-a-native-mobile-app-infographic

Stellenausschreibung eines großen App-Hersteller:

Skills & Requirements

[...]

Testing - Honestly, our code coverage is a little lacking. We'd love your help to change this.

[...]

Warum sollten wir agil entwickeln?

Eine alte Weisheit

If you want to go fast, go alone.

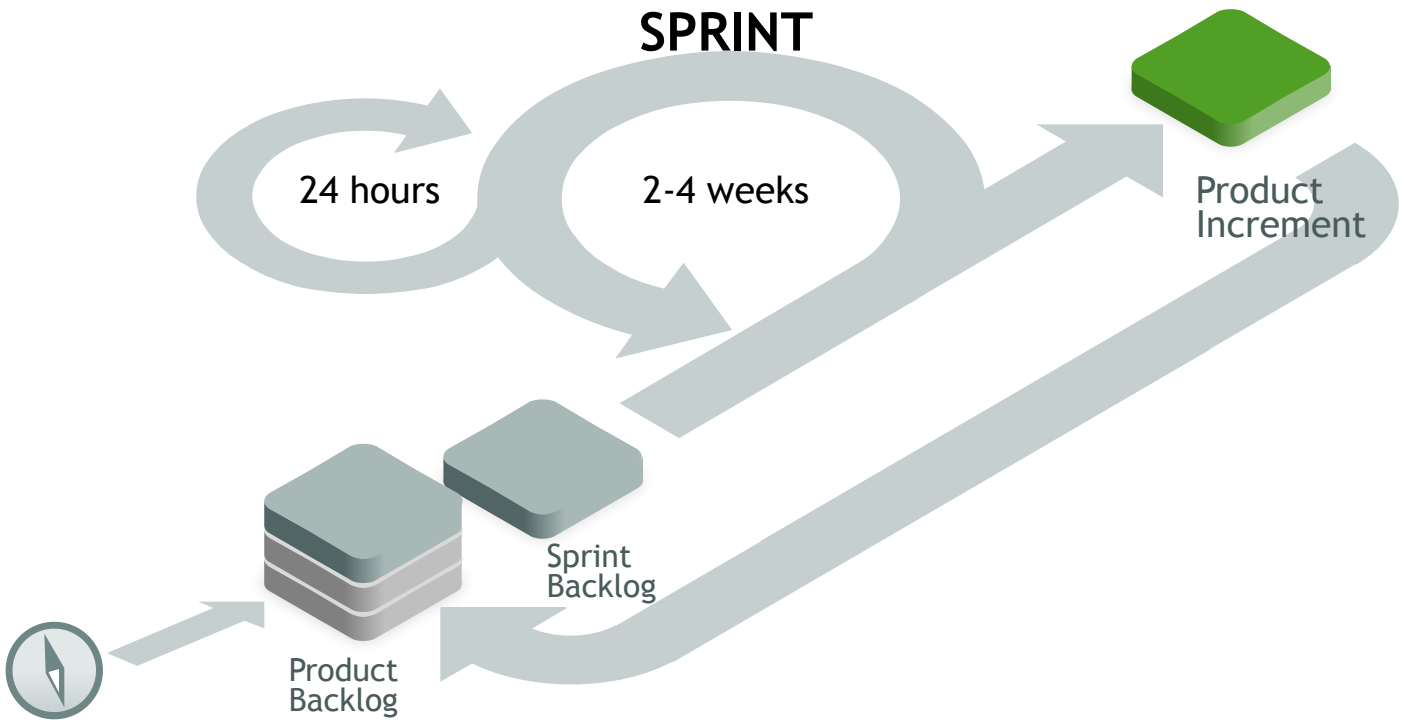
If you want to go far, go together.

Was sind die Besonderheiten der iOS Entwicklung?

- Sehr schnelllebiger Markt
  - Technologien
  - Geräte
  - Nutzerverhalten
  - Design
- hoher Anspruch an Qualität und Design
- Kurzer Weg zum Nutzer, Nutzer erwartet schnelle Updates bei Fehlern

- App Store als einziger Verkaufskanal
  - wesentlich höherer Konkurrenzkampf als bei konventioneller Software

Der Scrum Flow



Wie kann agile Entwicklung helfen?

- Time to market ist kürzer
  - wenige bis keine überflüssigen Funktionen
    - Beta-Testing/Analytics
    - App muss nicht gleich gesamten Umfang haben (1.0,1.1, …)
- mit XP/Clean Code: höhere Qualität durch
  - schnelles Feedback - von Nutzern und Auftraggeber
  - Crash-Analysen
  - schnelle Korrekturen

Ok, und wie machen wir es richtig?

Agil - Scrum

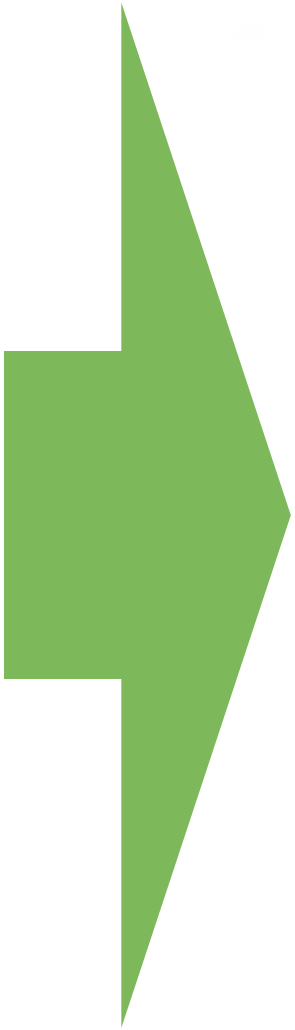
kurze Entwicklungszyklen
fast feedback
enger Kundenkontakt
iterative Planung
…

Clean Code

Design Patterns
Code Metriken
Refactoring
…

Extreme Programming

Continuous Integration
Tests
richtige Frameworks und Tools
…

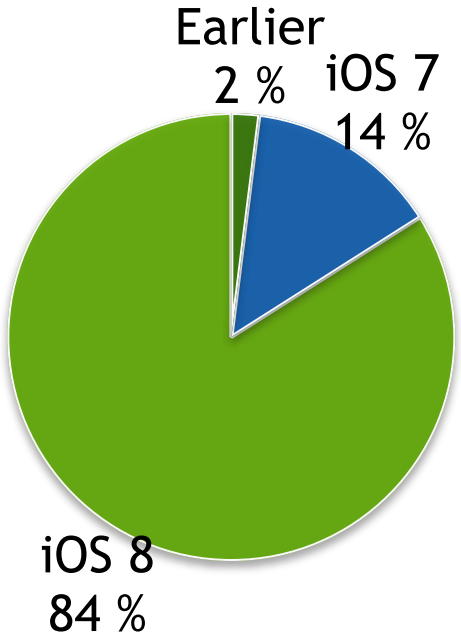


Qualität

Die Entwicklungsumgebung und die Sprache

Die Ausgangslage

- iOS Version: 8.4 (seit 1. Juli 2015)
- Sprachen: Objective-C, Swift (wird Open Source)



Verteilung von iOS Versionen
(Stand 22.06.2015)

Swift

+	-
Funktional	kein Refactoring Support
Typisiert	Noch: weniger Community Support
Einfachere Lesbarkeit (bye bye [[…]])	Noch: Aufwand mit alten Frameworks
Sicherer: keine nil-no-ops	(fast) keine Code Metriken
tuples, Closures	
keine header-Dateien	
wird für viele Plattformen verfügbar	

Die Entwicklungsumgebung

AppCode

Xcode

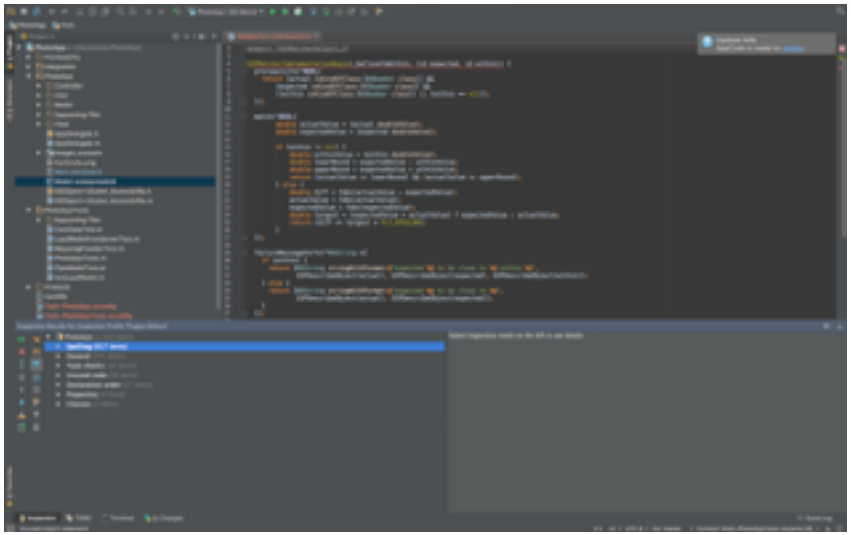
+	-
Interface Builder integriert	quasi kein Refactoring möglich
CoreData Modellverwaltung	unkomfortables Debugging
Erweiterungen installierbar	UI nur minimal anpassbar
UI Debugging	keine auto imports
Performance Tests	fast keine quick fixes
Instruments	
Playgrounds (Swift)	

AppCode (JetBrains)

+	-
Sehr gutes Refactoring (ObjC)	kein kompletter Workflow
UI anpassbar	noch: Kein Refactoring von Swift Code
Plugins	
auto imports	
Quickfixes	
„Code inspections“	

IDE: Die Lösung

+



Xcode

Konfiguration
Interface Builder
CoreData

AppCode

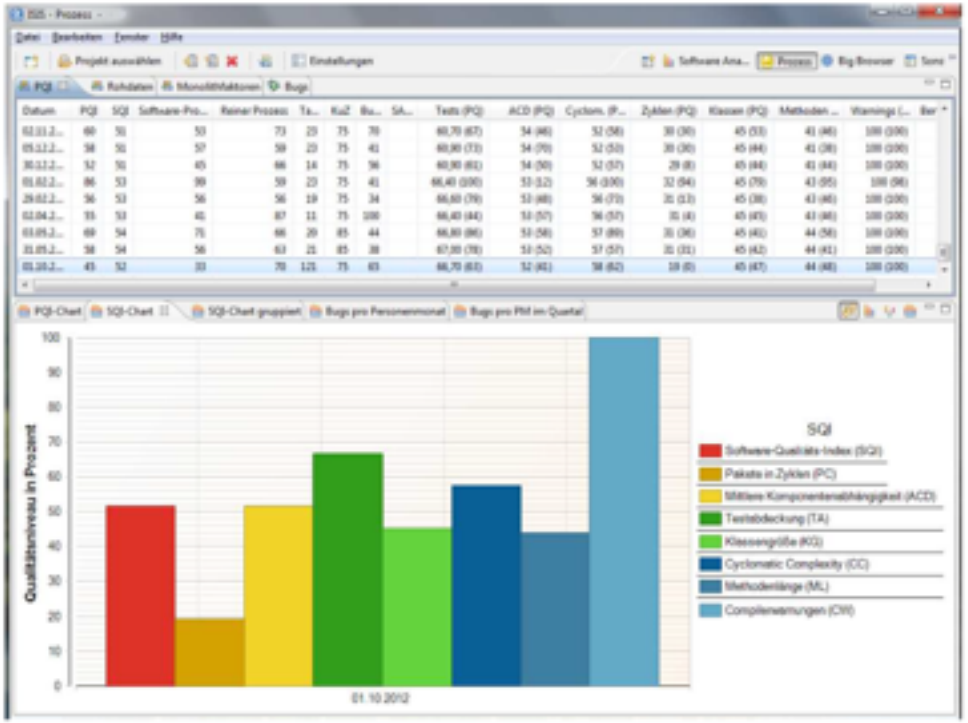
Development
Refactoring
Debugging

Code Metriken und Continuous Integration

Wie messen wir Softwarequalität?

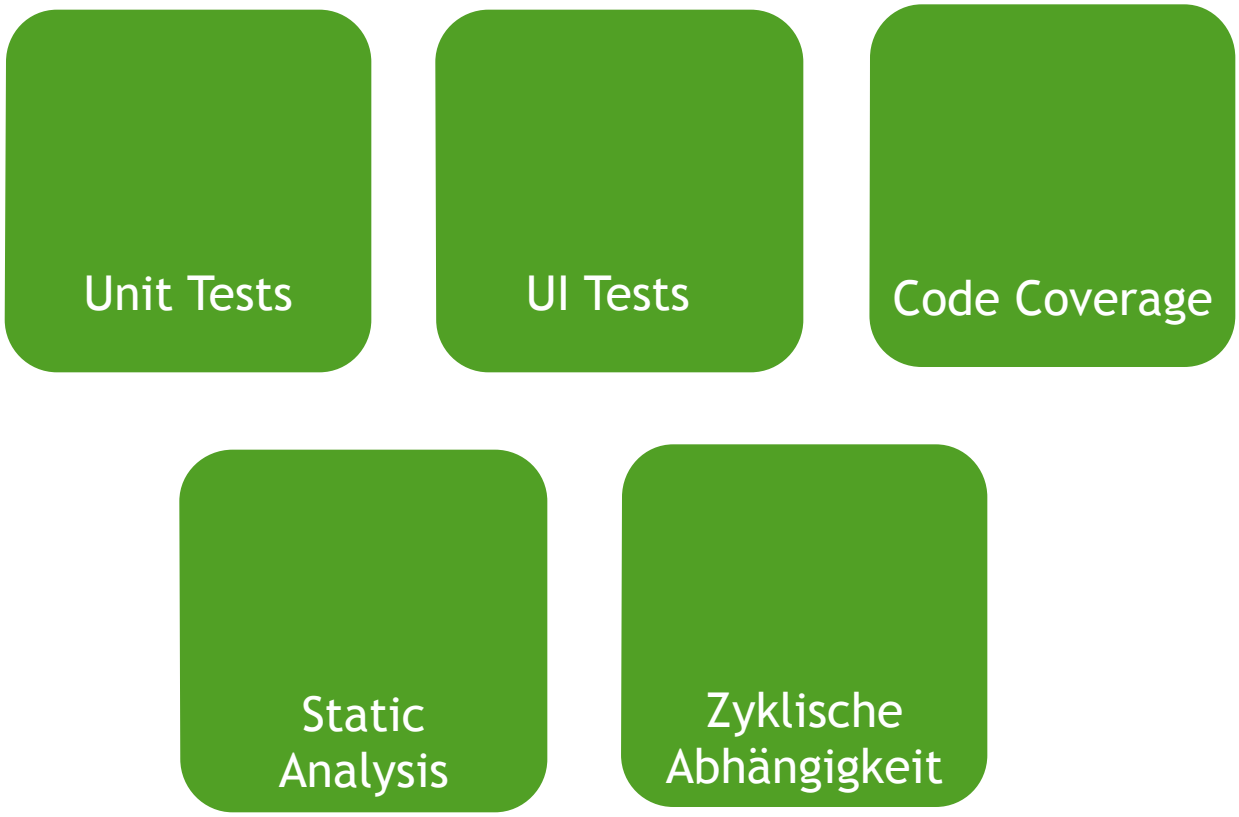
andrena objects: Messung der Qualität mit SQI

Aus den **Indikatormetriken** wird der Softwarequalitätsindex (SQI) berechnet



https://www.andrena.de/code-assessment

Continuous Integration: Anforderungen



Continuous Integration: Jenkins

Vorteile:

1. Viele Plugins
2. Kostenlos
3. Große Community

https://wiki.jenkins-ci.org/display/JENKINS/ChuckNorris+Plugin

Continuous Integration: Unit Tests

Testergebnis : (root)

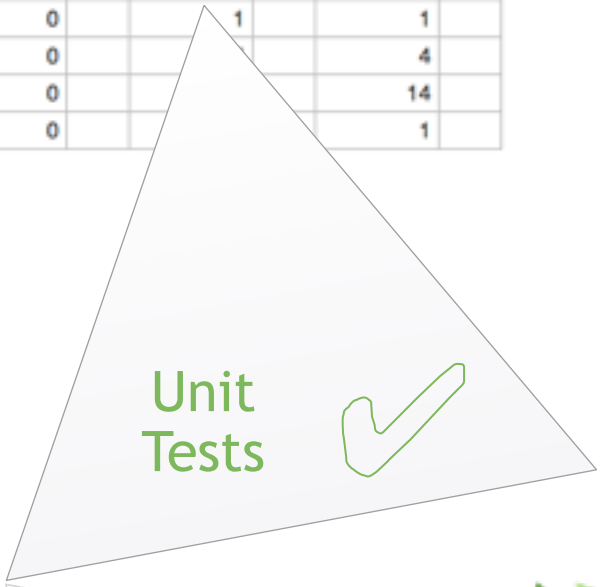
Fehlschläge (±0)

Tests (±0)
Dauer: 1,4 Sekunden
Beschreibung hinzufügen

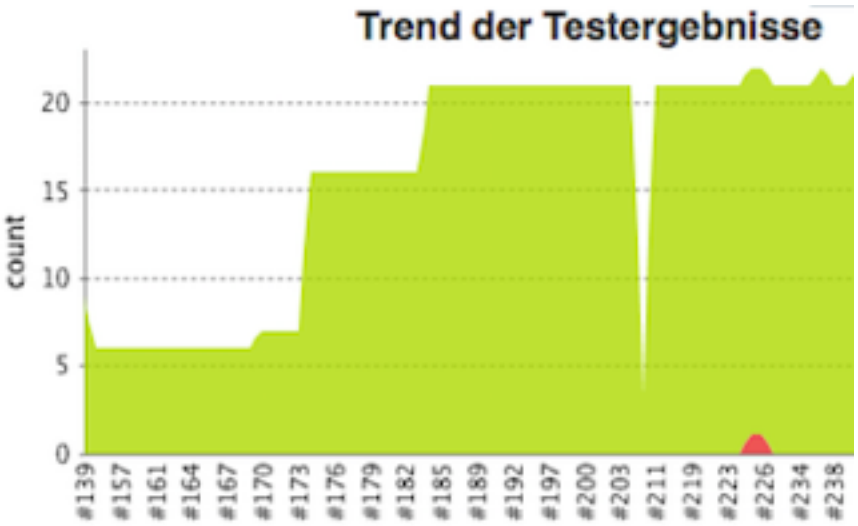
Alle Tests

Klasse	Dauer	Fehlgeschlagen	(Diff.)	Übersprungen	(Diff.)	Pass	(Diff.)	Summe	(Diff.)
CoreDataTest	0,28 Sekunden	0		0		2		2	
LoadModelFromServerTest	0 ms	0		0		1		1	
PhotoAppTests	0,13 Sekunden	0		0				4	
PipeModelTest	0 ms	0		0				14	
UIAutomation	1 Sekunde	0		0				1	

Tools: xcodebuild, ocunit2junit



Testergebnisse



Continuous Integration: Code Coverage

Testabdeckung

Cobertura Testabdeckung

Trend

Zusammenfassung der Testabdeckung nach Project

Name	Packages	Dateien	Klassen	Zeilen	Verzweigungen
Cobertura Testabdeckung	100% 1/1	57% 16/28	57% 16/28	45% 288/637	39% 81/207

Aufschlüsselung der Testabdeckung nach Package

Name	Dateien	Klassen	Zeilen	Verzweigungen
<default>	57% 16/28	57% 16/28	45% 288/637	39% 81/207

Tools: xcodebuild, gcovr

Code Coverage ✓

Continuous Integration: Static Analysis

PMD Ergebnis

Vergleich mit letzter Analyse

Alle Warnungen	Neue Warnungen	Behobene Warnungen
196	0	0

Zusammenfassung

Gesamt	Hohe Priorität	Normale Priorität	Niedrige Priorität
196	0	23	173

Details

Verzeichnisse | Dateien | Typen | Warnungen | Details | Normal | Niedrig

Quellverzeichnis	Gesamt	Verteilung
PhotoApp	163	
PhotoAppTests	33	
Gesamt	196	

Tools: OCLint, Clang, hfcca

Continuous Integration: Static Analysis

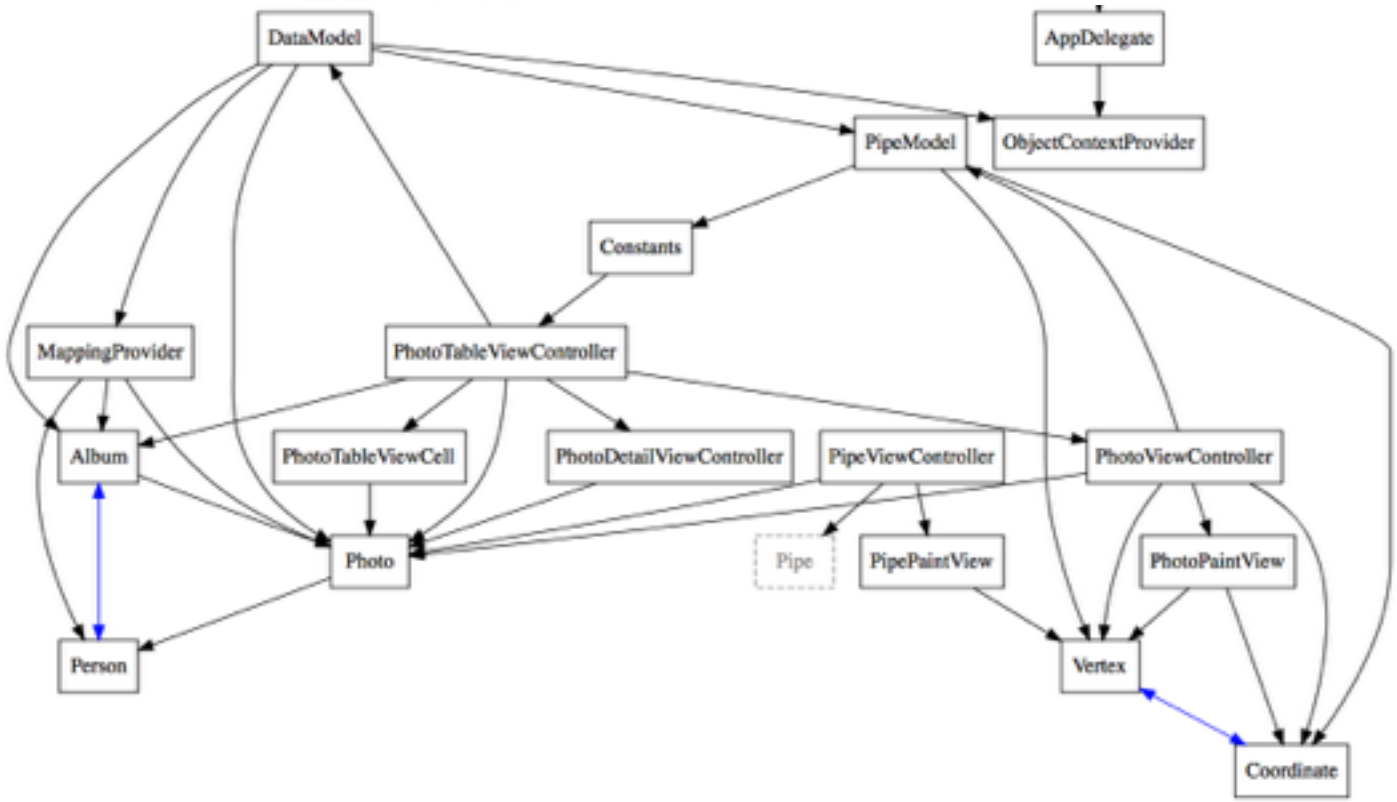
Details

Verzeichnisse | Dateien | **Typen** | Warnungen | Details | Normal | Niedrig

Typ	Gesamt	Verteilung
collapsible if statements	2	
empty if statement	4	
high cyclomatic complexity	2	
high ncss method	3	
ivar assignment outside accessors or init	14	
long line	118	
long method	2	
long variable name	24	
redundant if statement	1	
replace with container literal	1	
short variable name	14	
switch statements should have default	2	
too few branches in switch statement	1	
too many methods	1	
unnecessary else statement	1	
unused local variable	1	
unused method parameter	3	
use early exits and continue	1	
useless parentheses	1	
Gesamt	**196**	

Static Analysis ✓

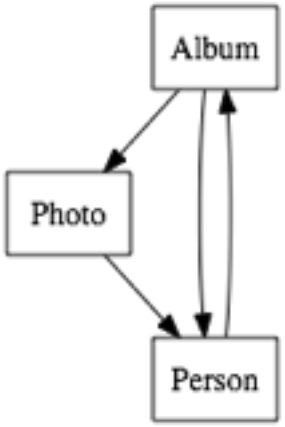
Continuous Integration: Dependency Analysis

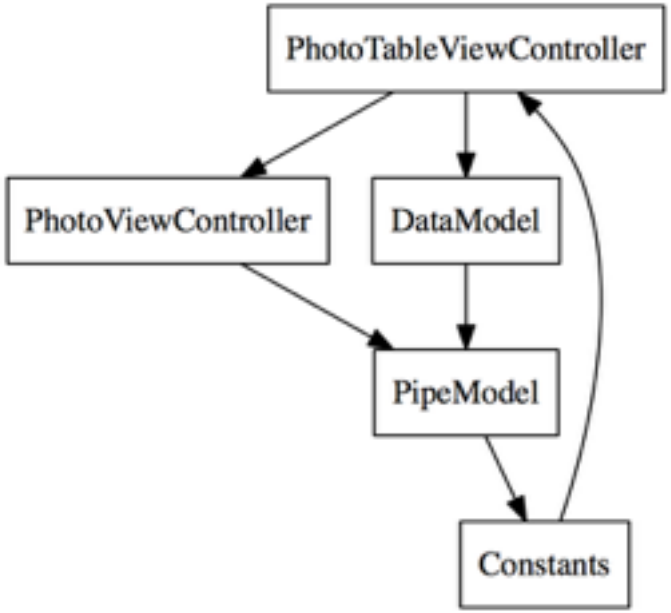


Hmm.. da gibts aber Zyklen!

Link zur pdf-Datei mit allen Klassenzyklen

Continuous Integration: Dependency Analysis

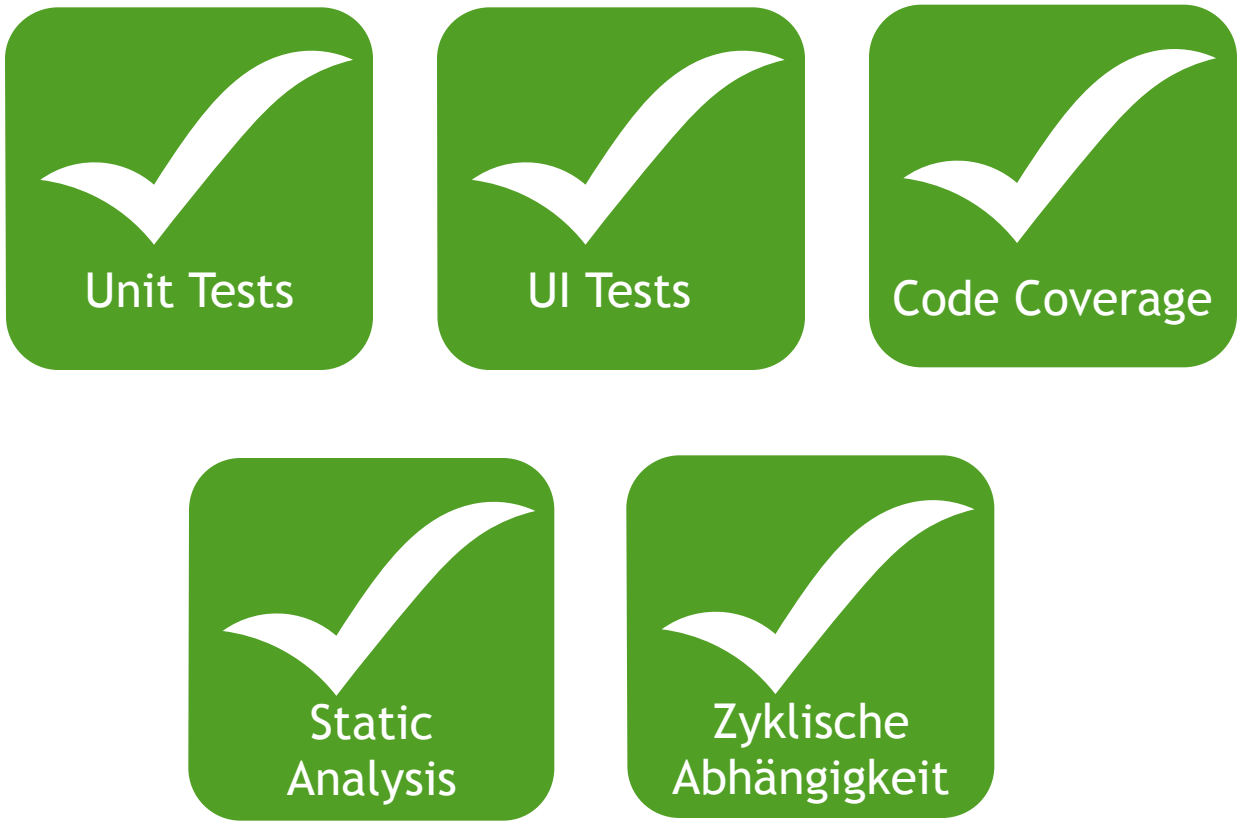




Tools: selbstentwickelt

Zyklische Abhängigkeiten ✓

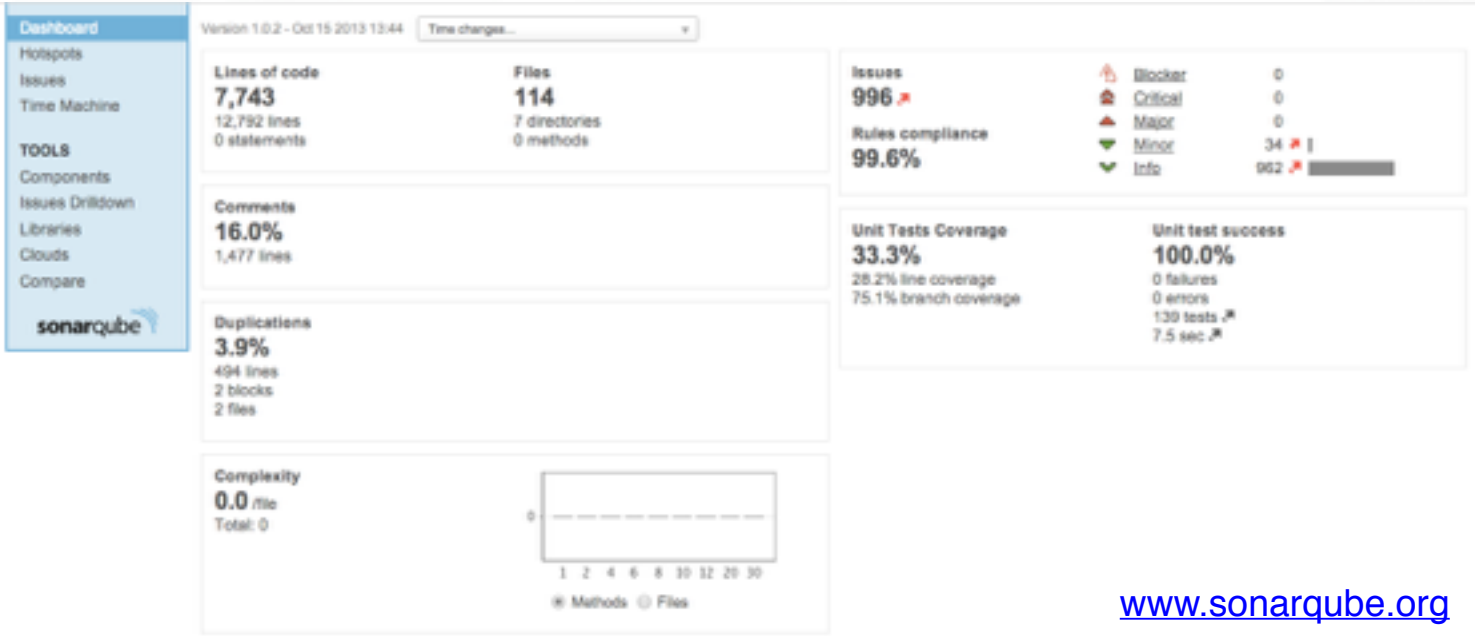
Continuous Integration: Zusammenfassung



Continuous Integration: Was fehlt?

Swift - Unterstützung

SonarQube



www.sonarqube.org

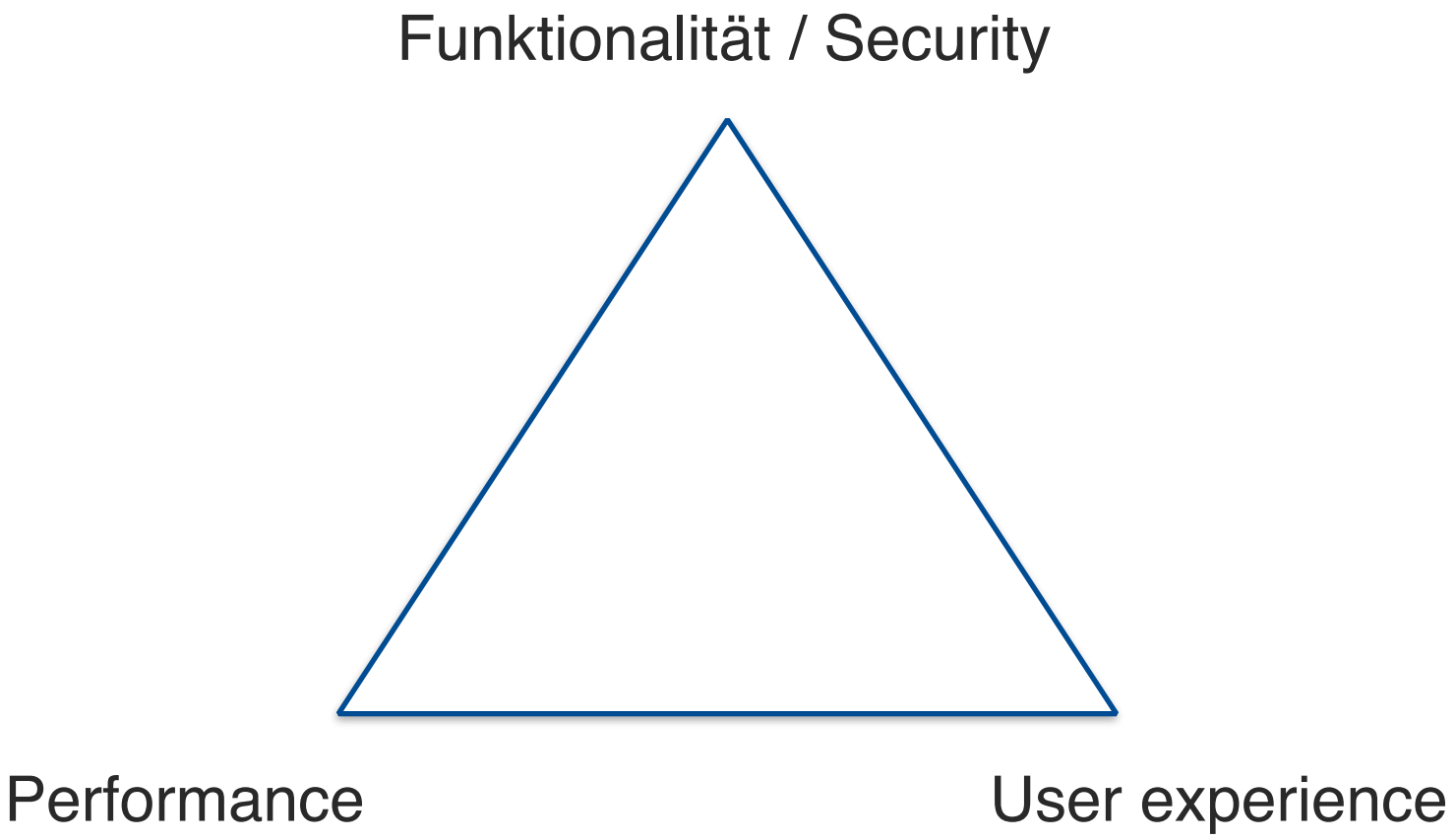
	Objective C	*Swift*
Community Edition	als Opensource Plugin	nicht verfügbar
Kommerzielle Version (ab 5000 EUR/Jahr)	Kommerzielles Plugin (erweiterte Metriken)	Kommerzielles Plugin

Die Teststrategie

Mobile: Der Unterschied

- Unterschiedliche Geräte
- Native/Hybrid/Web-Apps
- Test: Mix aus Simulator und echten Geräten
- User experience sehr wichtig
- Viele Variablen beim Nutzer
  - Netzwerk-Verfügbarkeit
  - Displaygrößen
  - OS-Versionen
  - Hardware-Rechte(!)
  - verfügbarer Speicher
  - …

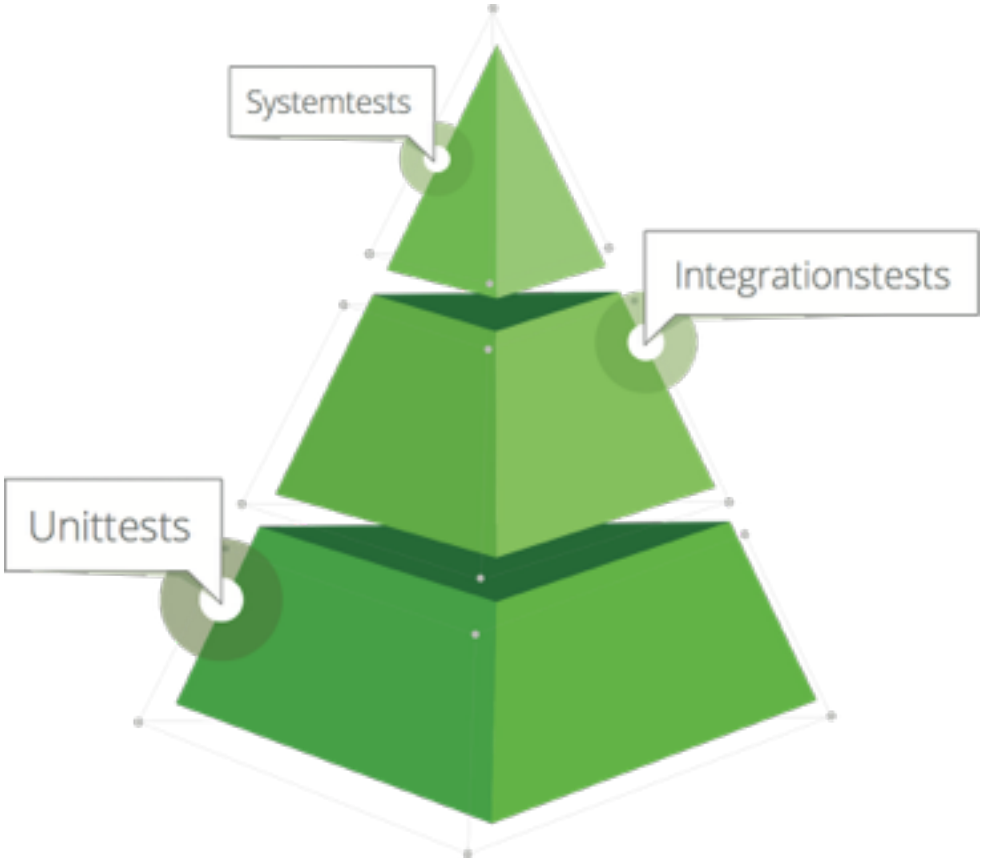
Teststrategie: Die Planung



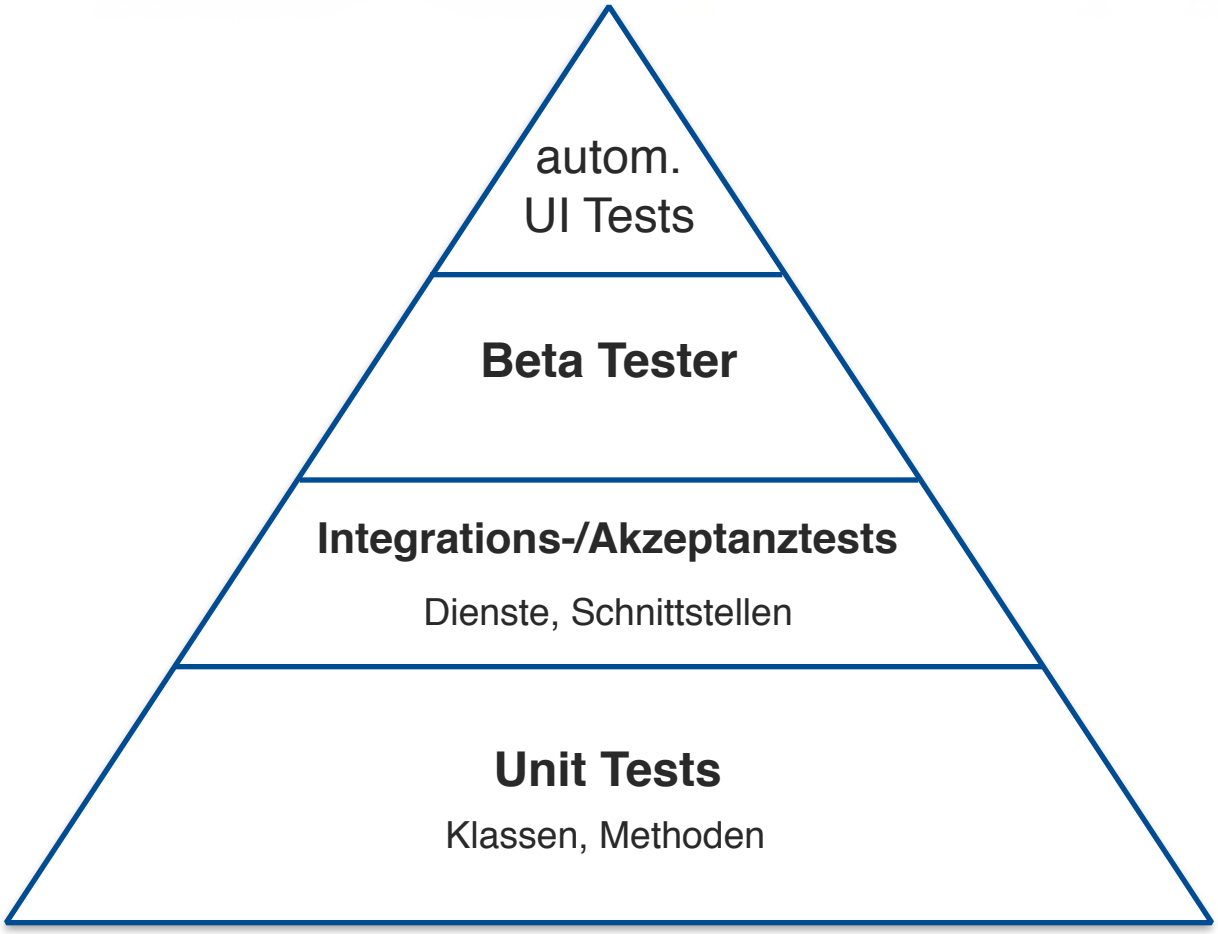
Teststrategie

- Testumgebung stabilisieren
- Testbereiche (UI, Controller, Backend, Netzwerk, …) klar trennen
- Isoliert Testen (Netzwerk)
- Unit Tests sind wichtigste Komponente

Teststrategien: Die klassische Testpyramide

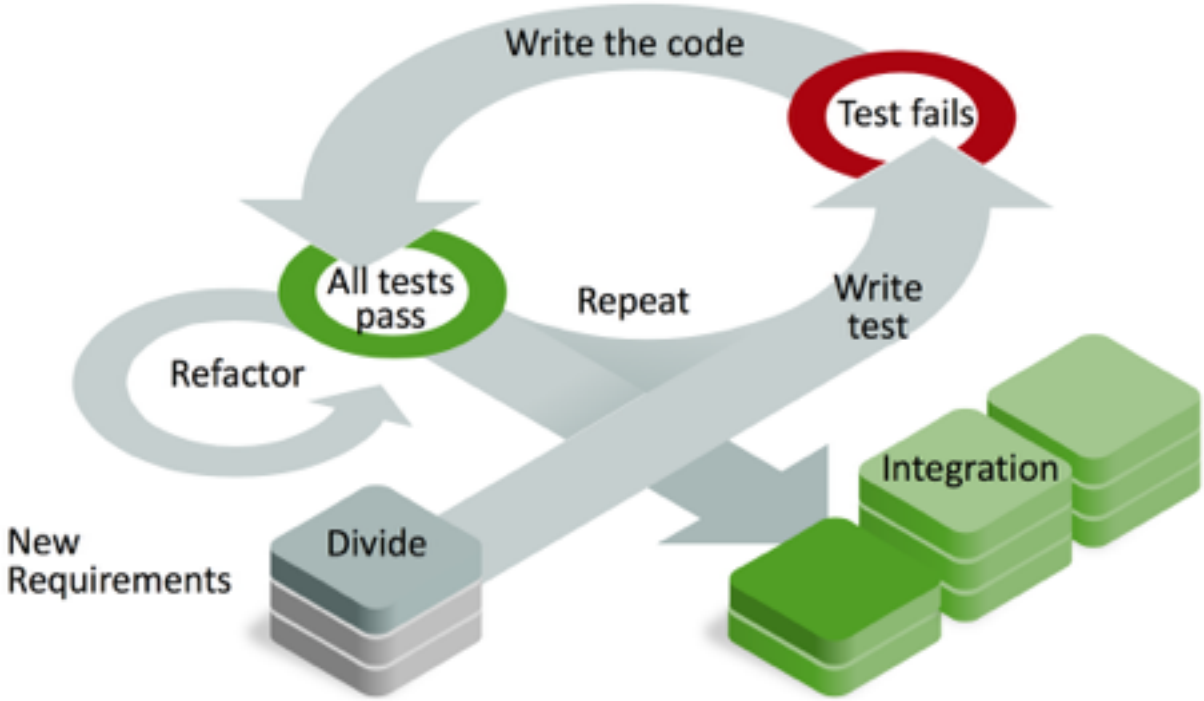


Teststrategie: Die mobile Testpyramide



Unit-Tests

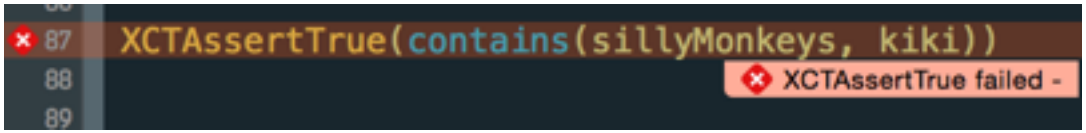
Test Driven Development



- In kurzen Zyklen arbeiten
  - Zyklus: roter Test, grüner Test, Refactoring
- Aufgetretene Fehler mit einem Test absichern

Unit Test: Test Frameworks

- Test-Framework ist empfehlenswert für
  - bessere Lesbarkeit
  - bessere Strukturierung der Tests
  - aussagekräftigere Fehlermeldungen




```
-  XCTAssertTrue(contains(sillyMonkeys, kiki), "Expected
sillyMonkeys to contain 'Kiki'")

// Quick
+  expect(sillyMonkeys).to(contain(kiki))
```

https://raw.githubusercontent.com/Quick/Quick/v0.4.0/Documentation/NimbleAssertions.md

Matcher: OCHamcrest, Kiwi, Expecta, Nimble

```
1 // OCHamcrest
2 assertThat(@"foo", is(equalTo(@"foo")));
3 assertThatUnsignedInteger(foo, isNot(equalToUnsignedInteger(1)));
4 assertThatDouble(baz, is(equalToDouble(3.14159)));
```

```
1 // Kiwi
2 [[@"foo" should] equal:@"foo"];
3 [[foo shouldNot] equal:theValue(1)];
4 [[baz should] equal:theValue(3.14159)];
```

```
1 // Expecta
2 expect(@"foo").to.equal(@"foo"); // `to` ist syntaktischer Zucker
3 expect(foo).notTo.equal(1);
4 expect(baz).to.equal(3.14159);
```

```
1 // Nimble
2 expect(„foo“).to(equal(„foo“))
3 expect(foo).toNot(equal(1))
4 expect(baz).to(equal(3.14159))
```


Unit-Tests: Specta (BDD)

```
describe(@"Thing", ^{

  beforeAll(^{

  });

  beforeEach(^{

  });

  it(@"should do stuff", ^{

   });

  it(@"should do some stuff asynchronously", ^{
    waitUntil(^(DoneCallback done) {
     done();
    });
  });
```


Unit-Tests: Xcode Integration

```
15  @implementation ViewController
16
17  + (void)testedMethod
18  {
19      NSLog(@"%s", __PRETTY_FUNCTION__);
20  }
21
22  + (void)partiallyTestedMethod
23  {
24      NSLog(@"%s", __PRETTY_FUNCTION__);
25      BOOL val = NO;
26      if (val) {
27          NSLog(@"%@", @"Not tested");
28      }
29  }
30
31  @end
32
```

XCoverage: https://github.com/dropbox/XCoverage

Unit-Tests: Kurz und knapp

Frameworks

Specta // Expecta // (OCMock)

Swift: Quick // Nimble // (OCMock)

- Zeit für Tests einplanen
- Grundfunktionalität prüfen
- sollte angemessen hohen Anteil des Codes abdecken

Integrationstests

Integrations-/Akzeptanztests

- eigene API (Golden Request / Golden Response)
  - Funktionalität
  - Performance/ Lastverhalten
  - Sicherheit
- User Experience: Umfrage mit Test-Benutzern
  - Bedienbarkeit
  - Flow durch die App
  - Store-Feedback ist „Test-Ergebnis“

Automatisierte UI-Tests

UI-Tests: Die Alternativen

	Appium	UIAutomation	Xcode 7
Sprache	Java, Objective C, Swift, PHP, node.js, Python	JavaScript	Swift, Objective-C?
Testvoraus-setzungen	keine	keine	keine
Plattformen	iOS, Android	iOS	iOS
CI Build	möglich	möglich	möglich
Inoffizielle APIs	nein	nein	nein
echte Geräte	ja	ja	ja

UI-Tests: Appium

- Sprachen: Java, Objective-C, Swift, JavaScript, Ruby, PHP, C#, …
- Kein SDK oder Recompile notwendig
- iOS und Android werden unterstützt
- Verwendet keine nicht-öffentlichen APIs
- Baut auf UIAutomation von Apple auf
- Black Box Testing

UI-Tests: Die Alternativen

	Appium	UIAutomation	Xcode 7
Sprache	Java, Objective C, Swift, PHP, node.js, Python	JavaScript	Swift, Objective-C?
Testvoraus-setzungen	keine	keine	keine
Plattformen	iOS, Android	iOS	iOS
CI Build	möglich	möglich	möglich
Inoffizielle APIs	nein	nein	nein
echte Geräte	ja	ja	ja

Weitere Alternativen: KIF, Frank

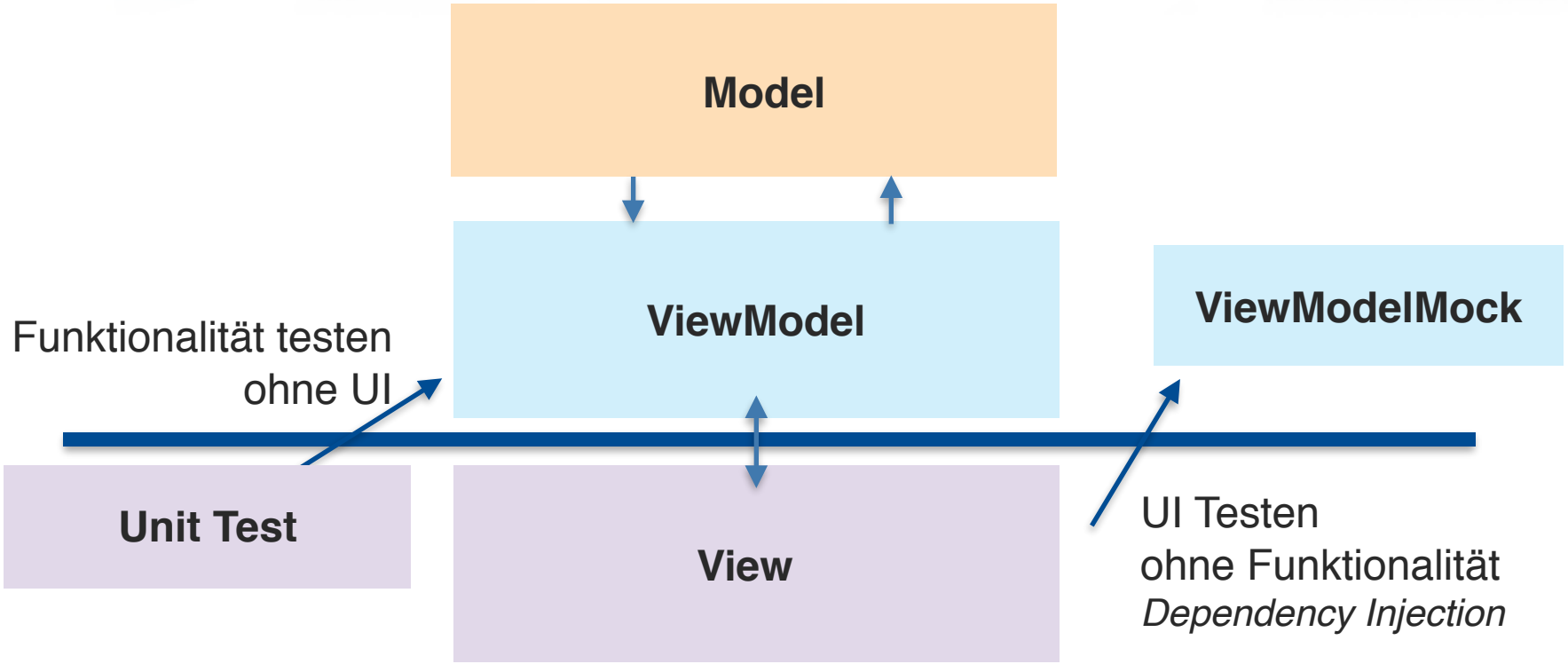
UI-Tests: Es geht auch einfacher

Das Problem:
1. ViewController machen *alles*
2. ViewController sind schwer zu testen

Das Ziel: Trennung von UI und Geschäftslogik und damit bessere Testbarkeit

Die Lösung: Das MVVM Pattern // Dependency Injection

Unit-Tests: Besser mit MVVM (Model - View - ViewModel)

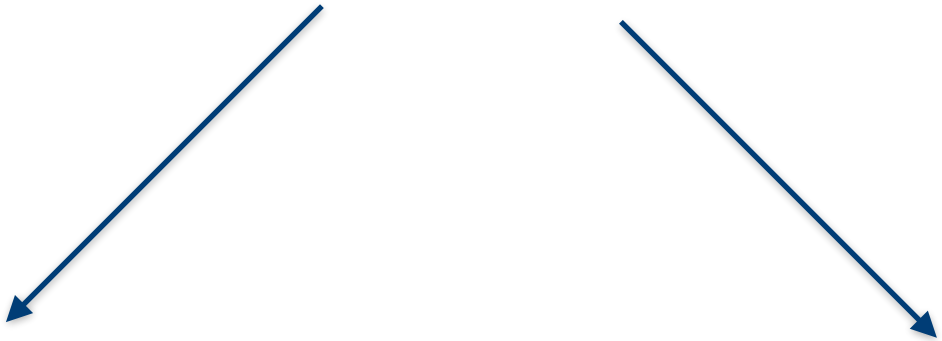


- Funktionalitäten können über Unit-Tests getestet werden
  - Performantes Erzeugen und Ausführen der Tests
  - Isolierte Tests möglich

Kundeneinbeziehung

Kundeneinbeziehung

Wie erhalten wir Kundenfeedback?



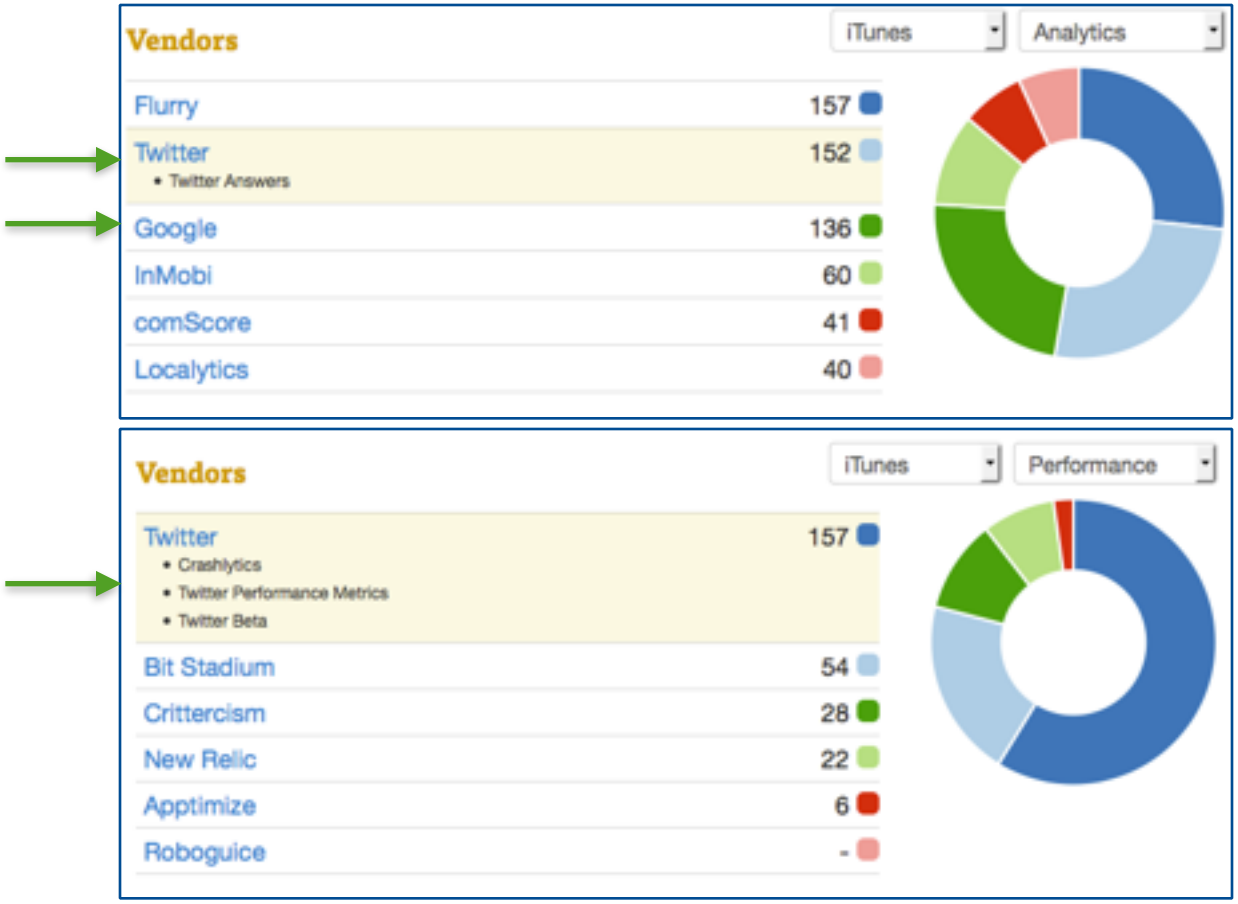
<u>Direktes Feedback</u>
Persönlicher Kontakt mit PO
Tester „live“ beobachten
Feedbackoption in der App

<u>Nutzer-Feedback Frameworks</u>
Beta-Verteilung
Crashreports
Mobile Analytics

Nutzer-Feedback Frameworks: Wunschkriterien

- Crash Reports
- Beta-Verteilung
- Analytics
- Integrierbar in CI
- Android Unterstützung
- Kostengünstig

Ranking Mobile SDKs (May 2015)



https://sourcedna.com/stats/

Kundeneinbeziehung: Unsere Wahl

https://www.crashlytics.com/

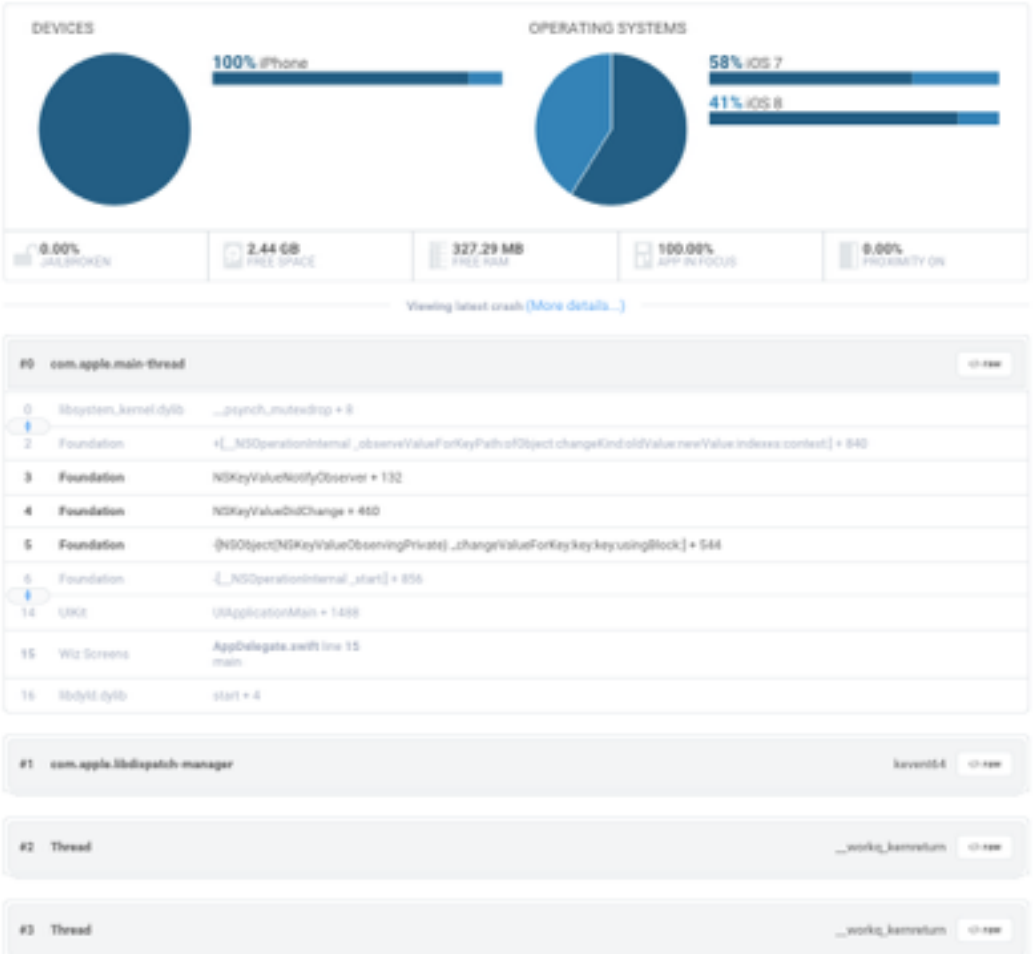
Google Analytics

www.google.com/analytics/mobile/

Crashlytics: Desktop integration

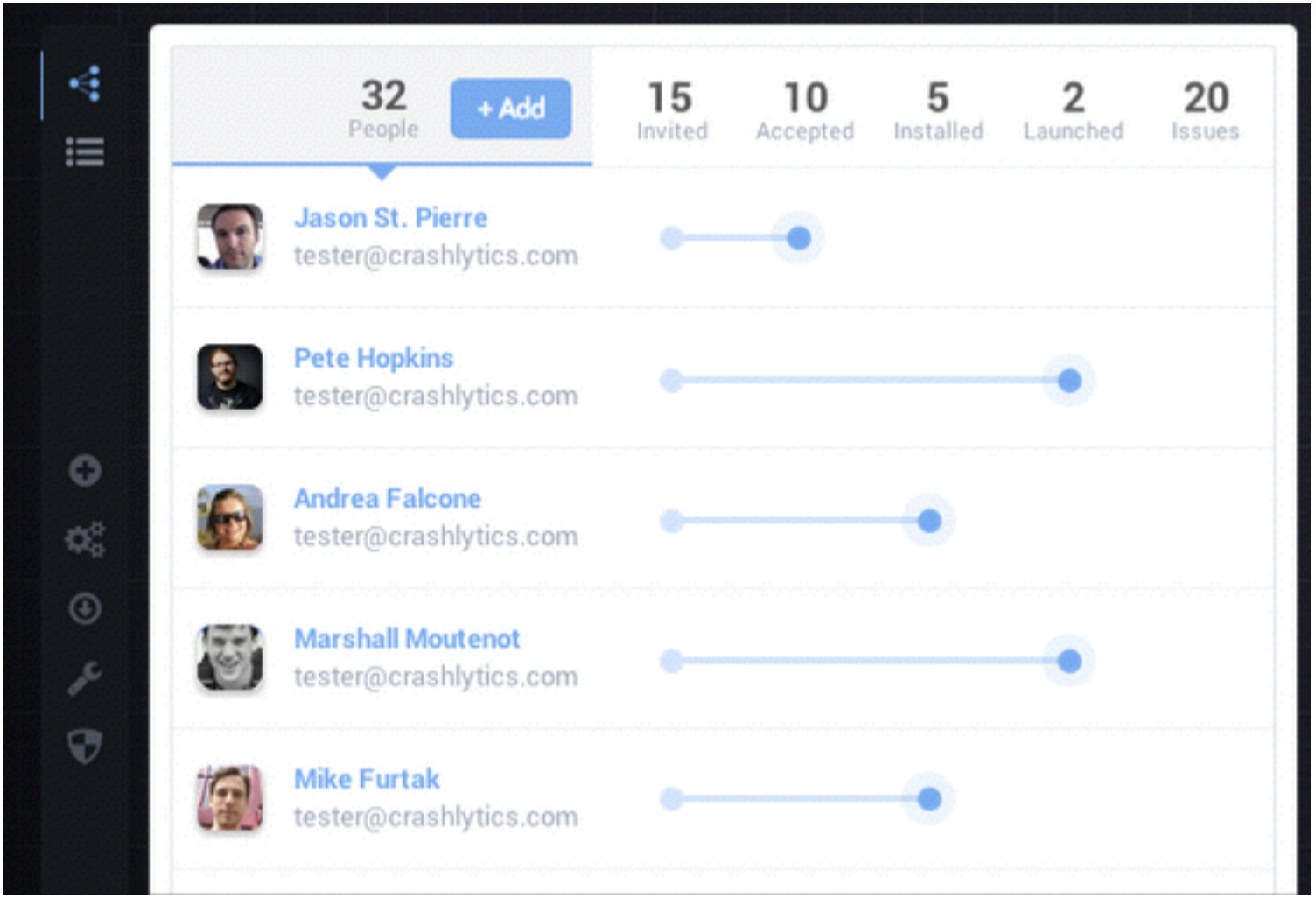
Crashlytics: Crashreports

Crashlytics: Crashreports



Crashlytics: User Statistic

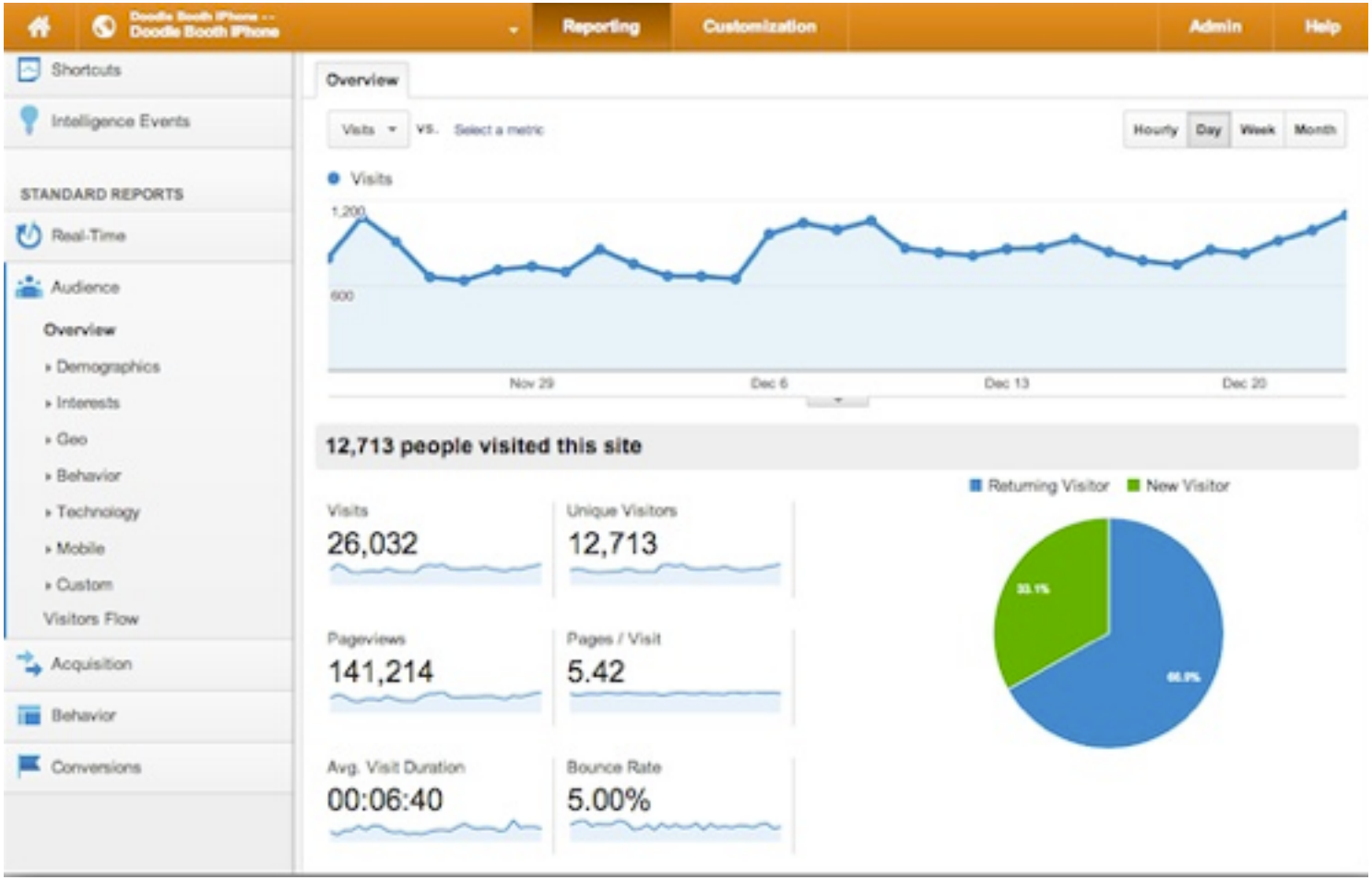
Crashlytics: Beta



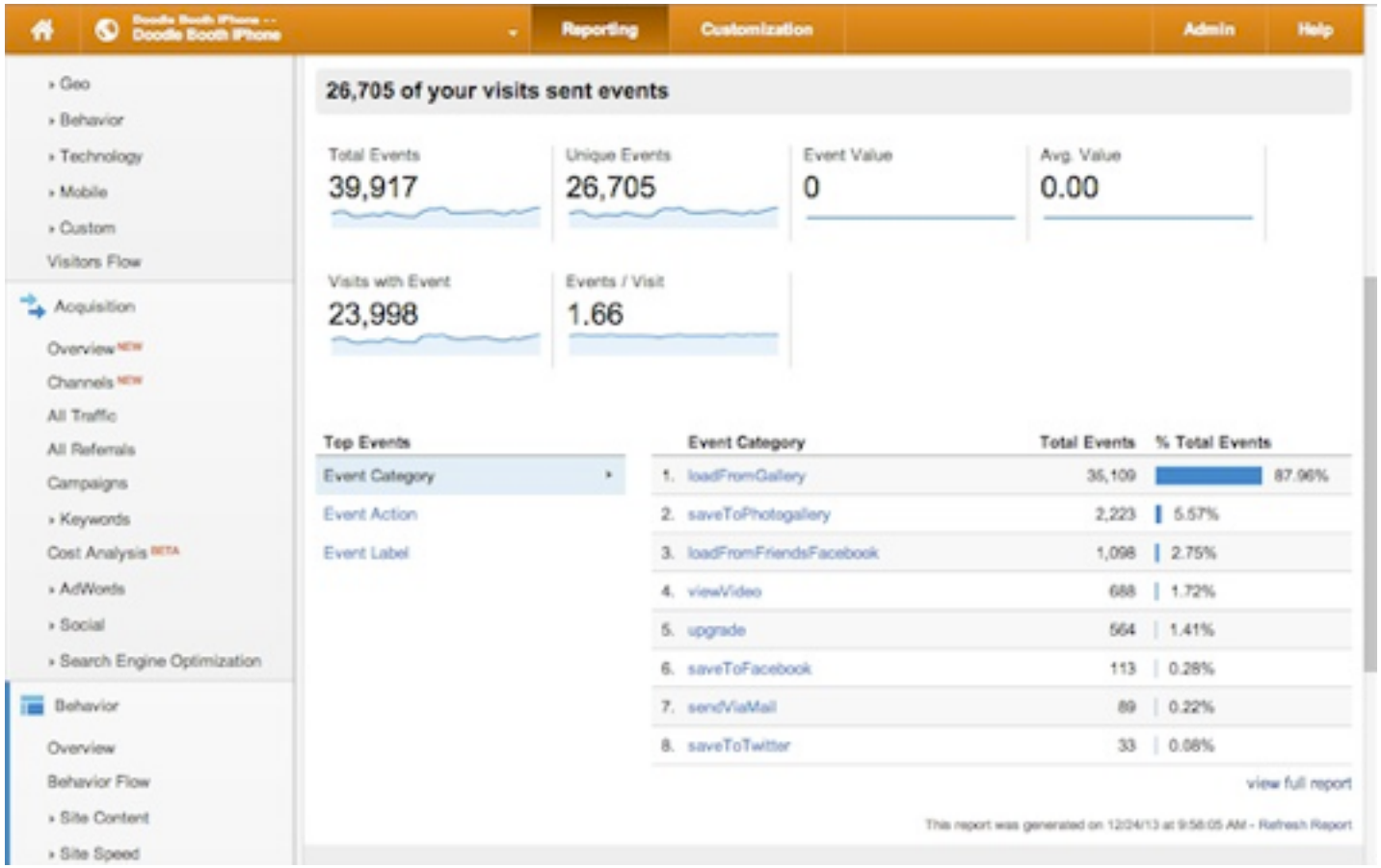
Crashlytics: Features

Crash Reports		eMail Notifications	
Beta-Testing		android Unterstützung	
Analytics		Integrierbar in Jenkins	
Integration		kostenlos	

Google Analytics: Genauere Benutzerdaten



Google Analytics: Ereignisse

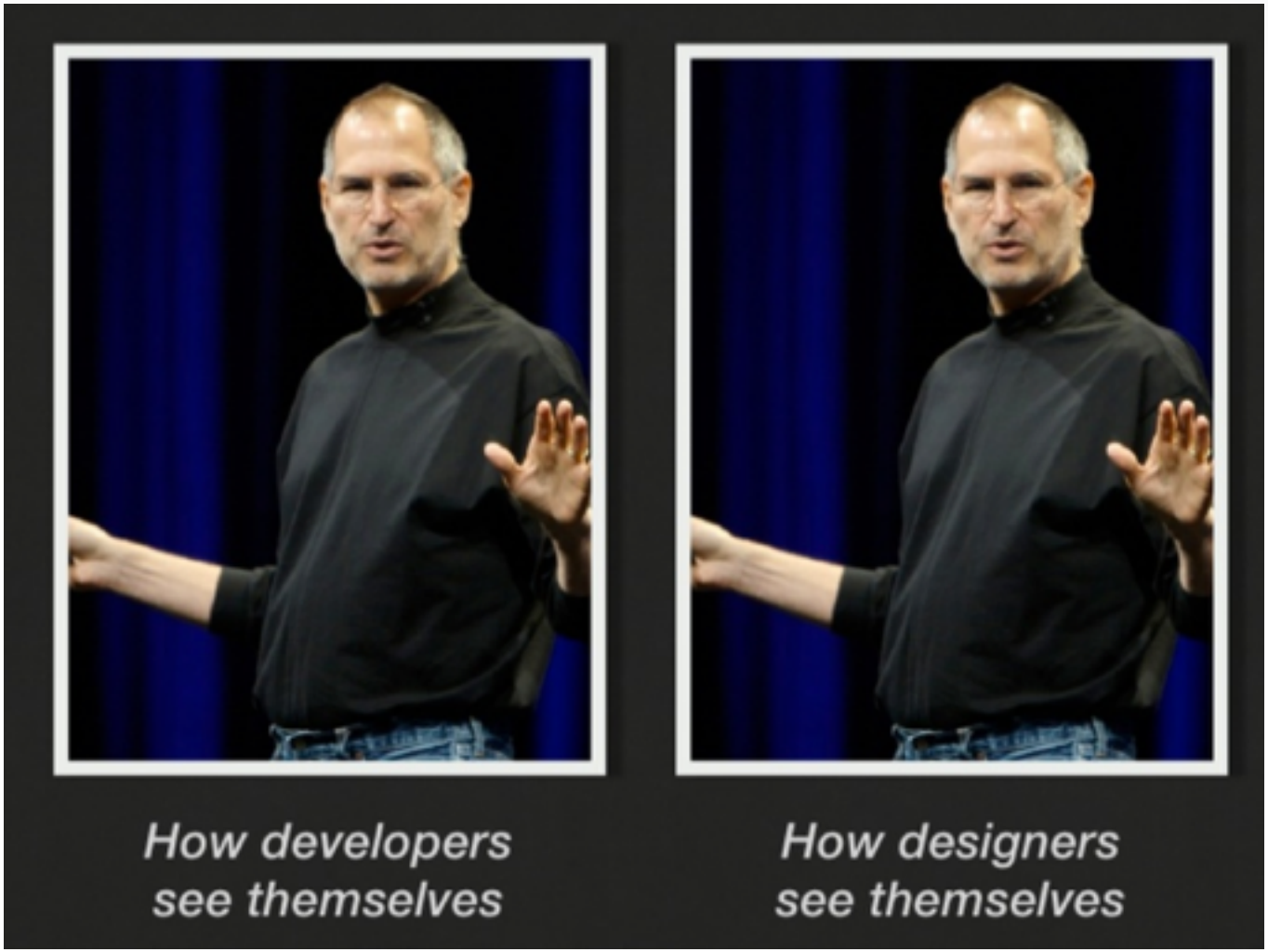


User Experience: Developer und Designer

User Experience: Developer und Designer

Agile iOS Stewart Gleadow @stewgleadow

User Experience: Developer und Designer



Agile iOS Stewart Gleadow @stewgleadow

User Experience: Developer und Designer

Gutes Design und intuitive Bedingung ist für (iOS-) Apps sehr wichtig

Designer ist ein wichtiger Teil eines (agilen) App-Teams

Zusammenarbeit und gegenseitiger Respekt erforderlich

Collective Code Ownership und Frameworks

Collective Code Ownership

Versionsverwaltung mit Git

- Wie kann es mit Storyboards gelingen?
  - Gleiche Version von Xcode im gesamten Team
  - Mehrere Storyboards
  - Storyboards oft mergen
    - FileMerge in Xcode verwenden

Wie Dateien organisieren?

- Beispiel
  - Application
  - Model
  - View
  - Resources
  - Library

Frameworks

- Dependency manager verwenden: Cocoapods
  - Seit Version 0.36 Swift kompatibel

- Viele Projekte verwenden zu viele Frameworks
  - Hohe Abhängigkeiten
  - Große Builds - ja nach Sprache
  - Gefahr, dass Frameworks nicht -schnell genug - angepasst werden
  - Hoher Aufwand für Anpassungen über Funktionalität des Frameworks hinaus
  - Oftmals Komplizierte Fehlersuche

- **Es sollte ausreichend Zeit in die Auswahl der Frameworks investiert werden**

- Must haves: MagicalRecord, mogenerator, RestKit, AFNetworking

Zusammenfassung

agile iOS- Entwicklung: Zusammenfassung

Kundeneinbeziehung und User Experience

Continuous Integration und Code Metriken

Agile iOS Entwicklung (Scrum)

Teststrategie

Nimble und Quick
Specta und Expecta

IDE und Clean Code

Programmiersprache

Danke für die Aufmerksamkeit

Fragen