ThoughtWorks studios

Agile Practices for iPhone Development

by ThoughtWorks Studios

Introduction

Since Apple® released the iPhone® SDK in 2008 we have seen an explosion of third party applications, with over 150,000 apps available and more than 4 billion downloads. The market for paid apps is estimated to be more than $250 million per year.

We have worked on some of these publicly available applications[1] and we anticipate an increasing demand for similar projects in the future. We also see demand for applications deployed and delivered inside organizations allowing access to internal applications while on the move.[2] We expect this to increase dramatically with the improved deployment support in iOS 4.0.

These projects generally favor small focused teams with highly iterative practices. In particular, the important new user experience paradigms that are associated with these devices mean that a compelling user experience requires early and frequent iteration of designs.

However, many of the tools that we take for granted in other development environments are missing or immature in the iPhone development space. We have found that high quality deliverables require a commitment to automated testing and continuous integration. Our commitment to this has led us in the past to help the community fill the gaps that exist in these tools. Many of the tools in other spaces were started or incubated by ThoughtWorks employees[3] (e.g. JMock, NMock, NUnit, Selenium, Webdriver). OCMock[4] (an Objective-C implementation of mock objects) was also written by a ThoughtWorks employee.

This white paper is an attempt to summarize our experience in the projects we have already undertaken, and to point out the gaps in the current set of tools and techniques. We hope that our contribution to the conversation will help guide tools evolution for this important application platform.

Motivation

The first wave of applications to be developed for the platform tended to be quickly developed to target very specific users. Getting to market early was key. As the platform matures, we are seeing more applications that form a core part of the business. And start-ups need to make an investment in the continuing quality of their software asset.

[1] See for example “New iPhone Application Using Offshore Agile” http://www.thoughtworks.com/pimsleur
[2] We are also seeing more reports of iPad enterprise acceptance amongst executives http://www.zdnet.com/blog/btl/tech-bigwigs-in-love-with-apples-ipad-push-business-case/34744
[3] http://opensource.thoughtworks.com
[4] OCMock Website http://www.mulle-kybernetik.com/software/OCMock/

Enterprises will have to maintain portfolios of custom applications, potentially for a long time and over multiple generations of devices. Enterprise users have different expectations of support and maintenance, and their operations teams will be asked to maintain an increasing number of mobile applications.

Early indications are that the iPhone is gaining traction in enterprises: AT&T recently reported that enterprise customers purchase 40% of iPhones.[5] In addition, our own clients who are interested in learning to leverage this new platform for their own business applications have approached us. Additionally there is increasing eagerness for applications that leverage the iPad™ unique selling points, and we are working with clients on such apps.

Our Analysis

Since rigorous engineering practices are at the core of how we develop software, we decided to evaluate the ecosystem in terms of how it supports the core Agile practices that are a part of the Extreme Programming (XP)[6] body of methods. Specifically, our lens focused on the following practices:

- Whole Team
- Test Driven Development
- Continuous Integration
- Small Releases

Whole Team

To effectively build any software application the whole team needs to be engaged and available. This means having customers and users who are engaged and available to the development team. For the small focused apps that thrive on the iPhone and similar devices this is even more critical. For most organizations all aspects of iOS development are new and unfamiliar. With such constraints it is crucial that the team is able to iterate its designs very quickly. This can only happen if the team is working closely.

The importance of User Experience

An iPhone application is the user interface. Build something that is not useful, usable or delightful and it will rapidly gather negative ratings, wither and die. Apple's very strong emphasis (and enforcement through the approval process) of a consistent set of user conventions means that iOS applications reinforce these affordances and train users with a powerful set of expectations on the behavior of applications on the device.

However, applications deployed within the enterprise are not naturally subject to these forces. Many enterprises do not contain dedicated and experienced User Experience (UX) experts. Even if

[5] AT&T reports 40% of iPhones sold to enterprise customers:
http://www.zdnet.com/blog/btl/at-t-exec-4-out-of-10-of-our-iphone-sales-to-enterprises/35145
[6] Wikipedia entry of Extreme Programming
http://en.wikipedia.org/wiki/Extreme_programming

there is an in-house UX team they probably do not have experience with these new devices and their conventions. This can lead to a poorly designed or alien-feeling user interface. On a device like the iPhone where the user experience is so consistent, even small divergences from existing user conventions are extremely jarring.

A combination of on-team UX expertise and an iterative approach is important to deliver a consistent and effective user experience. The right kind of UX expertise is difficult to access - even at ThoughtWorks where our UX experts[7] are very much in demand across the company. When a team is constrained in its access to UX expertise, the iterative approach and fast customer feedback become even more crucial.

Test Driven Development

In ThoughtWorks, Test Driven Development[8] (TDD) has been at the core of our development practices for the last decade. For us it is a core part of the design process. Many of us like to refer to Test Driven Design or Behavior Driven Design (BDD). So one of the first things we do on a project is decide on the unit testing approach that we will use.

Unit testing

Unit testing has been supported on Apple's Objective-C frameworks for a long time via the OCUnit testing framework.[9]

More recently OCUnit has been integrated into Apple's development environments and XCode wizards. Apple's recommended approach divides the tests into Logic Tests and Application Tests. Logic Tests run during the build process. One disadvantage of this approach is that it is not possible to debug Logic tests. Application Tests are built into a separate target that can be run and debugged on a real device.

Other approaches to unit testing are the Google™ Toolkit (GTM)[10] and GHUnit.[11] XCode wizards do not support these tools but they do provide a more flexible approach to testing. The recommended practice is to create separate targets for unit testing on simulators or the device itself. While GHUnit is interesting, particularly for the fact that it can create standard XUnit XML output, at present we prefer Google Toolkit.

Mock objects

A common unit testing technique on other platforms is mock testing. We replace parts of the system with fake objects and test interactions of the part of the system under test with these

[7] Experience Design Consulting
http://www.thoughtworks.com/experience-design-consulting
[8] Wikipedia entry for Test Driven Development (TDD) http://en.wikipedia.org/wiki/Test-driven_development
[9] Unit Testing With OCUnit http://developer.apple.com/tools/unittest.html
[10] Google Toolbox for Mac http://code.google.com/p/google-toolbox-for-mac/wiki/iPhoneUnitTesting
[11] GHUnit http://rel.me/2009/02/21/unit-testing-for-mac-os-x-and-iphone-ghunit/

fakes.[12] Mock testing is well supported on the iOS platform due to the dynamic nature of Objective-C. The long-standing OCMock[13] framework by ThoughtWorks' Erik Doernenburg is widely used.

Of particular interest is Doernenburg's mock-based approach to testing Cocoa controllers.[14] While this discussion involves testing desktop applications the approach and techniques are identical for iOS applications.

Acceptance/UI Testing

For UI functional and acceptance testing there are currently no "one-hundred percent satisfactory" solutions, although some interesting tools are emerging.

The most promising approaches learn from the successful approaches of Sahi, Selenium and Selenium Webdriver. They embed an HTTP server into the application under test that provides a simple HTTP protocol for driving the UI and retrieving information about the current state of the view. This enables integration with a variety of well-understood external tools such as Cucumber[15] (and our own Twist Agile test automation tool[16]).

Choosing a UI Testing Strategy

Until the UI testing frameworks mature we recommend light automated UI testing (smoke testing) using one of the frameworks we discuss above, sufficient to exercise basic functionality. We believe that a large investment in any particular UI testing framework is not yet justified. Until the tools stabilize, teams will need to invest more time in manual testing.

Since some issues are not evident in device simulators it is particularly important to test on the oldest supported device, since this brings out issues that are masked in the simulator and newer devices with different memory and performance constraints.

Continuous Integration

In the past, ThoughtWorks' teams have found that the technical hurdles to implementing automated continuous integration on the iOS platform outweigh the potential benefits that it would bring. Our teams have successfully used lightweight automated continuous integration, compiling the app with 'treat warnings as errors' turned on for example, but this is not a permanent solution for the need.

While this is a pragmatic choice for small teams it is more difficult to sustain quality across a portfolio of projects in the longer term. The tools and frameworks are rapidly maturing to the point where automated continuous integration can become as ubiquitous as it is on other development platforms.

---

[12] Mock Objects http://www.mockobjects.com/
[13] OCMock http://www.mulle-kybernetik.com/software/OCMock/
[14] Testing Cocoa Controllers with OCMock http://erik.doernenburg.com/2008/07/testing-cocoa-controllers-with-ocmock/
[15] Cucumber
http://cukes.info/
[16] Twist Agile Test Automation
http://www.thoughtworks-studios.com/agile-test-automation

A current limitation is poor support for automatically provisioning and running applications on the simulator or a real device. It is possible to achieve this using Applescript® to drive the XCode environment but this is still awkward and makes testing some scenarios difficult. For example it is useful to be able to test the behavior of the application when it is closed and then re-opened. But this is not simple if closing and re-opening the system is not possible.

Small Releases

Automated deployment in an enterprise is currently somewhat cumbersome. Provisioning and deploying applications to devices within the enterprise will need to be improved before we can consider a true continuous deployment solution for iOS apps.

The increasing demand for enterprise mobile applications will present unique challenges for already stretched operations staff. Mobile applications tend to be small and narrowly focused, and operations teams will be asked to manage many of these applications in the coming years.

Improvements in iOS 4.0 promise to address this issue[17], but it is too early to assess these new provisioning approaches. We will be watching future announcements closely.

Conclusion

The Agile practices and techniques that ThoughtWorks has been using and refining for many years still provide value in development of mobile applications. However the tools that support these practices are not yet as mature as in other platforms.

To build an effective team we believe organizations should invest in UX expertise and expect to iterate their designs aggressively. Development practices such as automated unit testing and continuous deployment are important in such highly iterative teams.

Currently due to relatively immature Acceptance and UI testing frameworks we recommend that teams concentrate primarily on manual testing on real devices in the short term, while carefully evaluating the emerging UI testing frameworks.

Continuous deployment support is critical to the success of mobile applications in the longer term, particularly in organizations that will support a portfolio of applications. We'd like to see improved scriptability of deployment and provisioning tools so that automation can be more widely used.

We are excited by the potential for these new devices and we expect them to play an increasingly important role as enterprises embrace mobile technologies. We will be doing our part to help evolve the supporting tools and techniques.

[17] "iPhone OS4 Offers a lot for the Enterprise", Computerworld
http://www.computerworld.com/s/article/9175149/iPhone_OS_4_offers_a_lot_for_the_enterprise

Acknowledgements

This paper collects the learning from many people and project teams, including Giles Alexander, Farooq Ali, Ola Bini, Duncan Cragg, Evan Bottcher, Erik Doernenburg, Matthew Dunn, Peter Hodgson, John Kordyback, Derek Longmuir, Tyler Mercier, Adam Monago, Jeff Norris, Jason Sallis, Danilo Sato, Chris Stevenson, Chad Wathington, Jeff Wishnie, Jonathan Wolter, Andy Yates.

About ThoughtWorks Studios

ThoughtWorks Studios is a global leader in Agile ALM tools and training, and its products can be found in development organizations seeking sustainable Agile adoption. The company's Adaptive ALM solution provides a platform for managing all aspects of software development, from requirements definition and project management to test automation, quality assurance, and release management. It consists of the integration of three products: Mingle® (project management), Twist® (test automation) and Cruise® (release management). Each tool is available as part of a complete lifecycle solution or as a stand-alone product. Backed by more than 16 years of experience in Agile delivery, ThoughtWorks Studios is the product division of ThoughtWorks, Inc., the pioneering leader in Agile development. ThoughtWorks Studios has 200 customers in more than 20 countries, including 3M, Honeywell, BBC, eBay, Barclays, Vodafone, McGraw-Hill and Rackspace. The company headquarters is co-located in San Francisco and Bangalore, with offices in London and select cities in Europe, Asia and Australia. For more information, visit www.thoughtworks-studios.com

Mingle, an Agile management and collaboration tool, provides a common workspace for all team members and an automated system of record for all projects. Mingle can adapt to any existing workflow process and easily manages daily development activities. Offering true-to-life visibility into the entire development process for all stakeholders, Mingle helps development teams become more open and collaborative.

Twist, an automated testing solution, provides English-like constructs, making the testing process more productive for all team members. As applications grow in complexity, Twist helps to more easily maintain complex test suites. These suites keep pace with application development and are held as long-living assets.

Cruise provides both continuous integration and release management and can drive development and IT operations to collectively plan continuous product releases. Cruise offers deployment pipelines and a zero-configuration build grid, which simplify the release management process. Unlike open-source tools, Cruise scales to meet the needs of a complex development project with numerous dependencies.

ThoughtWorks Studios

CALL: 512-467-4956 (sales)
415-238-6497 (main) North America
+91 80 4064 9703 | Rest of the world

EMAIL: studios@thoughtworks.com
WEB: www.thoughtworks-studios.com