MAD FOR MOBILE

John Frizelle
Philip Hayes
Cian Clarke

#redhat #rhsummit
THE MOBILE CENTER OF EXCELLENCE

APPS DON'T NEED TO

- Cost $100k
- Take 6 months to develop
- Live for decades
- Be big monoliths on the client side
THE MOBILE CENTER OF EXCELLENCE

WHY YOU NEED ONE

- In-house skills for an important delivery channel
- Not Corporate IT => Fast IT

MOBILE IS DIFFERENT

- Lots of small apps, fast update cycles
- Offline support means backends can fail - records still exist on mobile device
- Move fast & break stuff
REINVENT THE RULES

AGILE

DEVOPS

MICROSERVICES
AGILE

DEVOPS

MICROSERVICES
THREE SIMPLE TRUTHS OF SOFTWARE DEVELOPMENT

1. It is impossible to gather all the requirements at the beginning of a project.

2. Whatever requirements you do gather are guaranteed to change.

3. There will always be more to do than time and money will allow.
WHAT IS AGILE?

- A **time boxed, iterative** approach to software delivery that builds software **incrementally** from the start of the project, instead of trying to deliver it all at once near the end.

- Break projects down into little bits of user functionality called **user stories**, prioritizing them, and then **continuously delivering** them in short **two week cycles** called **iterations**.
WHAT IS AGILE?
HOW DOES IT WORK?

- Make a list
- Size things up
- Set some priorities
- Start Executing
- Update the plan as you go
HOW IS IT DIFFERENT?

- Analysis, design, coding, and testing are continuous activities
- Development is iterative
- Planning is adaptive
- Roles Blur
- Scope can (and will) vary
- Requirements can (and will) change
- Working software is the primary measure of success
HOW IS IT DIFFERENT?
## AGILE VS WATERFALL

<table>
<thead>
<tr>
<th></th>
<th>Waterfall</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality</td>
<td>Poor - testing at the end</td>
<td>Good - testing from day one</td>
</tr>
<tr>
<td>Visibility</td>
<td>Poor - have to wait until the end</td>
<td>Good - ship full features incrementally</td>
</tr>
<tr>
<td>Risk</td>
<td>High - schedule, technical, product</td>
<td>Low - early &amp; iterative feedback</td>
</tr>
<tr>
<td>Change</td>
<td>Poor - stick to the plan</td>
<td>Good - welcome &amp; adapt to change</td>
</tr>
</tbody>
</table>
HOW IS IT DIFFERENT?

Frequent Release Events
“Agile Methodology”

Smoother Effort
Less Risk

Rare Release Events
“Waterfall Methodology”

Effort Peaks
High Risk
BACKLOG PRIORITISATION

- Backlog = All remaining work
- Frequent & Small reviews
- Managed by Product Owner
- Constant refinement - break down stories
- Look 3 to 5 sprints out
USER STORIES

- High level features or requirements
- Just enough to start a conversation
- Stories can split or join
- No more than a few days work each
ESTIMATION

- Don't use days - it never works
- Relative sizing based on complexity
- Agree what a 1, 2, 5 point story is
- Use Planning Poker to avoid consensus bios
PLANNING

- Velocity = speed from story to product
- #Iterations = total effort / velocity
- Ongoing & adaptive
- Use Burndown Charts
PLANNING
ITERATIONS

- Time Boxed - Usually 2 weeks
- Take highest priority stories from Backlog
- Produce a shippable increment
- Past performance is a guide to future performance
ITERATIONS

Plan  Collaborate  Deliver

Agile Project Management: Iteration
AGILE MANIFESTO

HTTP://AGILEMANIFESTO.ORG/

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.
AGILE AND DEVOPS

Agile what?

Waterfall

"Agile"

Continuous Integration

Continuous Delivery

DevOps

Agile teams within the Enterprise

An Agile Enterprise
WHAT IS DEVOPS?
Agile relationship between Development and IT Operations
Devops tools

- CI Server (JenkinsCI - recommended, CodeShip, CircleCI etc.)
- Git SCM
- Ansible
- Red Hat Mobile Application Platform
- CI Server plugins (JenkinsCI pipeline, GitHub, Bitbucket plugins etc.)
- Build tools (Gulp, Grunt, Maven, Gradle etc)
- Testing frameworks (Mocha, Jasmin, Cucumber etc.)
- Command line tools (fhc, oc, custom tools).
- External services (GitHub, GitLab, BitBucket, Slack, IRC etc.)
- Monitoring (Nagios)
INTEGRATION LIFECYCLE

Developers produce application code and tests are passing locally.

Pull / Merge: Failed project tests; dev team is notified (Slack, email, etc.).

Code changes/new code pushed to project Git repo (Git pull request).

Update pull request label with pass or fail information.

JenkinsCI polls the Git repo for new commits/changes. Start new CD pipeline run if code changes found.

CD pipeline gets project code and performs tests locally.

If app tests fail:

If app tests pass:

Git push application code to RHMAP.

Build application/Store Cordova binary.

Deploy/promote app between environments.
Unit and Acceptance test best practices
Flexible deployment options
Lifecycle management
Health endpoints
Controlled UAT and Production deployment (cloud and MBaaS services)
App builds (build farm)
App deployment (MDM integration)
AGILE

DEVOPS

MICROSERVICES
WHY MOBILE MICROSERVICES?

TRADITIONAL APPROACH

- System of Record
- Monolithic
- Multi Featured
- Long Development Cycle
- Costly
- Infrequent, Large Updates

THE MOBILE WAY

- System of Engagement
- Simple, Nimble Apps
- Highly targeted to need
- Fast Development Cycle
- Affordable
- CD & CI
MICROSERVICES
IT'S JUST SOA RIGHT?!

WHAT ARE THEY?
- Small, composable services
- Responsible for small pieces of work
- Talk over "dumb" pipes, with "smart" endpoints

WHAT ARE THEY NOT?
- Monoliths
- A "Silver Bullet"
- Just SOA Again
MOBILE IS DIFFERENT

FAST TO DEPLOY
• Quick release cycle
• Loosely coupled API Integrations

PERFORMANT NETWORK CONNECTION
• Can make any request
• Payload size not a big concern

SLOW TO DEPLOY
• Release cycle up to 1-week with public app store reviews
• Tightly coupled API Integrations

LOSSY EDGE NETWORKS, 3G BEST CASE
• HTTP overhead slow – need to make fewer requests
• Payload must be small
  - trimmed for mobile
UPDATING IS DIFFERENT

<table>
<thead>
<tr>
<th></th>
<th>Hours</th>
<th>Days</th>
<th>Weeks to Months</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Application</td>
<td>Deploy ✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deploy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mobile App</td>
<td>Force update</td>
<td>User’s device automatically updates ✔️</td>
<td></td>
</tr>
<tr>
<td>Private App store</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mobile App</td>
<td>Upload to appstore</td>
<td>Review</td>
<td>User updates app</td>
</tr>
<tr>
<td>Public App store</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time

---

7.5
PERFORMANCE IS DIFFERENT
MICROSERVICES ARE

- Suited to mCOE Development Practices
- Suited to mobile
- Different from developing a monolith
MOBILE COE

Brings new release cycles
Brings a new pace of business
Delivers software differently
Gives greater ROI on $ IT spend
SECTION HEADLINE
SECTION HEADLINE

#redhat #rhsummit
MAJOR HEADLINE

SUBTITLE

SMALLER TITLE

normal text

#redhat #rhsummit
LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

#redhat #rhsummit