ThoughtWorks®

Technology Radar

OUR THOUGHTS ON JAVASCRIPT, APIS, CONWAY'S LAW, RE-DECENTRALIZATION AND MUCH MORE

JULY 2014

thoughtworks.com/radar

WHAT'S NEW?

Here are the trends highlighted in this edition:

CHURN IN THE JAVASCRIPT WORLD

We thought the rate of change in the Ruby open source space was rapid until the full rush of JavaScript frameworks arrived. JavaScript used to be a condiment technology, always used to augment other technologies. It has kept that role but expanded into its own platform with a staggering rate of change. Trying to understand the breadth of this space is daunting, and innovation is rampant. Like the Java and Ruby open source spaces, we hope it will eventually calm to at least a deluge.

CONWAY'S LAW

Conway's Law, that states that "organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations", keeps appearing in unexpected places. One of the keys tenants of the Agile Manifesto is "People over Processes and Tools", and we see Conway's Law reinforcing this idea both negatively and positively. Some companies are mired in siloed structures that add needless friction to engineering efforts, while more enlightened companies use team organization to drive the kinds of architectures they want. We're learning the peril of ignoring Conway's Law and the benefits of leveraging it.

MICROSERVICES AND THE RISE OF THE API

We are seeing an incredible amount of interest in microservice architectures, as well as an emphasis on the importance of the API both within an organization and as a bridge to the outside world. In a microservice architecture a large number of very small services are deployed and linked up to build systems, with the services mapping closely to business concepts and value. In order to make this approach work, teams need good discipline around building, testing, integrating and then managing the services. This edition of the Radar tracks some of the specific tools and techniques for microservices.

RE-DECENTRALIZATION

The Internet began life as a distributed system, but over the last decade or so we have seen an increasing amount of centralization of both services and data. As an example, over 90% of the world's email moves through just 10 providers. Similarly with Cloud computing, a small number of providers service the vast majority of our Cloud needs. Prompted in part by revelations about the US' stranglehold on Internet infrastructure, and a desire to maintain more individual and organizational control, we see a need for "re-decentralization" of both data and infrastructure.

ABOUT THE TECHNOLOGY RADAR

ThoughtWorkers are passionate about technology. We build it, research it, test it, open source it, write about it, and constantly aim to improve it – for everyone. Our mission is to champion software excellence and revolutionize IT. We create and share the ThoughtWorks Technology Radar in support of that mission. The ThoughtWorks Technology Advisory Board, a group of senior technology leaders in ThoughtWorks, creates the radar. They meet regularly to discuss the global technology strategy for ThoughtWorks and the technology trends that significantly impact our industry.

The radar captures the output of the Technology Advisory Board's discussions in a format that provides value to a wide range of stakeholders, from CIOs to developers. The content is intended as a concise summary. We encourage you to explore these technologies for more detail. The radar is graphical in nature, grouping items into techniques, tools, platforms, and languages & frameworks. When radar items could appear in multiple quadrants, we chose the one that seemed most appropriate. We further group these items in four rings to reflect our current position on them. The rings are:

ADOPT

We feel strongly that the industry should be adopting these items. We use them when appropriate on our projects.

TRIAL

Worth pursuing. It is important to understand how to build up this capability. Enterprises should try this technology on a project that can handle the risk.

ASSESS

Worth exploring with the goal of understanding how it will affect your enterprise.

HOLD

Proceed with caution.

Items that are new or have had significant changes since the last radar are represented as triangles, while items that have not moved are represented as circles. The detailed graphs for each quadrant show the movement that items have taken. We are interested in far more items than we can reasonably fit into a document this size, so we fade many items from the last radar to make room for the new items. Fading an item does not mean that we no longer care about it.

For more background on the radar, see thoughtworks.com/radar/#/faq

CONTRIBUTORS

The ThoughtWorks Technology Advisory Board is comprised of:

Rebecca Parsons (CTO)
Martin Fowler(Chief Scientist)
Badri Janakiraman
Brain Leke
Claudia Melo
Erik Doernenburg
Evan Bottcher
Hao Xu
Ian Cartwright
James Lewis
Jeff Norris
Jonny LeRoy
Mike Mason
Neal Ford
Rachel Laycock
Sam Newman
Scott Shaw
Srihari Srinivasan
Thiyagu Palanisamy

THE RADAR

TECHNIQUES

ADOPT
1. Forward Secrecy
2. Segregated DOM plus node for JS Testing

TRIAL
3. Capture domain events explicitly
4. Development environments in the cloud
5. Event Sourcing
6. Focus on mean time to recovery
7. Humane registry
8. Inverse Conway Maneuver
9. Living CSS Style Guides
10. Machine image as a build artifact
11. Masterless Chef/Puppet
12. Perimeterless enterprise
13. Provisioning testing
14. Real user monitoring
15. REST without PUT
16. Structured logging
17. Tailored Service Template

ASSESS
18. Bridging physical and digital worlds with simple hardware
19. Datensparsamkeit
20. Machine image pipelines
21. Pace-layered Application Strategy
22. Property-based unit testing
23. Tangible interaction

HOLD
24. Cloud lift and shift
25. DevOps as a team
26. Ignoring OWASP Top 10
27. Testing as a separate organization
28. Velocity as productivity

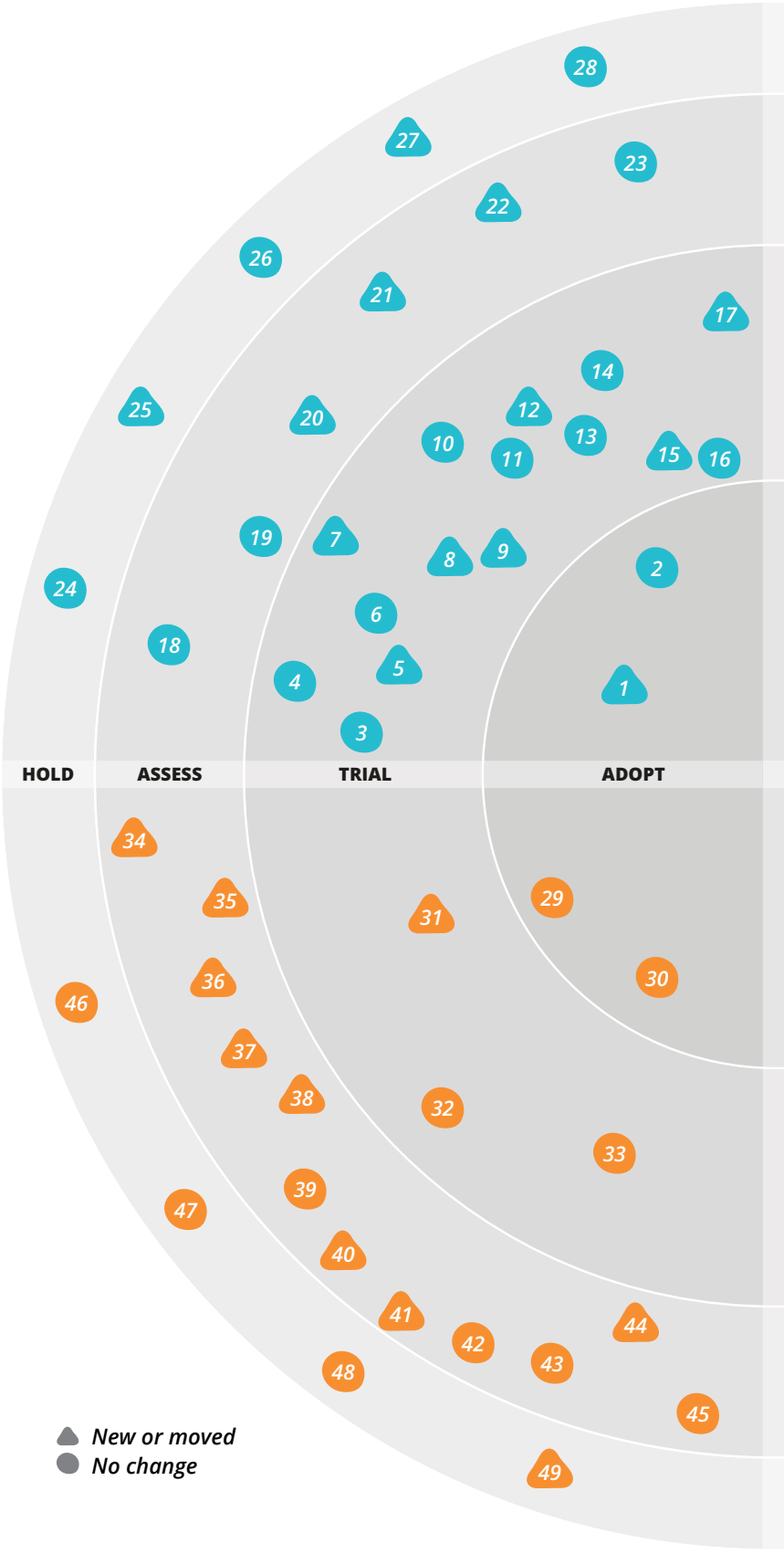
PLATFORMS

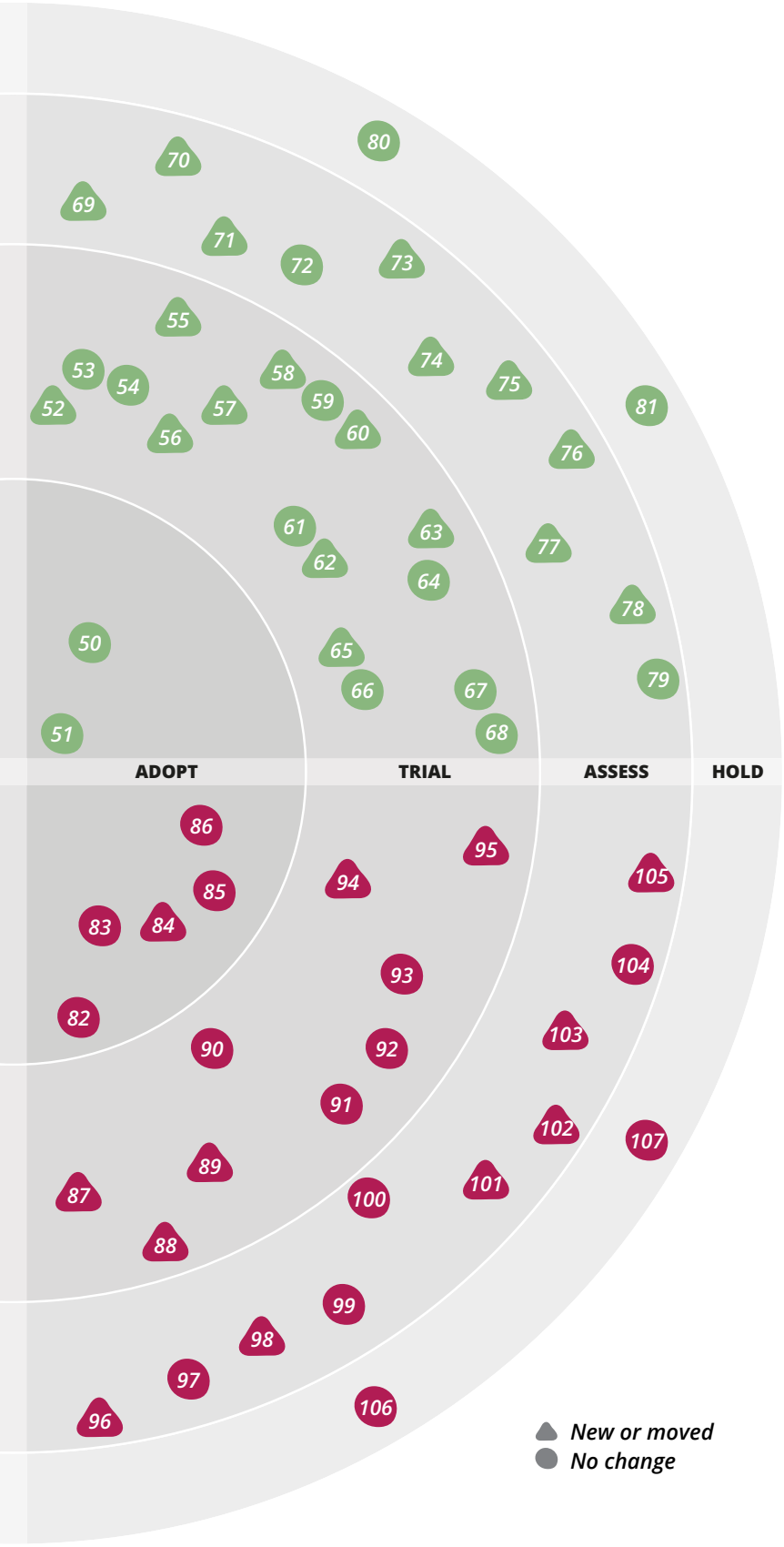
ADOPT
29. Hadoop 2.0
30. Vumi

TRIAL
31. iBeacon
32. PostgreSQL for NoSQL
33. Private Clouds

ASSESS
34. ARM Server SoC
35. CoAP
36. DigitalOcean
37. Espruino
38. EventStore
39. Low-cost robotics
40. Mapbox
41. OpenID Connect
42. SPDY
43. Storm
44. TOTP Two-Factor Authentication
45. Web Components standard

HOLD
46. Big enterprise solutions
47. CMS as a platform
48. Enterprise Data Warehouse
49. OSGi





THE RADAR

TOOLS

ADOPT
50. Ansible
51. Dependency management for JavaScript

TRIAL
52. CartoDB
53. Chaos Monkey
54. Docker
55. Flyway
56. Foreman
57. GenyMotion
58. Go CD
59. Grunt.js
60. Gulp
61. Moco
62. Packer
63. Pact & Pacto
64. Prototype On Paper
65. Protractor for AngularJS
66. SnapCI
67. Snowplow Analytics & Piwik
68. Visual regression testing tools

ASSESS
69. Appium
70. Consul
71. Flume
72. Hosted solutions for testing iOS
73. leaflet.js
74. Mountebank
75. Papertrail
76. Roslyn
77. Spark
78. Swagger
79. Xamarin

HOLD
80. Ant
81. TFS

LANGUAGES & FRAMEWORKS

ADOPT
82. Dropwizard
83. Go language
84. Java 8
85. Reactive Extensions across languages
86. Scala, the good parts

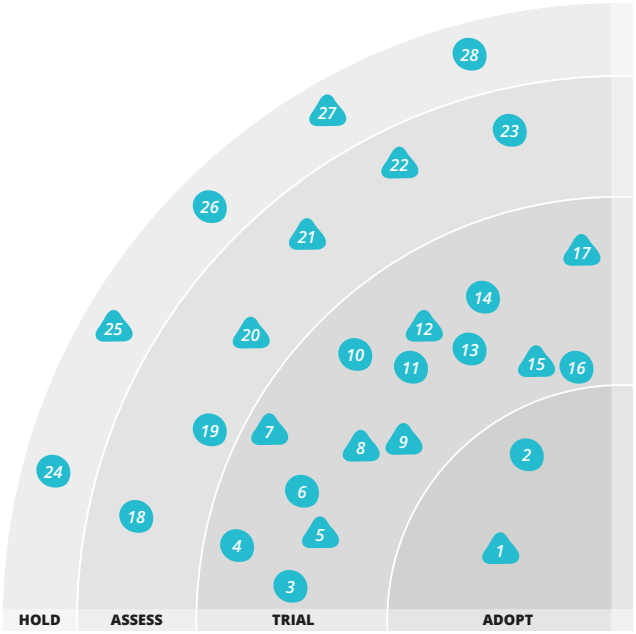
TRIAL
87. AngularJS
88. Core Async
89. HAL
90. Hive
91. Nancy
92. Pester
93. Play Framework 2
94. Q & Bluebird
95. R as Compute Platform

ASSESS
96. Elm
97. Julia
98. Om
99. Pointer Events
100. Python 3
101. Rust
102. Spray/akka-http
103. Spring Boot
104. TypeScript
105. Wolfram Language

HOLD
106. Handwritten CSS
107. JSF

TECHNIQUES

Forward Secrecy (sometimes known as "Perfect Forward Secrecy" or PFS) is a cryptographic technique that protects previous communications sessions even if a server's master keys are later compromised. Despite being simple to enable for HTTPS connections, many servers are not configured this way, and we recommend enabling forward secrecy to improve security. Note that we don't generally like the word "perfect" when used to describe cryptographic protocols — even the best protocol can be broken by a flaw in implementation, random number generator, or by advances in cryptanalytic techniques. Even so, it's important to enable the best security available, whilst keeping informed of new attacks and protocol improvements.



Event Sourcing ensures that all changes to application state are stored as a sequence of events. Not just can we query these events, we can also use the event log to reconstruct past states, and as a foundation to automatically adjust the state to cope with retroactive changes. Complementary to the capture of business meaningful events, the technique has positive implications for analytics in driving greater customer insight.

A Microservice architecture by it's very nature increases significantly the number of applications, services, and interactions in your deployed environments. Our projects are showing renewed focus on building **Humane Registries** (martinfowler.com/bliki/HumaneRegistry.html) which aggregate information about running services from the live environment and present it in a form for humans to comprehend. These registries favor up-to-date information from running services instead of human-curated documentation.

Conway's Law asserts that organizations are constrained to produce application designs which are copies of their communication structures. This often leads to unintended friction points. The '**Inverse Conway Maneuver**' recommends evolving your team and organizational structure to promote your desired architecture. Ideally your technology architecture will display isomorphism with your business architecture.

ADOPT

1. Forward Secrecy
2. Segregated DOM plus node for JS Testing

TRIAL

3. Capture domain events explicitly
4. Development environments in the cloud
5. Event Sourcing
6. Focus on mean time to recovery
7. Humane registry
8. Inverse Conway Maneuver
9. Living CSS Style Guides
10. Machine image as a build artifact
11. Masterless Chef/Puppet
12. Performance Monitoring
13. Perimeterless enterprise
14. Provisioning testing
15. REST without PUT
16. Structured logging
17. Tailored Service Template

ASSESS

18. Bridging physical and digital worlds with simple hardware
19. Datensparsamkeit
20. Machine image pipelines
21. Pace-layered Application Strategy
22. Property based unit testing
23. Tangible interaction

HOLD

24. Cloud lift and shift
25. DevOps as a team
26. Ignoring OWASP Top 10
27. Testing as a separate organization
28. Velocity as productivity

TECHNIQUES *continued*

A **living CSS style guide** is a page on your site that uses your current CSS styles and acts as a reference for all the currently available visual elements and design patterns. This helps to tightly integrate design into your delivery process by promoting co-ownership of the UI and avoids duplication of styling across your application. Styling changes are visible in the guide immediately and changes propagate across your site from a central location. A sensible way to do this is with a well organized SASS/LESS file structure with semantically named elements that separates structure, aesthetics, and interaction.

With the proliferation of single-page JavaScript applications, we have found that slow Ajax calls, excessive DOM manipulation, and unexpected JavaScript errors in the browser can have a big impact on perceived website responsiveness. It is very useful to collect and aggregate this profiling information from real end-user's browsers. **Real user monitoring** (newrelic.com/real-user-monitoring) provides early warning and diagnosis of production issues, and helps pinpoint them to a specific locality.

In the last radar we talked about Capturing Explicit Domain Events, putting emphasis on recording the business-meaningful events that have triggered state transitions instead of just CRUD'ing entities. REST interfaces commonly use PUT to update resource state, however it's often better to POST to record a new event resource which captures intent. **REST without PUT** has a side-benefit of separating command and query interfaces and forces consumers to allow for eventual consistency.

We see multiple organizations creating a **Tailored Service Template** which can be used to quickly seed new services, pre-configured to operate within that organization's production environment. The template contains a default set of decisions such as web frameworks, logging, monitoring, build, packaging, and deployment approaches. This is a very useful technique for encouraging collaborative evolution while retaining lightweight governance.

Many deployments requires machine images for different server roles like applications and services, databases, and reverse proxies. Because building a machine image from scratch, using an operating system ISO and provisioning scripts, can take a considerable amount of time it can be useful to create a **build pipeline for machine images**. The first stage in the pipeline sets up a base image according to general standards in the organisation. Subsequent stages can then enhance the base image for different purposes. If several applications or services have similar requirements, an application server for example, the pipeline can be extended by an intermediate stage, which takes the base image and provides an image with an application server but no application/service. These pipelines are not linear, they are trees that are branching out from the base image.

Gartner's **Pace-layered Application Strategy** approach to architecture attempts to articulate the fact that decisions about architecture shouldn't be a one-size fits all approach. Instead, it is important to take a balanced view to your technology portfolio in terms of where to be conservative, and where to take risks. While we have qualms about some of the more prescriptive recommendations that seem to come with Pace, in general we like the concept and many organizations could benefit from adapting similar models.

We value unit testing on projects and we like techniques such as **property-based unit testing** which augment it. This is a practice of using data generators to create randomized inputs within defined ranges. It allows a quick check for boundary conditions and other unanticipated failure modes and has burgeoning support on multiple platforms.

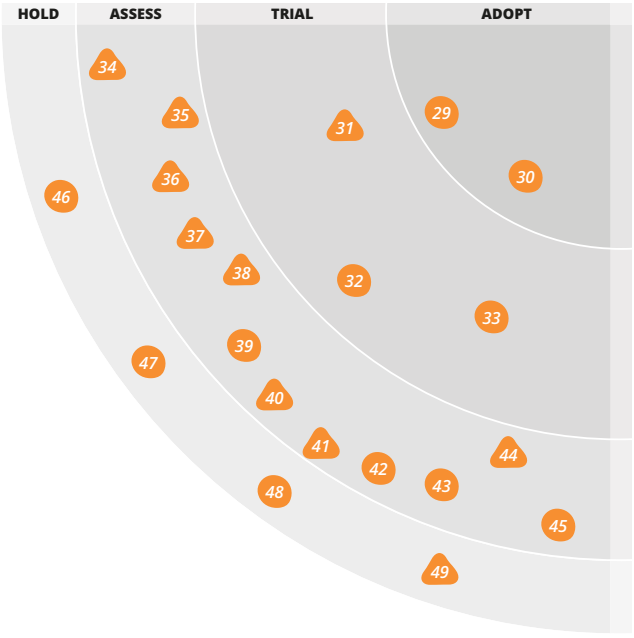
Some companies with good intentions create a separate **DevOps team**, which misconstrues the definition of DevOps. Rather than a role, DevOps is a cultural movement encouraging collaboration between operations specialists and developers. Rather than create yet another silo and suffer the consequences of Conway's Law, we advise you to embed these skills into teams, improving feedback loops and communication pathways by removing friction.

We continue to see organizations create separate Development and QA teams. Fast feedback is a core tenet of Agile and critical to the success of a project. Using a separate QA team slows down this feedback, creates an "us and them" mentality and makes it more difficult to build quality into the software. Testing should be a tightly integrated activity and isn't something the team can outsource. We recommend integrated teams where testers work closely with developers instead of having **testing as a separate organization.**

PLATFORMS

iBeacons are the Apple implementation of the broader category of beacons, which are small devices that use low energy Bluetooth (BLE) to provide fine-grained proximity information for mobile phones and other devices. Despite the hype surrounding iBeacons and the limitations to the accuracy and reliability of the information they provide, we do feel that they open interesting opportunities as trigger points for interacting with your users in a contextually relevant manner.

AMD recently released an 8-core **ARM SoC** (System on a Chip) designed for servers and has committed to releasing an ARM SoC with integrated graphics in 2015 (anandtech.com/show/7989). ARM-based servers are an interesting alternative to x86 because they are significantly more energy efficient. For some workloads, building an ARM-powered Cloud is preferable.



CoAP is an open standards communication protocol for the Internet of Things (IoT). While there is currently a proliferation of competing standards in the IoT space, we particularly like CoAP. It is specifically designed for resource-constrained devices and local radio networks. It uses UDP for transport, but is semantically compatible with HTTP. CoAP uses a web-based model of devices with their own URLs and a request-response paradigm that supports RESTful and decentralized approaches.

Although the IaaS space is crowded, there is room for new competitors to enter the market. **DigitalOcean** (digitalocean.com) has impressed us recently with its cost, speed and simplicity. If all you need is basic compute infrastructure, it is well worth a look.

Espruino is a microcontroller that natively executes JavaScript and thus lets the large number of JavaScript programmers get started very quickly. Using an event-based model similar to Node.js, Espurino devices can be very power efficient while still being responsive. Less powerful than a Raspberry Pi and slightly slower than an Arduino, Espruino makes an interesting alternative in low-power environments that need responsive behavior but can sacrifice some of the raw high level features and execution speed of those platforms.

Given the popularity of event sourcing, it is no surprise that tools in this space are maturing. **EventStore** (geteventstore.com) is an open source functional database for storing immutable events and performing complex event processing on the event streams. Unlike other tools in this space, EventStore exposes event streams as Atom collections which therefore require no special infrastructure such as message buses or highly specialized clients to use.

ADOPT
29. Hadoop 2.0
30. Vumi

TRIAL
31. iBeacon
32. PostgreSQL for NoSQL
33. Private Clouds

ASSESS
34. ARM Server SoC
35. CoAP
36. Digital Ocean
37. Espruino
38. EventStore (geteventstore.com)
39. Low-cost robotics
40. Mapbox
41. OpenID Connect
42. SPDY
43. Storm
44. TOTP Two-Factor Authentication
45. Web Components standard

HOLD
46. Big enterprise solutions
47. CMS as a platform
48. Enterprise Data Warehouse
49. OSGi

PLATFORMS *continued*

Mapbox (mapbox.com) is an open mapping platform we have used on several projects. It allows a developer to quickly add a map to an application and to style the map. Mapbox can serve as an alternative to conventional mapping platforms, and it also allows for mobile friendly maps.

OpenID Connect is a standard protocol for federated identity built on OAuth 2.0. It addresses a long-standing need for a simple, web-based protocol to exchange trusted authentication and authorization information. Previous standards like SAML or generic OAuth 2.0 have proven too broad and complex to ensure universal compatibility. Our hope is that OpenID Connect can provide a useful basis for secure access to RESTful microservices with authenticated end-user identity.

Two-factor authentication significantly improves security over simple password-based systems. RFC 6238 — Time-based One-Time Password Algorithm (wikipedia.org/wiki/Time-based_One-time_Password_Algorithm) — is a standard for two-factor authentication. “Standard” authenticator apps from Google and Microsoft provide tokens to smartphone users, and there are a number of other client and server implementations readily available. With providers such as Google, Facebook, Dropbox and Evernote using **TOTP**, there really is no excuse to continue using simple password-based authentication where stronger security would be appropriate.

OSGi (Open Service Gateway initiative) is a specification that aims to remedy the lack of a module system for Java, allowing for dynamic reloading of components. While some projects (notably Eclipse) use OSGi successfully, other uses have exposed the hazards of adding abstractions to platforms never designed for them. Projects that rely on OSGi to define a component system quickly realize that it solves only a small part of the overall problem, and often adds its own accidental complexity to projects such as more complex builds. Most projects now either use old-fashioned JAR files or microservice architectures to manage components, and await the native solution in Java in the Jigsaw module specification.

TOOLS

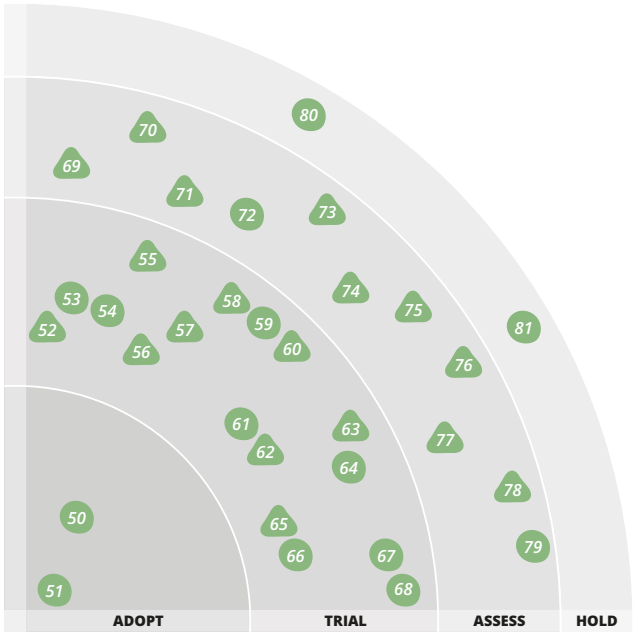
CartoDB is an open-source GIS tool built on PostGIS and PostgreSQL. It allows for storage and searching of geospatial data using SQL. It also provides a handy JavaScript library, CartoDB.js, for map styling anddata visualization.

Automated database migrations are common on agile projects, and we are happy to see advances in the tools for this space. **Flyway** makes it as painless as possible to automate changes to databases. While not as feature-rich as some competing tools, we have used it on multiple projects and appreciate its low friction.

The big Cloud providers have clearly raised the bar for provisioning, monitoring, and configuration, simplifying these tasks dramatically through powerful tools. Organizations that want to keep their compute and storage resources in-house are looking for similar solutions that work within their organizational context. **Foreman** (theforeman.org) has worked really well for us, and it is open-source software, too.

Device fragmentation in the Android world is often cited as a problem because it can be difficult to understand how your applications will behave on a large number of disparate platforms. **GenyMotion** (genymotion.com) is an emulator which can mimic the characteristics of a number of different Android devices. Our teams have found this very effective in giving fast feedback for our Android applications.

Owing to the increasing interest in Continuous Delivery and deployment pipelines, we see many teams trying to extend their Continuous Integration tooling with plugins that provide deployment pipelines at a visual level. **Go CD** is a tool that was built with the concept



of deployment pipelines at its core. Go CD has the ability to sequence workflows both sequentially and in parallel at many levels, to execute specific tasks only on certain machines as well as to deterministically promote and propagate artifacts, which is a key enabler for Continuous Delivery. These are capabilities that most Continuous Integration tools lack, and we recommend that teams who might have otherwise tried to build a deployment pipeline from their Continuous Integration server try Go CD instead. Go CD was built by ThoughtWorks, is open-source, and is available for free for all teams (go.cd). The source code is available under the Apache 2.0 license (github.com/gocd/gocd).

ADOPT
50. Ansible
51. Dependency management for JavaScript

TRIAL
52. CartoDB
53. Chaos Monkey
54. Docker
55. Flyway
56. Foreman
57. GenyMotion
58. Go CD
59. Grunt.js
60. Gulp
61. Moco
62. Packer
63. Pact & Pacto
64. Prototype On Paper
65. Protractor for AngularJS
66. SnapCI
67. Snowplow Analytics & Piwik
68. Visual regression testing tools

ASSESS
69. Appium
70. Consul
71. Flume
72. Hosted solutions for testing iOS
73. leaflet.js
74. Mountebank
75. Papertrail
76. Roslyn
77. Spark
78. Swagger
79. Xamarin

HOLD
80. Ant
81. TFS

TOOLS *continued*

Gulp is an alternative to Grunt. It is a command-line task automation tool that helps developers with SaaS compilation, autoprefixing, minification, concatenation and so on. Gulp's central idea is the use of streams, and its plugins are designed to do only one task.

We featured 'Machine image as a build artifact' in the last Radar, as an excellent way to implement fast spin-up, immutable servers. The thing holding this technique back was the difficulty in building images, especially when targeting more than one platform. **Packer** (packer.io) solves this, using your configuration management tool of choice to create images for a number of platforms including AWS, Rackspace, DigitalOcean and even Docker and Vagrant, although we have found the VMWare support to be problematic.

Consumer-Driven Contracts are a testing approach to help service interfaces evolve with confidence without unknowingly breaking consumers. The similarly named **Pact** (github.com/realestate-com-au/pact) and **Pacto** (thoughtworks.github.io/pacto) are two new open-source tools which allow testing interactions between service providers and consumers in isolation against a contract. Both have grown out of projects which are building RESTful microservices and show great promise.

Protractor is a testing framework based on Jasmine that wraps WebDriverJS with functionality specifically designed to execute end to end tests **for Angular.JS** applications. We've found it to be a standout in the rapidly evolving space of JavaScript testing frameworks. Despite being designed to run end-to-end tests with a real backend, Protractor tests can also be made to work with a stubbed HTTP gateway to run purely client side tests.

Mobile test automation is becoming increasingly important. **Appium** (appium.io) is a test automation framework which can be used to test mobile web, mobile native and mobile hybrid applications on iOS and Android. At the core, Appium is a webserver that exposes a REST API, receiving connections from a client, listening for commands, executing those commands on a mobile device and responding with an HTTP response representing the result of the command execution. It allows tests to be written against multiple platforms (iOS, Android) using the same API. Appium is open source with easy set up using npm.

Consul (consul.io) makes it simple for services to register themselves and discover other services via DNS or HTTP. It scales automatically, with service look up locally or across data centers. Consul also provides a flexible key/value store for dynamic configuration, with notification of configuration changes. The internal gossip protocol used by Consul is powered by the Serf library (serfdom.io), leveraging and building upon the membership and failure detection features.

When using techniques such as 'instrument all the things' and semantic logging, you may end up with huge amount of log data. Collecting, aggregating and moving this data can be problematic. **Flume** is a distributed system for exactly this purpose. It has a flexible architecture based on streaming data flows. With built-in support for HDFS, Flume can easily move multi-terabyte log data from many different sources to a centralized data store for further processing.

Leaflet.js (leafletjs.com) is a JavaScript library for mobile-friendly interactive maps. The library places a huge emphasis on performance, usability and simplicity, and as such works efficiently across mobile platforms and desktop browsers. It is a viable library to consider when building interactive maps for mobiles.

When testing services, we commonly need to stub out downstream collaborating services. Written by a ThoughtWorker, **Mountebank** (www.mbtest.org) is a lightweight service which you can configure via HTTP that is capable of stubbing and mocking HTTP, HTTPS, SMTP and TCP.

Papertrail is a log aggregation service that aggregates data from a variety of sources including web-servers, routers, databases and PaaS services. In addition to aggregation it provides search, filtering, and alerts and notifications out of the box. While undeniably convenient and expedient in many cases, we remain concerned about widespread adoption of services that centralize large quantities of data aggregated from multiple parties.

TOOLS *continued*

Roslyn, a .NET compiler platform under the Apache License 2.0, is a next-generation set of compilers for C# and VB.NET written entirely as managed code. It provides access to the compiler as a service and includes code analysis APIs allowing developers to access information from the compiler that was previously treated as a black box, for example syntactic and semantic models. The most immediate impact should be seen in enhancements to .NET IDEs through refactoring and code generation tools. We also expect to see improved code diagnostics and static analysis, although it will be interesting to see what the community comes up with. Meanwhile Xamarin has a Mono-compatible copy of Roslyn source code hosted on GitHub and plans to bundle Roslyn's compilers with Mono as it stabilizes, in addition to integrating the best parts into their code base.

For iterative processing such as machine learning and interactive analysis, Hadoop map-reduce does not work very well because of its batch-oriented nature. **Spark** is a fast and general engine for large-scale data processing. It aims to extend map-reduce for iterative algorithms and interactive low latency data mining. It also ships with a machine learning library.

Swagger (helloreverb.com/developers/swagger) is a standard way to describe a RESTful API so that documentation and client examples can be generated automatically. We think there's a need for some standards in this area and hope that this approach embraces Postel's law and avoids the tight-coupling and inflexibility of standards like WSDL. A number of tools are now available to render documentation and client pages from swagger-compliant descriptions (github.com/wordnik/swagger-ui).

LANGUAGES & FRAMEWORKS

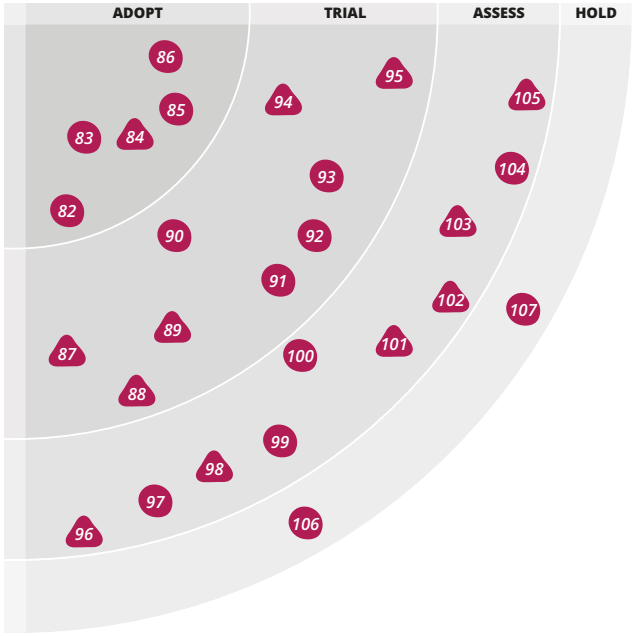
The team behind **Java 8** had to fight two battles: the community forces encouraging forever backwards compatibility (a long hallmark of Java) and the technical challenge of making a deep language change mesh with existing libraries and features. They succeeded on both fronts, breathing new life into the Java Language and placing it on par with other mainstream languages in terms of functional programming features. In particular, Java 8 has excellent syntactic magic that allows seamless interoperability between Lambda blocks, the new higher-order function feature, and SAM (Single Abstract Method) interfaces, the traditional way of passing behavior.

We continue to see JavaScript frameworks as a useful way to structure code and bring better coding techniques to JavaScript. **AngularJS** is used widely by ThoughtWorks projects. However we do advise teams to assess other good alternatives such as Ember.js and Knockout.js.

The Clojure **core.async** library allows asynchronous communication using channels, with similar syntax and capabilities to Google's Go language. The core.async library solves many common problems in an elegant way, cleaning up event callback setup and adding simple concurrency primitives. It also highlights one of the advantages of the Lisp nature of Clojure: channels add operators that are consistent with existing Clojure operators, seamlessly weaving new functionality into the language core. In addition, core.async is supported in both Clojure and ClojureScript (despite JavaScript's lack of threads), utilizing underlying platform abstractions to provide a consistent interface to both languages.

We see lots of teams creating RESTful interfaces without paying any attention to hypermedia. **HAL** (stateless.co/hal_specification.html) is a simple format for incorporating hyperlinks into JSON representations which is easy to implement and consume. HAL is well supported by libraries for parsing and representing JSON, and there are HAL-aware REST client libraries such as Hyperclient (github.com/codegram/hyperclient) which make it easy to navigate resources by following links.

Q (github.com/kriskowal/q) is a fully Promises/A+ compliant implementation in JavaScript that lets users compose promises arbitrarily deeply without the need for the deeply nested callbacks that obscure control flow. Q takes care of threading fulfilled values and rejected promises through the appropriate code paths. The space of Promises/A+ compliant libraries is currently very active with alternatives like **Bluebird** (github.com/petkaantonov/bluebird) also rapidly gaining mindshare.



ADOPT
82. Dropwizard
83. Go language
84. Java 8
85. Reactive Extensions across languages
86. Scala, the good parts

TRIAL
87. AngularJS
88. Core Async
89. HAL
90. Hive
91. Nancy
92. Pester
93. Play Framework 2
94. Q.js & bluebird
95. R as Compute Platform

ASSESS
96. Elm
97. Julia
98. Om
99. Pointer Events
100. Python 3
101. Rust
102. spray.io
103. Spring Boot
104. TypeScript
105. Wolfram Language

HOLD
106. Handwritten CSS
107. JSF

LANGUAGES & FRAMEWORKS *continued*

R is traditionally used as stand alone analysis tool by research teams. With improvements in packages like Rook and RJSONIO, it has become trivial to wrap the computational logic and expose it as an API. ThoughtWorks teams are using **R as Compute platform** to crunch large datasets in real time, using in-memory storage integrated with enterprise systems.

Elm is a functional programming language that is used to build web based user interfaces in a functional reactive style. Elm is strongly statically typed and built on the Haskell platform. Elm has a Haskell-like syntax but compiles down to HTML, CSS and JavaScript. While still in its very early days, individuals and teams interested in exploring highly interactive web based GUIs should look into this interesting little language.

Adopting the entire Clojure stack (the Clojure and ClojureScript languages, and optionally the Datomic database) offers some advantages like immutable data structures from user interface to backend. Several frameworks have appeared in the Clojure space to leverage its unique features, but the most promising so far is **Om**. Om is a ClojureScript wrapper around Facebook's React JavaScript reactive programming framework. Yet Om leverages the inherent immutability of ClojureScript, allowing automatic features like snapshots of UI state and undo. And due to the efficiency of ClojureScript's data structures, some Om applications run faster than identical ones based on the raw underlying React framework. We expect significant evolution and innovation to continue around Om.

Rust is a system programming language with modern affordances. It features a rich typing system, safe memory model and task-based concurrency. Compared to the Go language, Rust is more friendly to people who would like to write code in a functional style.

Spray/akka-http is a suite of lightweight Scala libraries providing client/server RESTful support on top of Akka. It fully embraces the Actor-, Future-, and Stream-based programming models used by the underlying platform. This lets you work on RESTful applications with idiomatic Scala code without worrying about wrapping around other Java libraries.

Spring boot (projects.spring.io/spring-boot) allows easy set up of standalone Spring-based applications. It's ideal for pulling up new microservices and easy to deploy. It also makes data access less of a pain due to the hibernate mappings with much less boilerplate code.

We are intrigued by the possibilities offered by the **Wolfram language**. Building on the symbolic approaches of the Mathematica language it also has access to a vast array of algorithms and data from the Wolfram Alpha project, which means that very succinct programs can analyze and visualize powerful combinations of real-world data.

ThoughtWorks – a software company and community of passionate individuals whose purpose is to revolutionize software creation and delivery, while advocating for positive social change. Our product division, ThoughtWorks Studios, makes pioneering tools for software teams who aspire to be great; such as Mingle®, Go™ and Twist® which help organizations better collaborate and deliver quality software. Our clients are people and organizations with ambitious missions; we deliver disruptive thinking and technology to empower them to succeed. In our 20th year, approximately 2500 ThoughtWorks employees – 'ThoughtWorkers' – serve our clients from offices in Australia, Brazil, Canada, China, Ecuador, Germany, India, Singapore, South Africa, Uganda, the U.K. and the U.S.

ThoughtWorks®