ThoughtWorks®

Technology Radar

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

JULY 2014

thoughtworks.com/radar

新增内容

下面是本期中重点突出的趋势

JavaScript 大爆发

在基于 JavaScript 的框架遍地开花之前，我们曾经认为 Ruby 在开源领域的变革速度已经够快了。当初的 JavaScript 是一种“龙套”技术，只是在其他技术无法顾及的角落中打杂。它一直这样默默“奉献”着，直到某一天，它以惊人的速度爆发成了一个“平台”。如今，要理解这一领域的广度已经极为困难，其创新速度则让人目不暇接。鉴于 Java 和 Ruby 在开源领域取得的巨大成功，我们有理由期待 JavaScript 在经过这场“大爆发”之后，也能成为一个平稳、成熟的技术平台。

康威定律

“一个组织的设计成果，其结构往往对应于这个组织中的沟通结构”，这就是康威定律，它总会在意想不到的地方发挥作用。《敏捷宣言》的一个核心理念是“个体与交互高于过程和工具”，我们发现康威定律从正反两方面支持了这一理念。有些公司受困于垂直管理结构，这给工程师们带来了不必要的阻力。更多有远见的公司则放手让开发团队打磨出自己所需的管理结构。我们既看到“漠视”康威定律所造成的危害，也看到了“拥抱”该定律所带来的好处。

微服务和 API 的兴起

我们发现人们对“微服务架构”的兴趣简直大到不可思议，而且他们都很重视 API，把它作为内部系统之间沟通的媒介，以及与外部系统之间沟通的桥梁。在微服务架构中，会部署大量非常小的服务，这些服务彼此关联以构成一个完整的系统，而且能够贴切地描述业务概念和业务价值。但是，若要充分发挥微服务架构的能量，团队必须在构建、测试、集成以及管理等方面进行良好的训练。本版《技术雷达》专门搜集了一些用于微服务的工具和技术。

再分散化

在诞生之初，互联网本是一个分布式系统，但在过去的十来年，我们却发现服务和数据的集中程度越来越高。例如，全世界超过 90% 的电子邮件是由区区 10 家提供商经手的。云计算领域也类似，少数提供商满足了我们绝大多数的云计算需求。但我们看到数据和基础设施现在又有了新的“再分散化”趋势，这一方面是因为美国对互联网基础设施的绝对控制被公之于众，另一方面也是因为越来越多的个人和组织希望自己拥有更多的控制权。

关于技术雷达

ThoughtWorks 人酷爱科技。我们对科技进行构建、研究、测试、开源、描写，并始终致力于改进 - 以求造福大众。我们的使命是支持卓越软件和掀起 IT 革命。我们创建并分享 ThoughtWorks 技术雷达就是为了支持这一使命。ThoughtWorks 中一群资深技术领导组成的 ThoughtWorks 技术顾问委员会创建了该雷达。他们定期开会讨论 ThoughtWorks 的全球技术战略和对行业有重大影响的技术趋势。

这个雷达以独特的形式记录技术顾问委员会的讨论结果，为从首席信息官到开发人员在内的各路利益相关方提供价值。这些内容只是简要的总结。我们建议您探究这些技术以了解更多细节。这个雷达是图形性质的，把各种技术项目归类为技术、工具、平台和语言及框架。如果雷达技术可以出现在多个象限，我们选择看起来最合适的象限。我们还进一步将这些技术分为四个环以反映我们目前对其的态度。这四个环是：

采用

我们强烈主张业界采用这些技术。我们会在适当时候将其用于我们的计划。

试验

值得追求。必须理解如何建立此功能。企业应该在风险可控的计划中尝试此技术。

评估

为了查明它将如何影响企业，值得作一番探究。

暂缓

谨慎研究。

自上次雷达发表以来新出现或发生显著变化的技术以三角形表示，而没有变化的技术以圆形表示。每个象限的详细图表显示各技术发生的移动。我们感兴趣的技术实在太多，远不是如此大小的文档能合理容纳的，因此我们略去了上次发表的雷达中包含的许多技术，为新技术腾出空间。略去技术并不表示我们不再关心它。

要了解关于雷达的更多背景，请参见 http://martinfowler.com/articles/radar-faq.html

ThoughtWorks 技术顾问委员会的成员

Rebecca Parsons（首席技术官）
Martin Fowler（首席科学家）
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

技术

采用
1. 前向保密
2. 隔离 DOM 并用 node 进行 JS 测试

试验
3. 显式捕获领域事件
4. 云开发环境
5. 事件溯源
6. 关注平均恢复时间
7. 人性化（微）服务注册表
8. 反向康威操纵
9. 现行 CSS 样式准则
10. 机器映像作为构建工件
11. Masterless Chef/Puppet
12. 周边企业
13. 配置测试
14. 真实用户监视
15. 没有 PUT 的 REST
16. 结构化日志记录
17. 定制服务模板

评估
18. 使用简单硬件连接物理世界和数码世界
19. Datensparsamkeit
20. 机器映像管道
21. Pace 分层应用程序战略 步进分层应用程序战略
22. 基于属性的单元测试
23. 有形互动

暂缓
24. 云提升加转移
25. DevOps 作为一个团队
26. 忽略 OWASP Top 10
27. 测试作为独立组织
28. Velocity 当成生产率

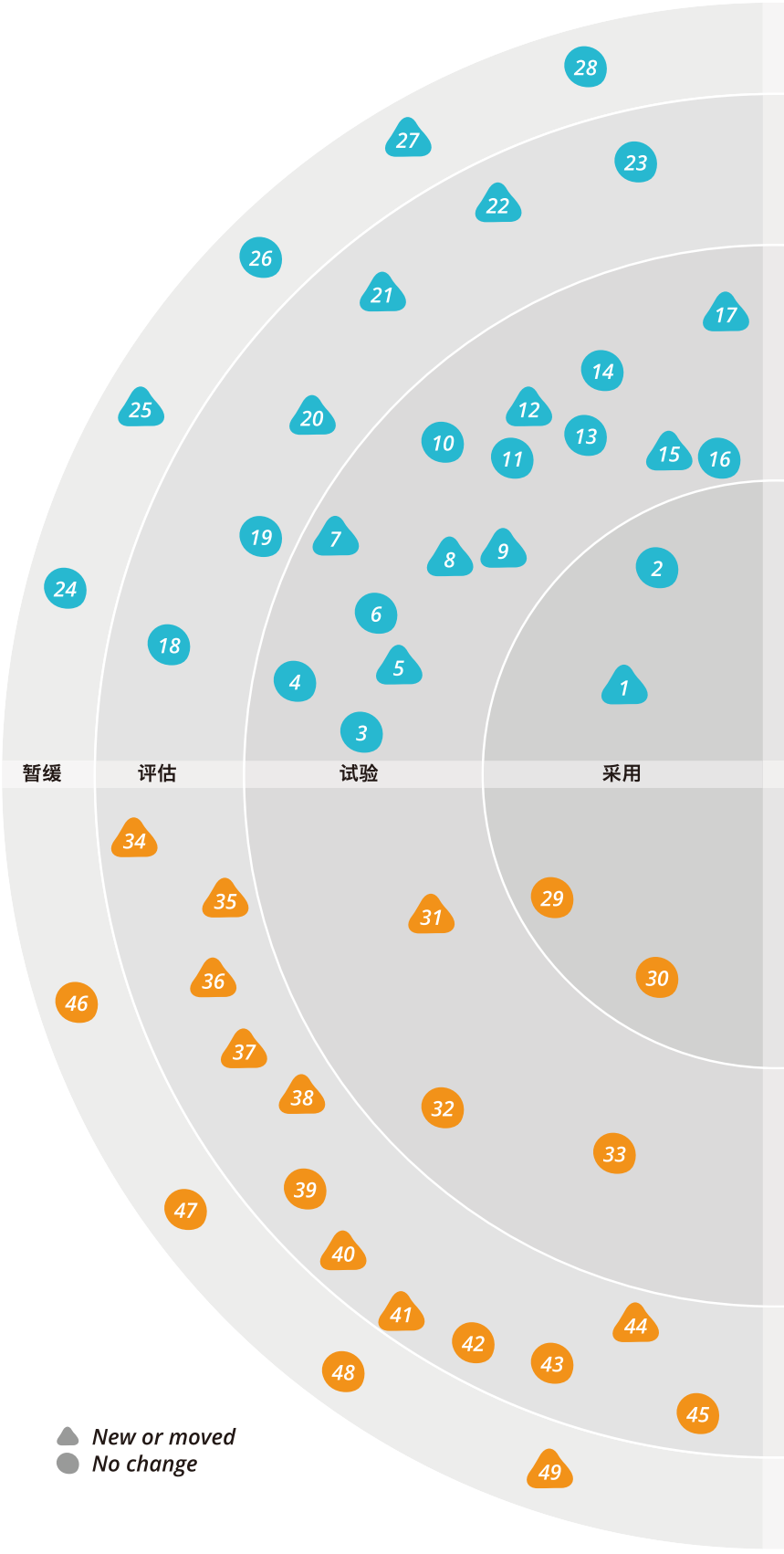
平台

采用
29. Hadoop 2.0
30. Vumi

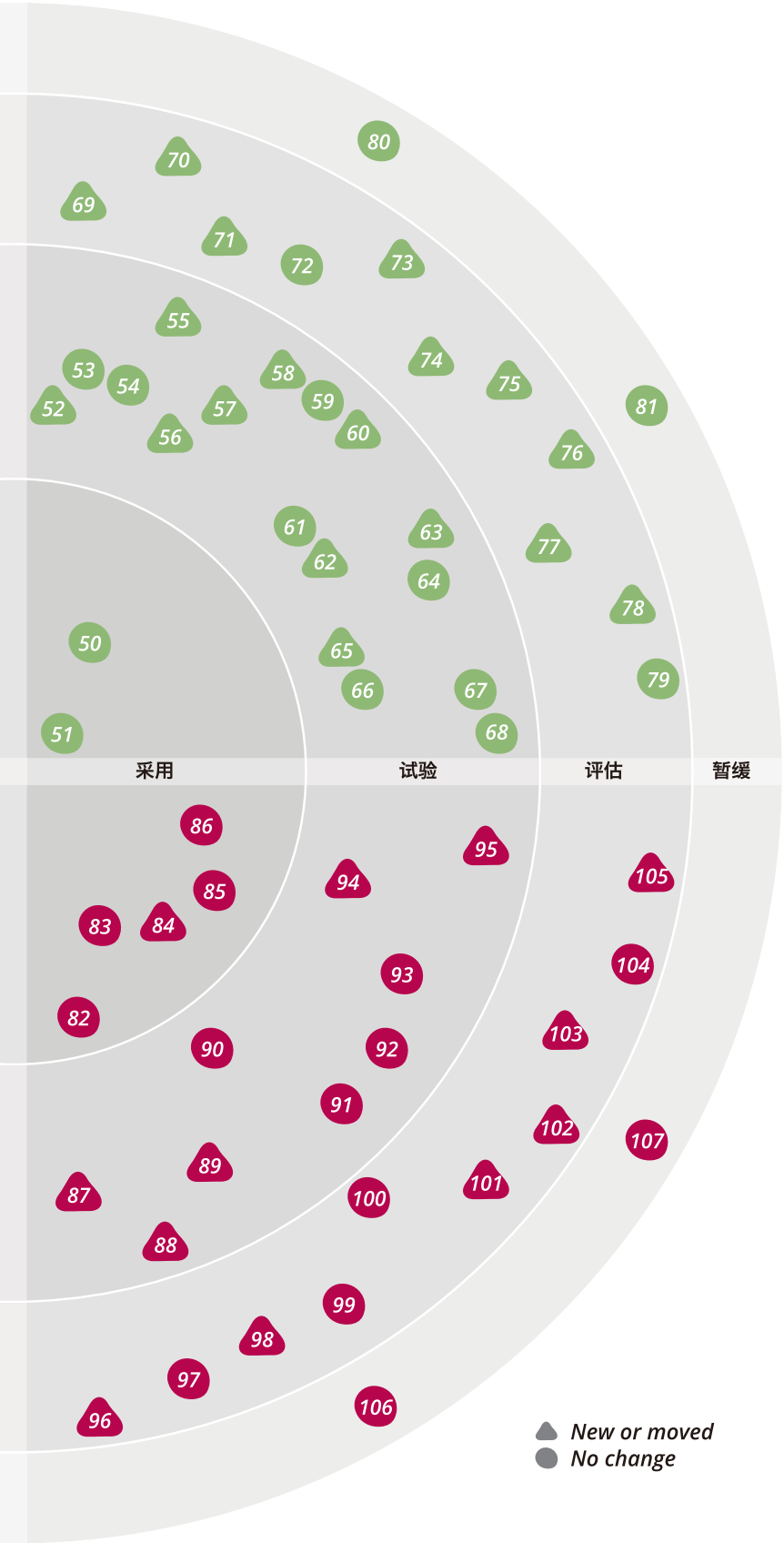
试验
31. iBeacon
32. 用于 NoSQL 的 PostgreSQL
33. 专用云

评估
34. ARM 服务器 SoC
35. CoAP
36. DigitalOcean
37. Espruino
38. EventStore
39. 低成本机器人
40. Mapbox
41. OpenID Connect
42. SPDY
43. Storm
44. TOTP 二元认证
45. Web 组件标准

暂缓
46. 大企业解决方案
47. CMS 即平台
48. 企业数据仓库
49. OSGi



THE RADAR



工具

采用

50. Ansible
51. JavaScript 的依赖关系管理

试验

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 和 Pacto
64. Prototype On Paper
65. 用于 AngularJS 的 Protractor
66. SnapCI
67. Snowplow Analytics 和 Piwik
68. 视觉衰退测试工具

评估

69. Appium
70. Consul
71. Flume
72. 用于测试 iOS 的托管解决方案
73. leaflet.js
74. Mountebank
75. Papertrail
76. Roslyn
77. Spark
78. Swagger
79. Xamarin

暂缓

80. Ant
81. TFS

语言和框架

采用

82. Dropwizard
83. Go 语言
84. Java 8
85. 跨语言的响应式扩展
86. Scala（好的部分）

试验

87. AngularJS
88. Core Async
89. HAL
90. Hive
91. Nancy
92. Pester
93. Play Framework 2
94. Q 和 Bluebird
95. R 作为计算平台

评估

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 语言

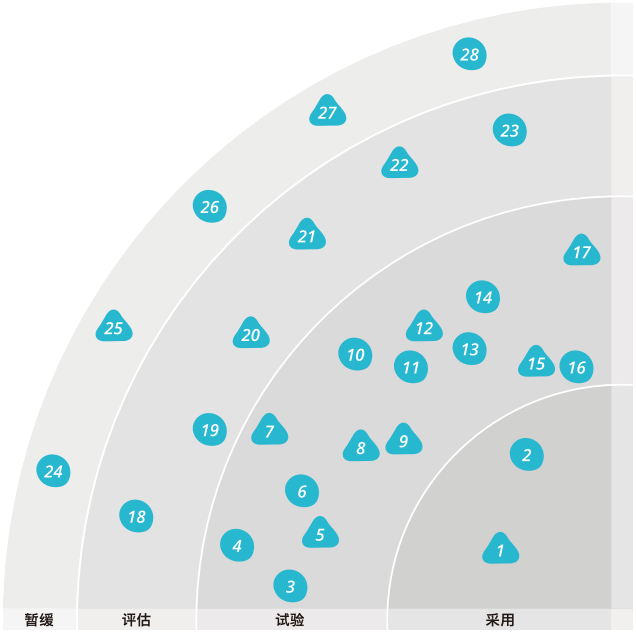
暂缓

106. 手写 CSS
107. JSF

技术

前向保密 (有时被称为完美前向保密，Perfect Forward Secrecy, PFS) 是一种加密技术，它能更好的保护当前的通讯会话：即使服务器主密钥在会话开始后被窃取了也不会威胁到当前会话。尽管为 HTTPS 连接启用对 PFS 的支持非常简单，但是众多服务器并没有按这种方式进行配置。我们建议启用前向保密来增强安全性。注意，当我们谈及“加密协议”时，一般情况下我们不喜欢用“完美”这个词 —— 因为即使是理论上最好的协议，也可能因为实现上的缺陷，比如弱随机数生成器或更先进的密码分析技术等，而遭到破解。但即使如此，启用当前可用的最佳安全协议也仍然是重要的；与此同时，也需要保持对新式攻击和新型加密协议的持续关注。

事件溯源可确保将所有针对应用程序状态的更改存储为一连串的事件记录。我们不仅能查询这些事件，还可以使用事件日志来回溯到过去的状态，并以此为基础自动调整状态以对这些“更改记录”进行妥善处理。与业务层面上的事件记录互相参考、补充，对于那些旨在提供“更宏观客户视野”的业务分析有着积极的影响。



微服务架构必然会大量增加您部署的环境中的应用程序、服务和互动的数量。我们的项目表现出对构建**人性化注册表** (http://martinfowler.com/bliki/HumaneRegistry.html)的持续关注。它汇集线上环境中正在运行的服务信息，并将其显示为一种让“人”更容易理解的形式。这些注册表自动从正在运行的服务中同步获取最新信息，而不用手工编写文档。

康威定律称，组织生产的应用程序设计必然是其沟通结构的缩影。这经常导致意外的冲突。“**反向康威操纵**”建议演进您的团队和组织结构，以促进所期望的应用程序架构。在理想情况下，技术架构将表现出与业务架构同构。

现行 CSS 样式指南是您站点上的一个页面，这个页面使用您当前的 CSS 样式，并以此作为所有当前可用的视觉元素和设计模式的参考。这种技术通过促进 UI 的“共同所有权”来将设计成果更紧密的整合进交付流程，并避免在应用程序中重复设计同一个样式。样式更改会立即在“指南”中显现，并扩散到整个站点。达到此目标的好方法之一，是采用井然有序的 SASS/LESS 文件结构，并通过语义化的设计元素来划分结构、美感和交互。

随着单页面 JavaScript 应用的激增，我们发现：浏览器中的 Ajax 调用缓慢、DOM 操纵过量以及 JavaScript 意外错误等问题，会极大地损害网站响应的友好性。从真实用户的浏览器中采集和汇总其“体验信息”非常有用。**真实用户监视**提供生产环境下的早期预警和诊断，并帮助我们把这些问题定位到特定位置。http://newrelic.com/real-user-monitoring

采用
1. 前向保密
2. 隔离 DOM 并用 node 进行 JS 测试

试验
3. 显式捕获领域事件
4. 云开发环境
5. 事件溯源
6. 关注平均恢复时间
7. 人性化（微）服务注册表
8. 反向康威操纵
9. 现行 CSS 样式准则
10. 机器映像作为构建工件
11. Masterless Chef/Puppet
12. 周边企业
13. 配置测试
14. 真实用户监视
15. 没有 PUT 的 REST
16. 结构化日志记录
17. 定制服务模板

评估
18. 使用简单硬件连接物理世界和数码世界
19. Datensparsamkeit
20. 机器映像管道
21. Pace 分层应用程序战略
    步进分层应用程序战略
22. 基于属性的单元测试
23. 有形互动

暂缓
24. 云提升加转移
25. DevOps 作为一个团队
26. 忽略 OWASP Top 10
27. 测试作为独立组织
28. Velocity 当成生产率

技术 接上页

在上期《技术雷达》中，我们介绍了“显式捕获域事件”，并着重记录由状态转换触发的业务层面的事件，而不是提供了 CRUD 操作的程序实体。REST 接口通常使用 PUT 来更新资源状态，然而，更好地方式是使用 POST 创建一个用来描述“操作意图”的新“事件资源”，比如将 PUT (changedProduct) 改为 POST (changingProductEvent) 。没有 **PUT** 的 **REST** 带来的额外好处是它分离了命令与查询接口，并能强制调用方支持“最终一致性”。

我们看到有多家组织创建了**定制服务模板**，用于快速植入新服务，进行预先配置，从而对组织的生产环境进行维护。该模板包含一组默认决策，例如，Web 框架、日志、监视、构建、打包和部署的方法。在仍然保持轻量级管理的前提下，该技术有助于鼓励通过合作进行演进。

许多部署方式都需要为不同角色的服务器提供机器映像, 比如应用服务器, 数据库服务器, 以及反向代理服务器等。由于使用操作系统 ISO 和配置脚本从零开始构建机器映像可能需要大量时间，因此**创建机器映像的构建管道**将很有用。这个构建管道的第一阶段将根据组织中的通用标准来构建基础映像。后续阶段会对基础映像逐级进行强化，从而用作不同用途。如果多个应用程序或服务的需求相似，就可以在官道上扩展出一个中间阶段。该中间阶段可以选择一个基础映像作为基准，为一个应用程序服务器（而非特定的应用程序/服务）提供一个映像。这些管道不是线性结构，而是从基础映像扩展出来的树形结构。

Gartner 的逐层步进 (**Pace-layered**) **应用策略**尝试说明如下事实：架构决策绝不是“一招鲜，吃遍天”的。相反，务必全面审视技术组合，将保守和激进的方面同时纳入考虑。尽管旨在推广 Pace 方法的某些俗套的“推荐辞”让我们有些担心，但是总的来说，我们还是推崇这个概念，并希望众多组织能够通过采用类似模型而获益。

我们重视项目的单元测试，而且推崇支持单元测试的技术（例如，**基于属性的单元测试**）。此实践运用数据生成器并基于所定义的有效范围来创建随机输入。它允许您快速检查边界状况和其他意外故障模式，而且它对多种平台的支持也正在迅速发展。

有些公司出于好意创建了独立的 **DevOps 团队**，这曲解了 DevOps 的定义。 DevOps 不是一个工作角色，而是一种旨在鼓励运维专家和开发人员紧密协作的文化运动。我们建议您不要创建这样一个独立的团队，那可能会让您吞下康威定律引发的苦果，而应该将这些技术嵌入团队，通过消除摩擦来强化反馈回路，疏通沟通路径。

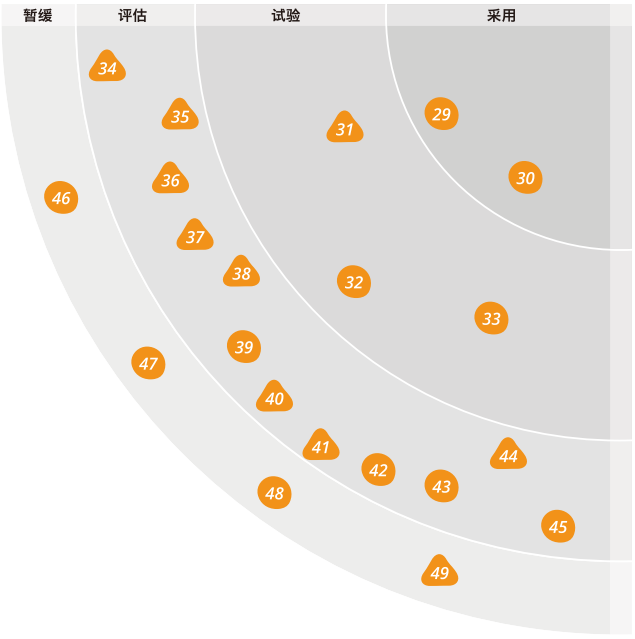
我们不断看到有些组织创建了“相互独立的”开发团队和 QA 团队。但，快速反馈是敏捷的核心教义，对项目成功至关重要。采用独立的 QA 团队会减慢反馈的速度，忘记“我们”的观念，代之以“我们和他们”的观念，让软件质量的提升变得更为困难。测试应该是一个紧密集成的活动，团队不能对其进行外包。我们建议采用“一体化团队”，让测试人员与开发人员紧密协作，而不是**将测试作为独立组织**。

平台

iBeacon是 Apple 实现的 Beacon 的“超集”，这些小设备采用低功耗蓝牙 (BLE) 为手机和其他设备提供精细的距离信息。无论人们将 iBeacon 吹捧得天花乱坠，还是诟病它所提供信息的准确性和可靠度，我们仍然认为它开启了一个有趣的契机：在特定的有限环境中，系统与用户交互的“触发点”。

AMD 最近发布了专用于服务器的 8 核 **ARM SoC**（单芯片系统），并承诺在 2015 年发布带有集成图形的 ARM SoC。基于 ARM 的服务器是 x86 的有趣替代，因为它们显然更加节能。对于某些工作任务，建议构建 ARM 驱动的云。

http://www.anandtech.com/show/7989/amd-announces-project-skybridge-pincompatible-arm-and-x86-socs-in-2015
http://www.anandtech.com/show/7724/it-begins-amd-announces-its-first-arm-based-server-soc-64bit8core-opteron-a1100



CoAP 是物联网 (IoT) 的开放式标准通讯协议。尽管目前 IoT 领域中竞争标准层出不穷，我们还是特别推荐 CoAP。CoAP 专为资源受限设备和当地无线电网络而设计，采用 UDP 进行传输，但在语义上与 HTTP 兼容。CoAP 协议融合了基于 Web 的设备模型与其自身的 URL 以及支持 RESTful 和分散方法的请求/响应范式等多项技术。

虽然 IaaS 领域已经拥挤不堪，但是还是有新的竞争者闯入了这个空间。**DigitalOcean** (https://www.digitalocean.com/) 最近以其成本、速度和简单性给我们留下了深刻的印象。如果您需要的只是基本的计算设施（Compute infrastructure），那么它值得一看。

Espruino 是一款能原生执行 JavaScript 的单片机，因此可以让众多 JavaScript 编程人员快速上手。Espurino 采用类似于 Node.js 的基于事件的模型， 因此它在保持响应的同时会非常节能。其功能略弱于 Raspberry Pi，速度略慢于 Arduino，如果您需要实现应答式行为，但可以牺牲部分原生高级特性和执行速度，那么 Espruino 将是这种低功耗环境下一个不错的选择。

鉴于事件溯源技术的流行，此领域中各种工具的日渐成熟也就不足为奇了。**EventStore** 是一种开源的函数式数据库，用于存储不可变事件和对事件流进行复杂的事件处理。与此领域的其他工具不同，EventStore 将事件流展现为 Atom 集合，因此不需要使用消息总线之类的特殊基础设施或高度专门化的客户端。
http://geteventstore.com/

Mapbox (www.mapbox.com) 是一个开放式地图平台，我们已将其用于多个项目。它允许开发人员快速将地图添加到应用程序并设置地图样式。Mapbox 可以替代传统地图平台，并可以实现“移动设备友好”的地图功能。

采用
29. Hadoop 2.0
30. Vumi

试验
31. iBeacon
32. 用于 NoSQL 的 PostgreSQL
33. 专用云

评估
34. ARM 服务器 SoC
35. CoAP
36. Digital Ocean
37. Espruino
38. EventStore geteventstore.com
39. 低成本机器人
40. Mapbox
41. OpenID Connect
42. SPDY
43. Storm
44. TOTP 二元认证
45. Web 组件标准

暂缓
46. 大企业解决方案
47. CMS 即平台
48. 企业数据仓库
49. OSGi

平台 接上页

OpenID Connect 是基于 OAuth 2.0 的用于“联盟身份管理”的标准协议。人们早就需要一种基于 Web 的简单协议，用来交换可信的认证和授权信息，而它的出现正好满足了这个需求。事实证明，以前的标准（例如 SAML 或 OAuth 2.0）都过于宽泛和复杂，难以确保普遍兼容。我们希望 OpenID Connect 能提供一个有用的基础，使人们能够以“认证过的最终用户”的身份对 RESTful 微服务进行安全访问。

相对基于密码的简单认证，**二阶段认证**显著提高了安全性。RFC 6238（基于时间的一次性密码算法）就是一种二阶段认证标准。Google 和 Microsoft 推出了“标准”认证器应用服务，用来为智能手机用户提供 token，除此之外也有许多可用的客户端和服务器端实现。既然 Google、Facebook、Dropbox 和 Evernote 等提供商都在使用 **TOTP**，那么随着人们更加注重安全性，也就没有理由去继续使用基于密码的简单认证了。

http://tools.ietf.org/html/rfc6238

http://en.wikipedia.org/wiki/Time-based_One-time_Password_Algorithm

https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2

http://www.windowsphone.com/en-us/store/app/authenticator/e7994dbc-2336-4950-91ba-ca22d653759b

OSGi（开放服务网关协议）是一种规范，旨在弥补 Java 模块系统的不足，实现组件的动态加载。尽管有些项目（尤其是 Eclipse）成功地使用了 OSGi，但另一些使用案例却暴露出一个问题：将这些抽象模型添加到那些并非为它们所设计的系统中时，可能导致危害。在依靠 OSGi 定义组件系统的项目中，人们很快就发现它只能解决整体问题中的一小部分，而它本身却经常给项目带来意外的复杂性，例如更复杂的构建流程。现在大多数项目不是使用基于 JAR 文件的老式组件，就是使用微服务架构。相比 OSGi，这些项目更期待 Java 原生的模块化解决方案，即 Jigsaw 模块规范。

工具

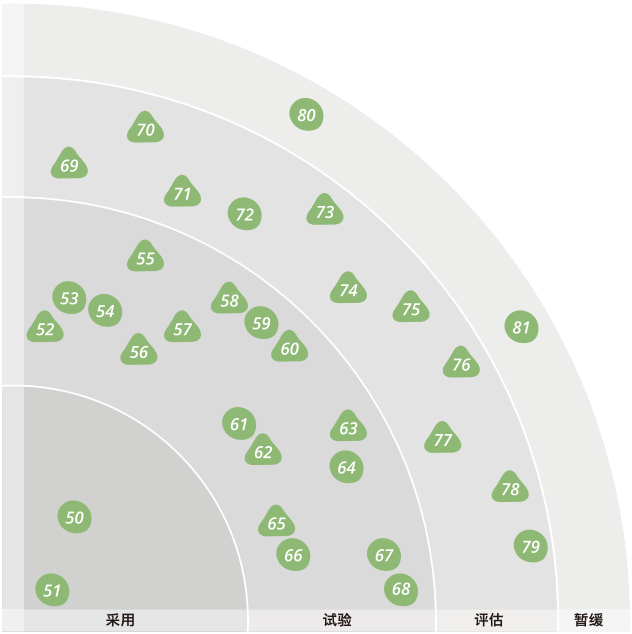
CartoDB 是一种基于 PostGIS 和 PostgreSQL 的开源 GIS 工具。该工具允许使用 SQL 来存储和搜索地理空间数据。除此之外，它还提供了一个很好用的 JavaScript 库——CartoDB.js，用于地图样式设置和数据可视化。

自动化数据库迁移在敏捷项目中已经很普遍了，该领域下的工具的进步也令人兴奋。**Flyway** 可以尽可能“无痛”地实现数据库的自动化更改。虽然功能的丰富程度还不及其它一些竞争对手，但我们已经在多个项目中使用过它。它的低冲突性令人印象深刻。

大型云提供商显然已经通过强大的工具增强了执行创建、监视和配置等任务的能力，并大大简化了操作难度。而另一方面，一些希望把计算和存储资源留在内部的组织则正在寻找适用于其组织环境的相似解决方案。**Foreman** 在这方面表现得很好，而且它是开源软件。http://theforeman.org/

在 Android 领域，设备碎片化是被经常提及的问题。 因为我们很难了解应用程序在各种不同的平台上的表现。**GenyMotion** (http://www.genymotion.com/) 是一种仿真器，可以模拟多种不同的 Android 设备的特性。我们的团队发现它能够非常有效地为我们的 Android 应用程序提供快速反馈。

随着对持续交付和部署流水线的兴趣的不断增加，许多团队试图用插件扩展其持续集成工具，以便将部署流水线可视化。**Go CD** 是一种以部署流水线为核心而构建的工具。Go CD 能够将多个级别的工作流组织成串联或并联序列，能够指定仅在某些机器上执行特定服务，还能稳定的推送和产出成品——这是持续交付的核心。这些功能都是大多数持续集成工具所欠缺的。我们建议那些希望尝试为其持续集成服务器构建部署流水线的团队使用 Go CD。Go CD 是由 ThoughtWorks 构建的开源软件，可免费供所有团队使用。http://www.go.cd/。并可在 Apache 2.0 许可证约束下



获取源代码：https://github.com/gocd/gocd/

Gulp 是 Grunt 之外的另一种选择。它是一种命令行任务自动化工具，可帮助开发人员进行 SaaS 编译、自动加前缀、压缩、连接及其他任务。Gulp 的核心理念是对“流”的广泛使用，它的每个插件都被设计成只完成一件任务。

我们在上期技术雷达中提出了使用机器映像作为构建工件。我们认它可以极大的帮助我们实现对一系列无需修改的服务器的快速上线。但是复杂的虚拟机映像的构建，尤其是针对多平台的虚拟机映像构建是该技术的制约因素。**Packer** 解决了这个问题。你可以使用配置管理工具为不同的平台生成虚拟机映像，包括 AWS、Rackspace、DigitalOcean 乃至 Docker 和 Vagrant。虽然我们发现该工具在 VMWare 平台上有较大问题但它仍不失为一款好工具。

采用
50. Ansible
51. JavaScript 的依赖关系管理

试验
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 和 Pacto
64. Prototype On Paper
65. 用于 AngularJS 的 Protractor
66. SnapCI
67. Snowplow Analytics 和 Piwik
68. 视觉衰退测试工具

评估
69. Appium
70. Consul
71. Flume
72. 用于测试 iOS 的托管解决方案
73. leaflet.js
74. Mountebank
75. Papertrail
76. Roslyn
77. Spark
78. Swagger
79. Xamarin

暂缓
80. Ant
81. TFS

工具 接上页

消费端驱动的契约测试是一种测试方法。它可以帮助服务接口在不断演化的同时不会无意中破坏消费端既有的使用方式。**Pact** (https://github.com/realestate-com-au/pact) 和 **Pacto** (http://thoughtworks.github.io/pacto/) 是两种名称非常相似的新型开源工具，他们可以依照契约对服务提供者和消费端的交互进行测试。两者都已应用于构建 RESTful 微服务的项目并展现出广阔的应用前景。

Protractor 是一种基于 Jasmine 的测试框架。它包装了 WebDriverJS 并提供了一系列专门为 **Angular. JS** 应用程序进行端到端测试的功能。我们发现它在各种快速发展的 JavaScript 测试框架中表现出众。虽然该框架是为使用真实后端的端到端测试而设计的，但 Protractor 也可使用模拟 HTTP 网关在纯客户端工作。

移动测试的自动化正变得越来越重要。**Appium** 是一种测试自动化框架，可用于测试 iOS 和 Android 上的 Web、本机和混合移动应用程序。Appium 的核心是一个 Web 服务器，它提供了一个 REST API 接收来自客户端的连接，侦听并在在移动设备上执行这些命令，并用 HTTP 响应返回命令的执行结果。它令我们使用相同的 API 就可以 针对多种平台 (iOS、Android) 编写测试。Appium 是开源的，可以使用 npm 轻松安装。(www.appium.io)

Consul 简化了通过 DNS 和 HTTP 进行服务的自注册和通过 DNS或 HTTP 查找其它服务的功能。它可自动扩展，在本地或跨数据中心查找服务。Consul 还提供用于动态配置的灵活的键/值存储，以及配置更改通知。Consul 使用的内部 gossip 协议是由 Serf 库支持的——该工具使用其分布式成员以及失效检测的功能。(http://www.consul.io/, http://www.serfdom.io/)

如果你使用了“完全工具化 (instrument all the things)”或“语义日志”之类的技术，您可能会得到海量的日志数据。收集、聚合和移动这些数据可能是比较困难的。而 **Flume** 就是专用于此目的的分布式系统。它有基于流式数据流的灵活架构。借助 HDFS 的内置支持，Flume 可以轻松地将上 T 字节的日志数据从许多不同的数据源汇集到中央数据存储器以供进一步处理。

Leaflet.js (www.leafletjs.com) 是一种对移动设备友好的 JavaScript 交互式地图库。该库特别强调性能、可用性和简便性，能够高效地在各种移动平台和桌面浏览器上工作。为移动设备构建交互地图时可以考虑使用此库。

在测试服务时，我们通常需要隔离下游协作服务。由 ThoughtWorker 编写的 **Mountebank** (http://www.mbtest.org/) 是一种可以通过 HTTP 配置的轻量级服务，能够方便的对 HTTP、HTTPS、SMTP 和 TCP 服务进行模拟。

Papertrail 是一种日志汇集服务，它从包括 Web 服务器、路由器、数据库和 PaaS 服务在内的多种源汇集数据。除了汇集，它还提供搜索、过滤、警报和通知功能。虽然在许多情况下它的便利性无可否认，但我们仍然担心广泛采用这种从多方汇集大量数据的服务会带来问题。

Roslyn 是 Apache License 2.0 下的一种 .NET 编译器平台，它是完全使用托管代码编写的下一代 C# 和 VB.NET 编译器。它允许将编译器作为服务进行访问；并包含代码分析 API，允许开发人员访问编译器生成的信息，例如语法和语义模型。这些信息以前均作为编译器黑箱的一部分而无法访问。它将首先在对 .NET IDE 的重构和代码生成工具的改进中发挥作用；我们还期待看到改进的代码诊断和静态分析功能，不过相信相关社区也能提出其他有趣的创意。另一方面，Xamarin 有一份与 Mono 兼容的 Roslyn 源代码托管在 GitHub 上，除了将最好的部分集成到自己的代码库，他们还计划等稳定以后将 Roslyn 的编译器与 Mono 捆绑在一起。

对于机器学习和交互分析等迭代处理，Hadoop map-reduce 的表现不是很好，这是其面向批处理的性质导致的。**Spark** 是一种用于大规模数据处理的快速通用引擎。它的目的是将 map-reduce 用于迭代算法和交互式低延迟数据挖掘。它还附带一个机器学习库。

Swagger 是一种描述 RESTful API 的标准方式，通过它可以自动生成文档和客户端示例。我们认为在此领域需要一些标准，希望这种方法能接受 Postel 法则，而避免 WSDL 之类的紧密耦合且不灵活的标准。现在有多种工具可用于根据 swagger 兼容的描述生成文档和客户端页面。
https://helloreverb.com/developers/swagger
http://en.wikipedia.org/wiki/Robustness_principle
http://en.wikipedia.org/wiki/Robustness_principle
https://github.com/wordnik/swagger-ui

语言和框架

Java 8 团队不得不面对两个难题：社区方面永远要求向后兼容性（这是 Java 长久以来的特征），而要进行深刻的语言级变革又必须“照顾”好现有的库和特性。他们成功解决了这两大难题，为 Java 语言注入了新的生命力，使其在函数编程特性上与其他主流语言并驾齐驱。特别是， Java 8 具有卓越的语法功能，可在 Lambda 代码块（新的高阶函数功能）和 SAM（单一抽象方法）接口（传统的传递行为方法）之间实现无缝的互操作。

我们发现 JavaScript 前端框架仍然是构造代码和将更好的编码技术运用到 JavaScript 的好办法。ThoughtWorks 的项目广泛使用了 **AngularJS**。但是我们也建议各团队评估一下其他优秀的替代选择，例如 Ember.js 和 Knockout.js。

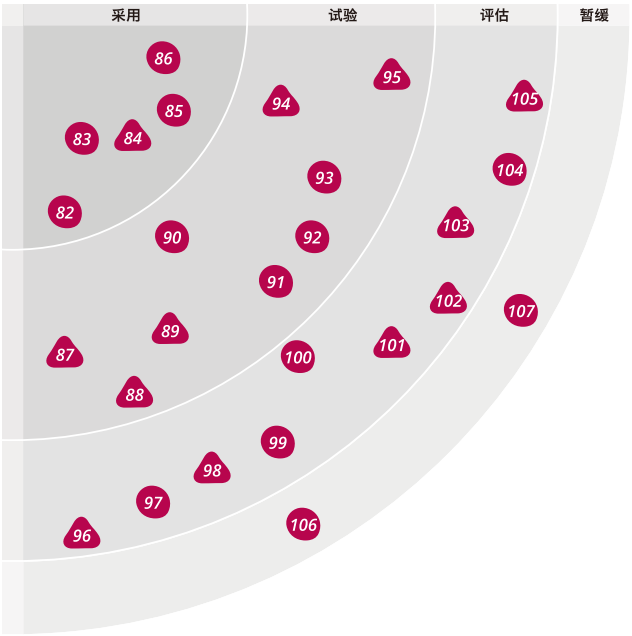
Clojure core.async 库允许使用 channel 进行异步通信，其语法和功能类似于 Google 的 Go 语言。core.async 库巧妙地解决了许多常见问题，它清理了事件回调设置并添加了简单的并发原语。它还充分发挥了 Clojure 某个 Lisp 式特性的优点：channel 添加了与现有 Clojure 运算符一致的运算符，从而无缝地将这个新功能纳入了语言核心。此外，core.async 在 Clojure 和 ClojureScript 中都得到了支持（尽管 JavaScript 缺少线程），利用底层平台抽象为两种语言提供一致的接口。

我们发现许多团队创建 RESTful 接口时都没有注意超媒体。**HAL** 是一种将超链接合并到 JSON 表示法中的简单格式，很容易实现和使用。很多库能很好地支持使用HAL来解析和展现 JSON，而且有些支持 HAL 的 REST 客户端库（例如 Hyperclient）通过访问链接方便了对资源的导航。
http://stateless.co/hal_specification.html
https://github.com/codegram/hyperclient

Q (https://github.com/kriskowal/q) 是 JavaScript 中一种完全遵循 Promises/A+ 的实现，它允许用户随心所欲地组合 promise，而不必用到会给控制流带来负面影响的深层嵌套回调。Q 可以通过相应的代码路径来传递执行成功时的值和执行失败时的值。遵循 Promises/A+ 的各种库目前非常活跃，不过以 **Bluebird** (https://github.com/petkaantonov/bluebird) 为代表的替代选择也在快速引起关注。

R 在传统上被研究团队用作独立分析工具。随着 Rook 和 RJSONIO 等软件包中的改进，包装计算逻辑并将其作为 API 展现已经没有意义。ThoughtWorks 团队正在使用 **R** 作为计算平台，在集成到企业系统的内存存储器中用于实时处理大型数据集。

Elm是一种函数式编程语言，可用于以函数式的响应风格构建基于 Web 的用户界面。Elm 有很强的静态性，以 Haskell 平台为基础构建。Elm 使用类似于 Haskell 的语法，但可以编译为 HTML、CSS 和 JavaScript。虽然这种有趣的小语言才刚刚诞生，但有兴趣探索“高度互动性 Web GUI ”的个人和团队都应该关注它。



采用
82. Dropwizard
83. Go 语言
84. Java 8
85. 跨语言的响应式扩展
86. Scala（好的部分）

试验
87. AngularJS
88. Core Async
89. HAL
90. Hive
91. Nancy
92. Pester
93. Play Framework 2
94. Q 和 Bluebird
95. R 作为计算平台

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

暂缓
106. 手写 CSS
107. JSF

语言和框架 接上页

采用整个 Clojure 堆栈 (Clojure 和 ClojureScript 语言，以及可选的 Datomic 数据库）可提供一些优势，例如从用户界面到后端不变的数据结构。在 Clojure 领域中出现了多种利用其独特功能的框架，但迄今为止最有前途的是 **Om**。Om 是以 Facebook 的 React JavaScript 响应式编程框架为中心的 ClojureScript 包装器。但 Om 利用了 ClojureScript 固有的不变性，可实现 UI 状态快照和撤消等自动功能。而且由于 ClojureScript 的数据结构的高效，一些 Om 应用程序运行起来比基于原始底层 React 框架的同类程序快。我们期待继续出现围绕 Om 的重大发展和革新。

Rust 是一种时尚简明的系统编程语言。它具有类型丰富的系统、安全的内存模型和基于任务的并发性。与 Go 语言相比，Rust 更加适用于希望以函数式风格编写代码的人。

Spray/akka-http 是一种轻量级 Scala 库套件，在 Akka 基础上提供客户端/服务器 RESTful 支持。它完全基于 Akka 提供的基于 Actor、Future 和 Stream 的编程模型，因此您可以用符合语言习惯的 Scala 代码处理 RESTful 应用程序，而不必担心围绕其它 Java 库的包装。

Spring boot 让基于 Spring 的独立应用程序实现了轻松设置。它是用来增强新型微服务的理想选择，而且部署起来很方便。由于采用了大大减少样版代码的 hibernate 映射，数据访问的困难也减少了。
(http://projects.spring.io/spring-boot/)

我们被 **Wolfram 语言**提供的可能性给迷住了。它构建于 Mathematica 语言的符号式思路的基础之上，还能访问 Wolfram Alpha 项目中各种各样的算法和数据，这意味着可以用非常简洁的程序分析现实世界中数据的强力组合并将其可视化。