JANUARY 2014

TECHNOLOGY RADAR

Prepared by the ThoughtWorks Technology Advisory Board

thoughtworks.com/radar

ThoughtWorks®

新增内容

下面是本期中重点突出的趋势：

- **生产中的早期预警和恢复** - 我们已经看到太多用于记录、监视、存储和查询运营数据的新工具与新技术。如果将这些与虚拟化和基础设施自动化带来的很短的恢复时间相结合，企业就可减少部署前需要的测试工作量，甚至可能将测试放到生产环境中进行。
- **隐私与大数据** - 虽然详尽数据收集和用于存储及分析这些数据的新工具和新平台能提供令我们激动的新业务见解，但我们也担心许多企业毫无必要地存储海量的个人数据。我们主张企业采取“datensparsamkeit”的态度，只存储来自客户的绝对最少的个人信息。
- **JavaScript 战车一往无前** - 作为一个重要的应用程序平台，以 JavaScript 为中心的生态系统还在继续发展。最近涌现了许多用于测试、构建和管理服务器端及客户端 JavaScript 应用程序相关性的有趣的新工具。
- **物理和数字的合并** - 低成本设备、开放的硬件平台和新的通讯协议正在使计算体验离开屏幕，进入我们的日常环境。用于跟踪个人生物指标的可穿戴设备的普及以及移动设备中与这类设备交互的硬件支持都是极好的例证。

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.htm

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
Darren Smith
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 捕获客户端 JavaScript 错误
2 移动设备的持续交付
3 在移动网络上进行的移动测试
4 隔离 DOM 及用于 JS 测试的节点
5 Windows 基础架构自动化

试验

6 显式捕获域事件
7 客户机和服务器用相同的代码渲染
8 HTML5 存储替代 cookie
9 监测一切
10 Masterless Chef/Puppet
11 微服务
12 周边企业
13 供应测试
14 结构化日志记录

评估

15 使用简单硬件连接物理世界和数码世界
16 协同分析和数据科学
17 Datensparsamkeit
18 云开发环境
19 关注平均恢复时间
20 机器图片作为建造工件
21 有形交互

暂缓

22 云提升和转变
23 忽略 OWASP Top 10
24 孤立指标
25 速度当成生产率

平台

采用

26 Elastic Sarch
27 MongoDB
28 Neo4J
29 Node.js
30 Redis
31 SMS and USSD as a UI

试验

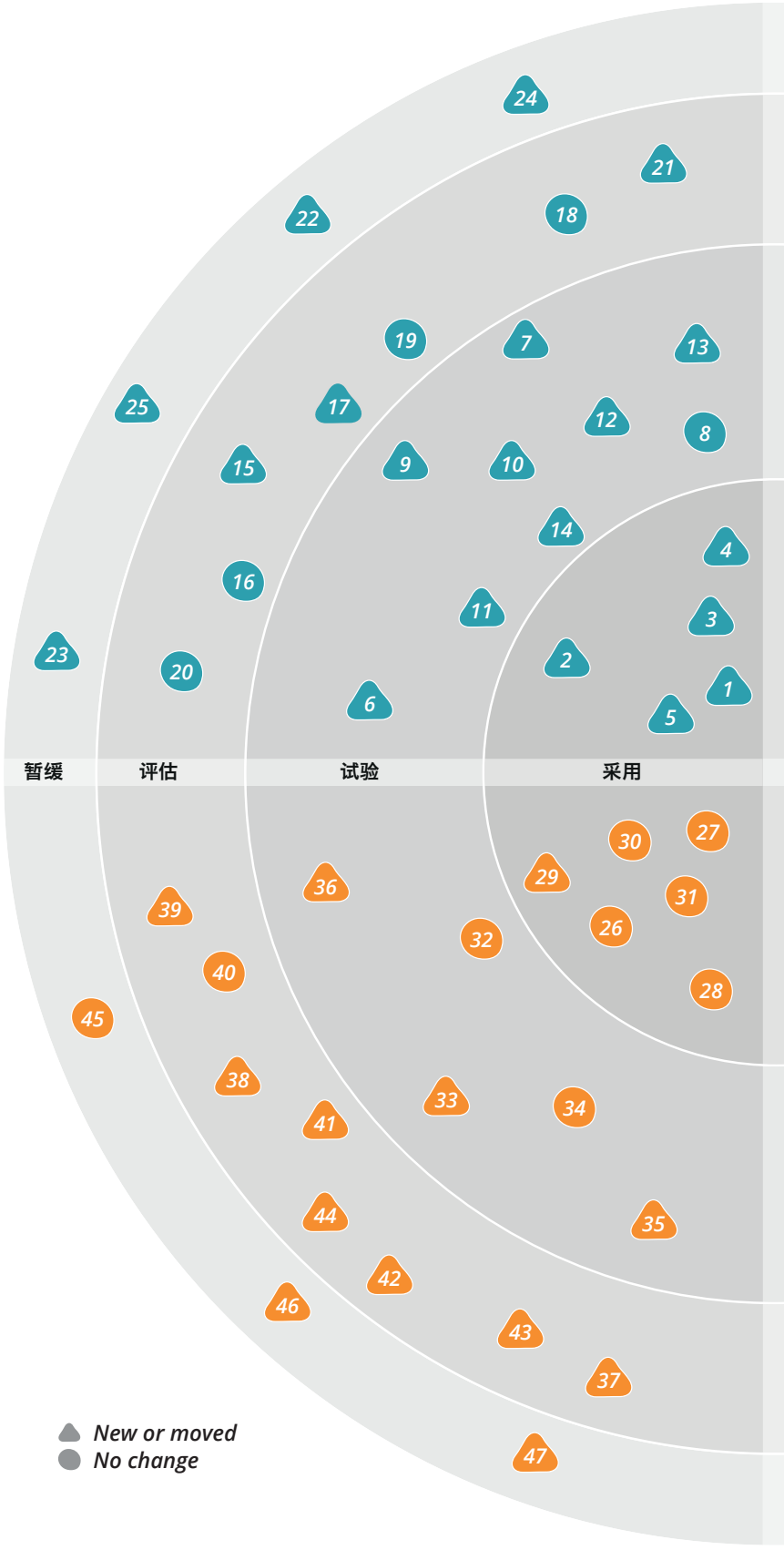
32 Hadoop 2.0
33 Hadoop-as-a-Service
34 OpenStack
35 PostgreSQL for NoSQL
36 Vumi

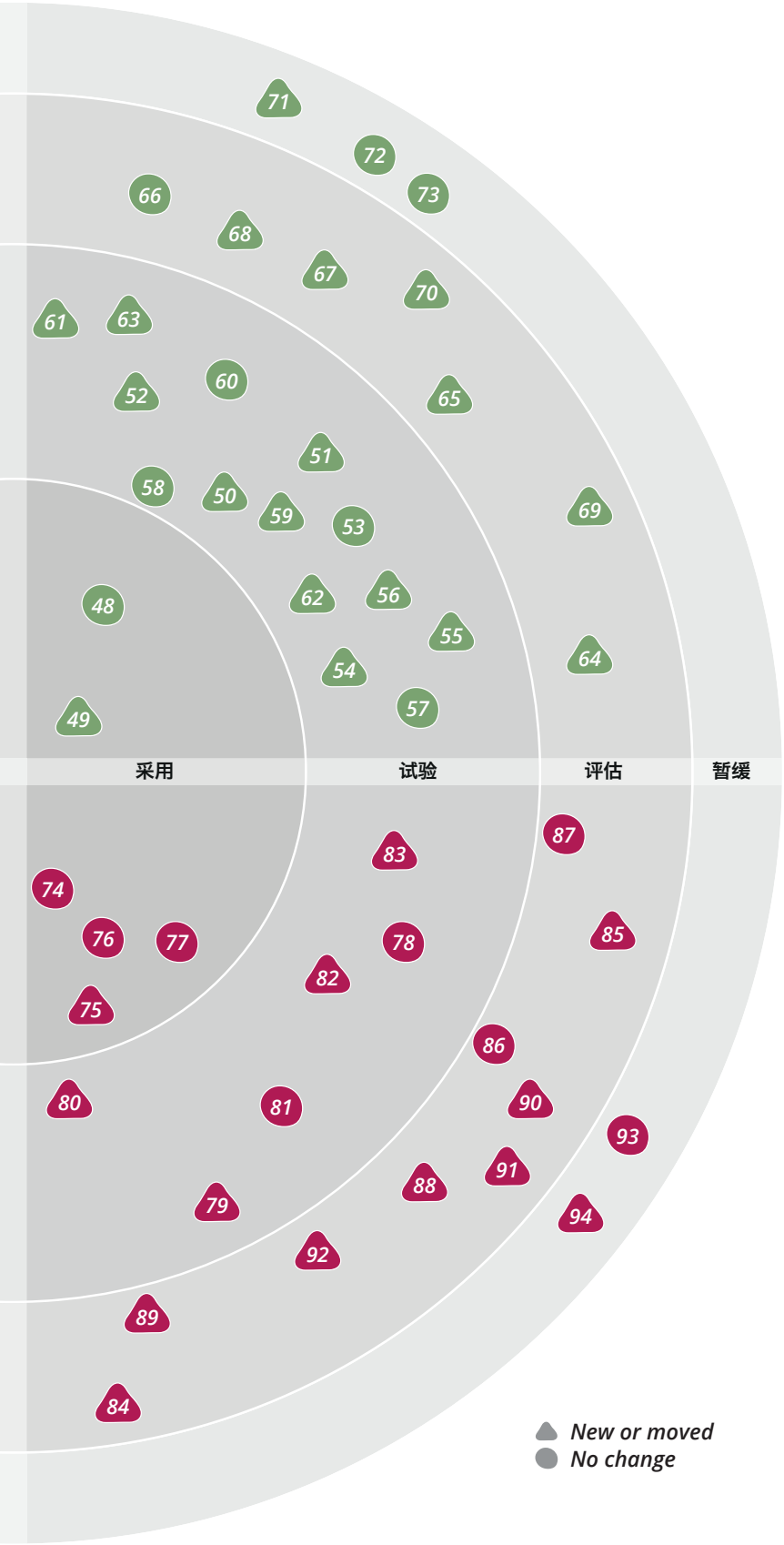
评估

37 Akka
38 Backend-as-a-service
39 Low cost robotics
40 PhoneGap/Apache Cordova
41 Private Clouds
42 Spdy
43 Storm
44 Web Components standard

暂缓

45 Big enterprise solutions
46 Centralized Data Warehouse
47 CMS-as-a-platform





THE RADAR

工具

采用
48 D3
49 Dependency management for javascript

试验
50 Ansible
51 Calabash
52 ChaosMonkey
53 Gatling
54 Grunt.js
55 Hystrix
56 Icon fonts
57 Librarian-puppet and Librarian-Chef
58 Logstash & Graylog2
59 Moco
60 PhantomJS
61 POP
62 SnapCI
63 Snowplow Analytics & Piwik

评估
64 Cloud Init
65 Docker
66 Octopus
67 Sensu
68 Travis for OSX/IOS
69 Visual regression testing tools
70 Xamarin

暂缓
71 Ant
72 Heavyweight test tools
73 TFS

语言和框架

采用
74 Clojure
75 Dropwizard
76 Scala the good parts
77 Sinatra

试验
78 CoffeeScript
79 Go Lang
80 Hive
81 Play Framework 2
82 Reactive Language Extensions
83 Web API

评估
84 Elixir
85 Julia
86 Nancy
87 OWIN
88 Pester
89 Python 3
90 Touch Events
91 TypeScript
92 Yeoman

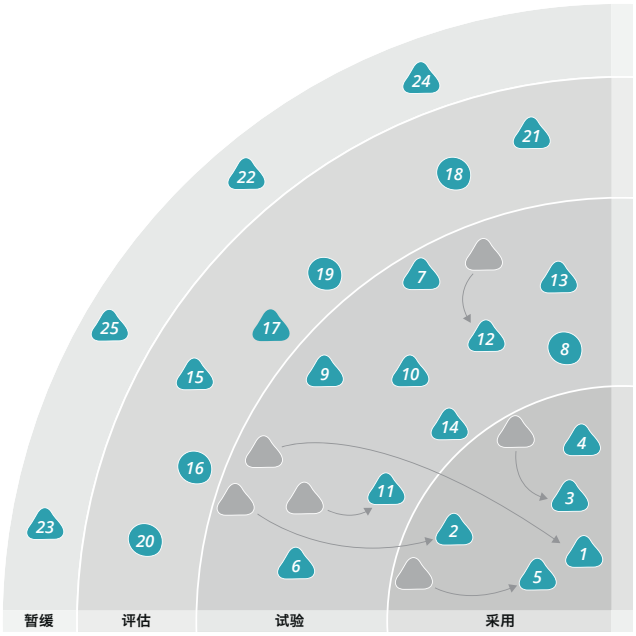
暂缓
93 Handwritten CSS
94 JSF

技术

捕获客户端 JavaScript **错误** 已帮助我们的交付团队找出了一些会影响用户体验的问题，这些问题通常只会在特定浏览器或插件配置下出现。过去一年已出现多家支持此需求的服务提供商。Web 应用程序不需在自身的数据库中存储这些错误，而可将它们直接记录到 Web 分析工具或现有的监视工具（例如 New Relic）中，以减轻存储需求。

自上次雷达发表以来，一些技术进步已使 **移动设备** 上本机 APP 的 **持续交付** 变得不那么困难。最新的开源 “改良 xcodebuild” Xctool 改进了 iOS 构建自动化和单元测试。iOS7 中新增的自动更新也减少了定期发行版的冲突。Travis-CI 现在支持 OS X 代理程序，消除了移动平台的无缝 CD 管道的又一障碍。我们在上次雷达发表时就混合方法的价值和移动测试自动化的意义提出的建议仍然有效。

随着客户端 JavaScript 应用程序日益完善，我们发现对工程完善性的需求也相应提高了。



常见的架构缺陷是从整个代码库对 DOM 的访问都不受约束，使 DOM 操纵与应用程序逻辑和 AJAX 调用混杂在一起。这会使代码难以理解和扩展。考虑分离是有用的解决思路。这牵涉到将所有文档对象模型访问（通常转换为所有 jQuery 使用）严格限制在一个瘦“隔离层”中。这种做法有一个可喜的副作用，那就是对于该 **隔离文档对象模型层** 之外的所有代码都可以使用 node.js 之类的精简 JavaScript 引擎独立于浏览器快速进行测试。

使用“监测一切”和语义日志记录之类的技术时，**显式捕获领域事件** 可能非常有用。您可以通过将这些转换作为第一级关注建模来避免猜测用户进行转换的用意。实现此结果的方法之一是使用将应用程序时间映射为业务意义事件的事件溯源架构。

HTML页面不仅在服务端生成，也越来越多地在浏览器中产生。许多情况下区分客户端和服务器模版生成仍然有必要，但随着JavaScript 模板技术越来越成熟，一种有趣的方法已经变得可行：**客户机和服务器用相同的代码生成页面。**

如果没有监测就无从获取信息以应对重要的业务事件。**监测一切** 的原则鼓励我们积极思考如何在开始部署软件时就实现这一目的。这将使我们能展现关键指标，对其加以监视，然后作出相关报告以提高运营效能。

Chef 和 Puppet 服务器版本集中管理用以变更配置的脚本，同时也是受管节点信息的中央数据库，并提供相应的权限控制。使用服务器版本的的缺点是：如果有多个客户机同时连接到它们，它们就会成为瓶颈。它们是单一故障点，需要加以整治才能变得坚实和可靠。有鉴于此，我们建议在服务器主要用于存储方法/清单时，将 chef-solo **或** standalone puppet 与版本控制系统配合使用。团队总是可以在需求出现时或者需要重复解决服务器已经解决的问题时引入这些服务器。

采用

1 捕获客户端 JavaScript 错误
2 移动设备的持续交付
3 在移动网络上进行的移动测试
4 隔离 DOM 及用于 JS 测试的节点
5 Windows 基础架构自动化

试验

6 显式捕获域事件
7 客户机和服务器用相同的代码渲染
8 HTML5 存储替代 cookie
9 监测一切
10 Masterless Chef/Puppet
11 微服务
12 周边企业
13 供应测试
14 结构化日志记录

评估

15 使用简单硬件连接物理世界和数码世界
16 协同分析和数据科学
17 Datensparsamkeit
18 云开发环境
19 关注平均恢复时间
20 机器图片作为建造工件
21 有形交互

暂缓

22 云提升和转变
23 忽略 OWASP Top 10
24 孤立指标
25 速度当成生产率

技术 接上页

我们获取和供应硬件的能力对我们的约束越来越少。但是由于这种能力为我们提供的灵活性大大增加，我们发现用于管理虚拟资产的软件资产的规模和复杂性把我们限制住了。使用在软件开发领域的技术（例如 TDD、BDD 和 CI）提供了管理这种复杂性的方法，使我们有信心以安全、可重复和可自动化的方式来改动我们的基础设施。虚拟机配置测试工具，如 rspec-puppet、Test Kitchen 和 serverspec，可用于大多数平台。

将日志作为数据处理可让我们更深入洞察自己所构建系统的运行活动。结构化日志记录就是以此技术为基础构建的，它使用包含语义信息的统一预定义消息格式，使 Greylog2 和 Splunk 等工具能产生更深刻的见解。

物理设备成本、大小、能耗的降低和简化使对软件开放物理域的设备有了爆发性增长。这些设备往往除了一个传感器和一个低功耗蓝牙或 WiFi 之类的通讯组件外没有多少部件。作为软件工程师，我们需要拓宽思路，把**使用简单硬件连接物理世界和数码世界** 考虑在内。我们已经在汽车、家庭、人体、农业和其他物理环境中看到了这种情况。由于在软件中可以实现快速迭代，制作此类设备原型机所需的成本和时间已相应减少。

我们渴望支持不断变化的业务模式，希望从过去的行为吸取经验来为每个访客提供最佳体验，因此我们总是面临尽可能多地记录数据的诱惑。与此同时，黑客比以往更加凶猛，破坏安全的大案层出不穷，而且我们现在知道政府机构也在进行规模空前的监控。Datensparsamkeit 一词来自德国的隐私权法规，它代表的理念是只存储对业务或适用法律绝对必需的个人信息。例如，可以不在访问日志中存储客户的完整 IP 地址，只存储前两个或前三个八位字节，不使用用户名记录传输历史，而使用匿名令牌。如果您从未存储信息，自然不需要担心有人盗窃它。

随着硬件和软件之间的分界线日渐模糊，我们看到传统计算越来越多地嵌入到了日常对象中。虽然互连的设备如今在零售场所、汽车、家庭和工作场所中已经普及，但我们仍然不清楚如何突破简单的玻璃屏幕将它们融入有用的计算体验。**有形互动是融合软件** 和硬件技术、架构、用户体验及工业设计的准则。目标是提供由物理对象组成的自然环境，在该环境中人可以操纵和理解数码数据。

随着云的采用率越来越高，我们不幸地看见了仅将云当作又一种主机服务提供方的趋势。一些大型供应商将现有的主机服务改换成“云”品牌，又很不幸地助长了这种 **云提升加转移** 的趋势。这些服务基本上不能提供任何真正的灵活性和随用随付的计价体制。如果您认为自己可以在不改换架构的情况下转移到云，那么您也许就没有做对。

IT 业基本上每过一周就会爆出一个损失数据、泄露密码或号称安全的系统被攻破的大丑闻。其实在软件开发期间有很好的资源可用于帮助确保人们把安全性当作头等大事考虑，我们不能再忽略它们；OWASP Top 10 就是一个很好的开始。

随着更多企业开办在线业务，我们注意到一种止步于**孤立指标** 的趋势。它们实现了特定的工具来收集和显示特定指标：有的工具用于页面视图和浏览器行为，有的用于运营数据，还有的用于整合日志消息。这导致了数据竖井以及工具之间为了收集对业务运营至关重要的业务情报的人工集成。这是一种分析域中由工具造成的分隔，损害了团队的决策能力。一种好得多的解决方案是使用集成的仪表板显示时间敏感域和团队相关信息，从而得到整合的近实时分析视图。

在各种我们不赞同的做法中，把 Velocity 等同于生产率这一做法正大行其道，因此我们感到必须在我们的暂缓环中把它明确标示出来。如果运用得当，Velocity将允许人们把“昨天的天气”合并到迭代计划流程中。Velocity 只是某个团队在某一时间的估计能力。随着团队完成磨合或解决诸如技术债构建服务器不稳定之类的问题，Velocity 可能提高。但是和所有指标一样，Velocity 可能被错误运用。例如，头脑发热的项目经理可能坚持要求不断提高 Velocity。把Velocity **当成生产率** 将导致团队实施不利于生产的行为，弃用行之有效的软件来优化这一指标。

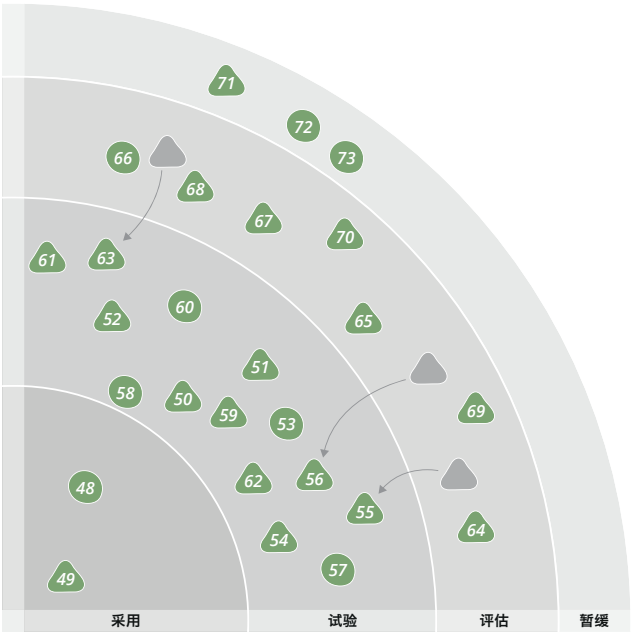
工具

我们注意到成功进行了 Hadoop 试点的组织正在先将其 Hadoop 基础设施整合为集中的受管平台，然后再将其推广到整个企业。**这些 Hadoop 即服务**平台的特点是具有与不同的核心 Hadoop 基础设置组件交联且在这些组件之间进行协调的控制层。这种平台的功能通常是通过更高级别的抽象呈现给企业的。这样的受管平台使组织有能力以相当统一的方式在整个组织中部署流程、基础设施和数据集。这些服务内置于专用数据中心和公用云基础设施中。

Akka 是一种用于在 JVM 上构建高度并行、分布和容错的事件驱动应用程序的工具包和运行时。它采用非常轻量的事件驱动模型，每 GB 内存可提供多达 270 万个处理节点。它还提供专为分布式环境而设计的、基于快速重建处理节点的容错模型。Akka 即可作为类库应用在 Web 程序中，也可以作为内核，用以构建分布式系统。

最近注重移动性的产品出现爆发式增长，强调新创意面市时间的“精益创业”做法也得到广泛运用，因而产生了**后端即服务** (BaaS) 产品的生态系统，这类产品为开发人员免去对后端的顾虑，使其可以集中精力于客户机应用程序。如果必须对新产品创意进行快速和低成本的检验，可以考虑将这些服务增加到您的工具包中。我们通常关于构建/购买/借用决策的建议仍然有效：要明确哪些功能领域对您的企业有战略意义，哪些是商品。对于有潜在战略意义的领域，务必规划出正确的迁移路径，既要让您能利用 BaaS 提供商快速起步，又要在架构发展而且出现需求时让您能够实施迁移以掌控此功能，并定制它来获得独特优势。

随着工业机器人成本下降、安全性和易用性提高，有用的商业机器人的世界正在敞开大门。Rethink Robotics 的 Baxter* 或 Universal Robotics 的 U5 之类的机器人使中小企业可以将原先人工执行的重复性任务变为自动执行。企业软件也越来越有必要将**低成本机器人**作为又一个参与者整合到价值流中。问题是如何使它们的人类同事也获得自如而且成效显著的体验。



近年来各个国家或组织内部物理存储数据的需求有了显著增长。人们很担心在云环境中托管的信息的敏感性。组织正在谋求把**专用云**作为一种备用手段，用于需要就近安置数据以便控制其访问和分发的情况。专用云提供了专供单一组织使用的云基础设施，它具有下列特点：按需自助服务、广泛网络访问、资源合用、快速弹性和可被度量的服务。

SPDY 是一种开放性网络协议，用于为 HTTP2 提议的 Web 内容的低延迟传输，现代化浏览器对其的支持已经越来越多。SPDY 确定子资源传输的优先次序，确保每个客户机只需要一个连接，从而缩短页面加载时间。SPDY 实现中使用传输层安全性配合传输报头 gzip 或 deflate 压缩格式，取代 HTTP 中的人类可读文本。它非常适合高延迟的环境。

异构化和超大数据量并不是大数据的唯一主题。在某些情况下，处理速度可能与数据量同样重要。**Storm** 是一种

采用
48 D3
49 Dependency management for javascript

试验
50 Ansible
51 Calabash
52 ChaosMonkey
53 Gatling
54 Grunt.js
55 Hystrix
56 fontsIcon
57 Librarian-puppet and Librarian-Chef
58 Logstash & Graylog2
59 Moco
60 PhantomJS
61 POP
62 SnapCI
63 Snowplow Analytics & Piwik

评估
64 Init Cloud
65 Docker
66 Octopus
67 Sensu
68 Travis for OSX/IOS
69 Visual regression testing tools
70 Xamarin

暂缓
71 Ant
72 Heavyw

工具 接上页

分布式实时计算系统。它具有与 Hadoop 类似的可伸缩性，吞吐率高达每秒一百万元组。通过它可以对 Hadoop 需要分批处理的数据进行实时处理。

在上一期雷达中，我们告诫企业不要使用在服务器端提供组件模型的传统 Web 组件框架。源自谷歌的 **Web 组件标准则大不一样**。它可帮助封装 HTML、CSS 和 JavaScript，使其不会干扰页面其他部分，也不会受到页面干扰，从而提供了创建可复用的窗口小部件的简便方法。开发人员可以根据需要随心所欲地使用框架。Polymer Project 提供了早期支持。

虽然集中集成数据进行分析和报告不失为良好的策略，但传统的**企业数据仓库** (EDW) 行动失败率超过 50%。大规模的先期数据建模导致数据仓库建设过度，这类仓库需要多年时间才能完成，而且维护成本高昂。我们在这一版雷达中要求暂缓这类老式 EDW 和技术。我们主张逐步向 EDW 方向发展。通过对仓库进行经常可以在生产中实现价值的小规模增建来测试和学习。非传统的工具和技术可以助一臂之力，例如使用 Data Vault 模式设计乃至 HDFS 之类的 NoSQL 文档存储库。

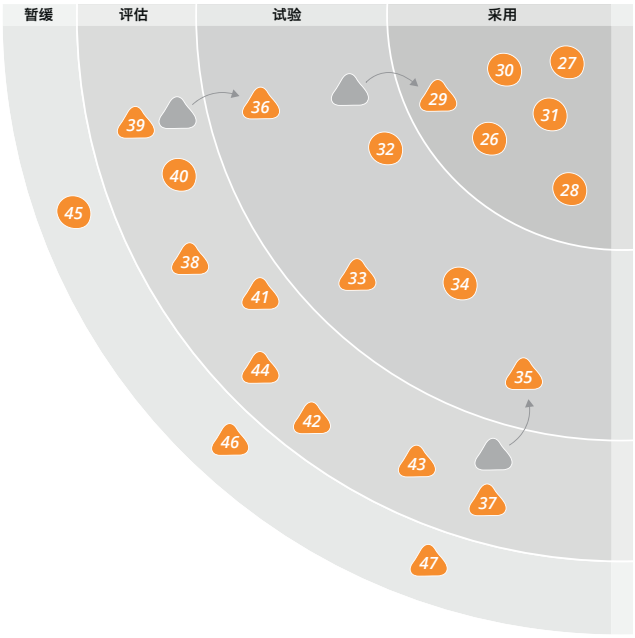
内容管理系统 (CMS) 可以有一席之地。许多情况下，从头开始编写编辑和工作流功能是不合理的。但是，当 **CMS 即平台**成为不仅仅管理简单内容的 IT 解决方案时，我们遇到过严重的问题。

平台

使用**依赖关系管理工具（针对 JavaScript）**可在运行时结构化 JavaScript 代码和加载依赖关系，从而帮助我们的交付团队处理大量的 JavaScript。尽管在大多数情况下这样可以简化工作，但是，延迟加载却使支持脱机模式变得复杂。不同的依赖关系管理工具具有不同的优势，所以，在选择时要根据自身环境。

Ansible 属于 DevOps 编排引擎一类，在 ThoughtWorks 各个项目内几乎受到普遍赞誉。它具备诸多实用工具，以及实用粒度级别的抽象性。

在移动性项目中，我们对 **Calabash** 的功能、渐进能力和成熟性印象颇深。它是一种自动化的验收测试工具，用于支持常见生态系统工具（如 Cucumber）的 Android 和 iOS 应用程序。在异构项目中，Calabash 是一项极具吸引力的选择。



我们遵循上一期雷达中提出的关注减少平均恢复时间的建议，想重点突出 Chaos Monkey（位于 Netflix's Simian Army 套件中）。**Chaos Monkey** 是一种可在正常操作过程中生产环境下随机禁用实例的工具。Chaos Monkey 在与综合监控同时运行且有某个团队随时待命时，有助于发现系统中出乎意料的弱点。这样反过来又使得开发团队能够提早建立自动发现机制，避免出现大家奋力应对措手不及的运行中断的局面。

我们开发 Node.js 应用程序的数个 ThoughtWorks 团队都在使用 **Grunt** 自动执行大多数开发活动，如缩小、编辑和 Lint 编译。许多常见任务都可以以 Grunt 插件的形式获得。如有必要，甚至可以以编程方式生成配置。

在分布式系统中管理具有依赖关系的 Web 很复杂，随着向粒度更细的微服务的迁移，这也成了更多的人所面对的问题。**Hystrix** 是 Netflix 提供的一个 JVM 库，可实现处理下游故障的模式，并提供对连接的实时监控、高速缓存和批处理机制，从而使服务间的依赖关系更加高效。与 Hystrix-Dashboard 和 Turbine 配合使用，该工具可用于建立弹性更强的系统，并提供关于吞吐量、等待时间及容错的近乎实时的数据。

对基于 HTTP 的微服务进行测试可能很痛苦且棘手。尤其是这两个场景：页面前端与微服务的通信和微服务之间的通信。应对这两个场景，**Moco** 可以说是得心应手。它是一个轻量级的存根框架，用于测试基于 HTTP 的端点。可通过两行 Java 或 Groovy 代码启动并运行嵌入式已存根服务，或通过几行 JSON 代码启动并运行独立式服务，来描述必需的行为。

长期以来，我们一直青睐使用手绘的低保真度原型来阐述用户交互，而不会纠结于图形设计的具体细节。**纸上原型**是一种工具，能通过 iOS 或 Android 上的摄像头捕捉画于纸上的单独实体模型，并将其连在一起，以便测试用户交互。这样恰好弥补了静态低保真纸的原型和更多高保真度原型设计技术之间的鸿沟。

采用
- 26 Elastic Search
- 27 MongoDB
- 28 Neo4J
- 29 Node.js
- 30 Redis
- 31 SMS and USSD as a UI

试验
- 32 Hadoop 2.0
- 33 Hadoop-as-a-Service
- 34 OpenStack
- 35 PostgreSQL for NoSQL
- 36 Vumi

评估
- 37 Akka
- 38 Backend-as-a-service
- 39 Low cost robotics
- 40 PhoneGap/Apache Cordova
- 41 Private Clouds
- 42 Spdy
- 43 Storm
- 44 Web Components standard

暂缓
- 45 Big enterprise solutions
- 46 Centralized Data Warehouse
- 47 CMS-as-a-platform

平台 接上页

在上一期的雷达中，我们提到过 ThoughtWorks 的 **SnapCI** -- 一项提供部署管道的托管服务。自那以后，我们已经见证了许多团队成功地将 SnapCI 应用于项目。如果您需要简单的云端连续交付解决方案，单击一下，SnapCI 就可为您提供。无硬件，无麻烦。

随着对数据隐私的审查持续加强，更多的公司关注与第三方分享 Web 分析方法。**Snowplow Analytics 和 Piwik** 即是两例开放代码的分析平台，可自我托管并提供前景广阔的功能集和路标。

Cloud-init 是一种简单而强大的技术，可于启动时在云实例上执行操作。与实例元数据结合使用时，Cloud-init 尤其有用，可使新启动的实例抽取所需的配置、依赖关系和软件，以发挥特殊的作用。与 Immutable 或 Phoenix 服务器模式结合使用时，Cloud-init 可创建极其敏感的轻量级机制，来管理在云端的部署。

Docker 开放代码项目在 ThoughtWorks 内部已经引发了浓厚的兴趣，势头正猛且日趋成熟。Docker 能够将应用程序打包并发布为便携式轻量级容器，而这些容器可以在笔记本电脑或产品集群上完全相同地运行。Docker 提供了创建和管理应用程序容器的工具，以及基于 LXC （Linux 容器）的运行时环境。

许多监控工具都是围绕着机器的理念所建立。而我们监控的是机器正在干什么，以及机器上运行的是哪些软件。就基于云计算的基础设施，尤其是类似于 Phoenix 和 Immutable 服务器的模式而论，该方法很有问题。机器来来去去，但重要的是业务保持运转。**Sensu** 使机器能够将自身表现为发挥特殊作用，然后 Sensu 在此基础上对其进行监控。当我们使用完机器后，可简单地将其注销。

针对 iOS 的所有部署都必须在 OS X 上执行。由于技术和许可限制，管理安装了 OS X 的服务器群既不容易，也不常见。尽管存在这些困难，受到Sauce Labs 公司支持的 **Travis CI** 目前可为 iOS 和 OS X 项目提供基于云计算的连续集成业务。

Web 应用程序的复杂度持续升高，已经强化了这一意识：不光是功能，外观也应当接受测试。由此催生了各种各样的**视觉衰退测试工具**，包括 CSS Critic、dpxdt、Huxley、PhantomCSS 和 Wraith。技术范围涵盖从 CSS 值的直截了当断言到实际的屏幕截图对照。而这一领域尚处于积极开发中，我们相信，应当将视觉衰退测试添加到连续交付管道中。

在可建立跨平台移动应用程序的各种可用选择中，**Xamarin** 提供了相当独特的工具箱。它支持将 C# 和 F# 作为主要语言，并绑定到特定于平台的 SDK，以及功能跨越 iSO、Android 和 Windows Phone 的运行时环境。应用程序编译为本地代码，而不是典型的跨平台方法，可在嵌入式浏览器中呈现基于 HTML 的 UI。这样使得应用程序的外观和感觉更加贴近本地人。使用该工具箱时，必须将特定于平台的 UI 层与其他的层分开，以确保代码可以横跨不同平台重复使用。由于包含运行时环境，应用程序二进制数会变得稍大。

我们继续见证团队将巨大的精力投入不可维护的 **Ant** 和 Nant 构建脚本。由于天生缺乏可表达性且工具提供的模块性太简洁，使得这些脚本难于理解和扩展。替代选择，如 Gradle、Buildr、和 PSake，都已经明确地证明了卓越的可维护性和生产能力。

语言和框架

Scala 因其很容易被新开发人员所熟悉而成为广受欢迎的大型语言。Scala 具有丰富多样的功能，但这并不是优点，因为其多种功能（如，隐式转换和动态类）可能给您带来麻烦。要成功使用 Scala，您需要对该语言开展研究，非常明确地知道哪些功能对您有用，以此创建您自己对 **Scala, 功能优良** 的定义。使用"功能标志"系统可禁用不要的功能。

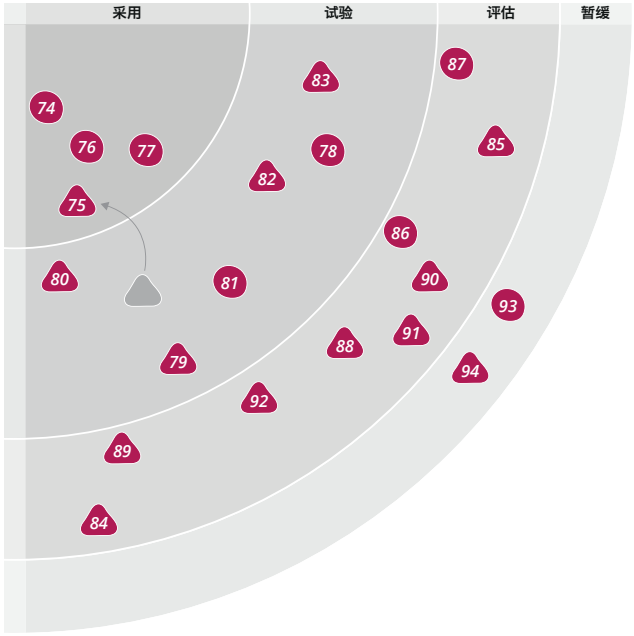
Go 语言的原始开发者是 Google，该系统编程语言可替代 C 和 C++。面世四年，Go 语言在其他领域也越发受到关注。Go 语言集小型的静态链接二进制文件和卓越的 HTTP 库于一体，在采用细粒度微服务架构的组织中颇受欢迎。

Hive 是一款构建在 Hadoop 上的数据仓库，提供类似 SQL 查询和数据定义的语言，该语言可将查询转换为可在整个 Hadoop 集群中运行的 MapReduce 作业。与其他令人满意的抽象一样，Hive 并不尝试否认 Hadoop 的底层细节，且支持定制 map/reduce 操作作为一种强大的扩展机制。尽管表面上类似于 SQL，但是 Hive 并不尝试替代关系数据系统上的低延迟实时查询引擎。我们强烈建议不要将 Hive 用于在线专门查询。

Play Framework 2 标志已引起内部热议。对于是否建议将其移至采用和保持，我们未能统一意见，分歧主要涉及其特定应用对象、应用方式和目标使用人群。虽然这些问题并非 Play 所独有，但是 Play 引起的争议远远多于常见的标准库与框架的争议。我们将重申上期技术雷达中提到的提醒信息，同时我们将监测 Play 如何日趋成熟并达到最佳状态。

响应式编程与随时间变化的流或值有关。通过数据流的元素、隐式并发和透明事件传递，这些技术实现了效率度高、延迟低的事件大规模高效处理。在上期的技术雷达中，我们提到 .NET 中的响应式扩展，Rx 在 Microsoft 的大量努力下，成为 .NET 框架的核心组成部分。此后，随着专用于 Objective C 的响应式 Cocoa 库、响应式扩展的 Java 端口、React JavaScript 库、基于 Haskell 的 Elm 语言和 Flapjax JavaScript 库的引进，我们将该标志扩展为**包含各语言的响应式扩展**。

不久之前，Microsoft 的 **Web API** 仍是通过 ASP.NET 构建 RESTful 服务的最佳选择。Web API 2 修正了大量瑕疵，对灵活路由、子资源和媒体类型支持效果更佳，且可测试性得到提高。Web API 2 依然是我们构建 .NET REST API 的首选库。



采用
74 Clojure
75 Dropwizard
76 Scala the good parts
77 Sinatra

试验
78 CoffeeScript
79 Go Lang
80 Hive
81 Play Framework 2
82 Reactive Language Extensions
83 Web API

评估
84 Elixir
85 Julia
86 Nancy
87 OWIN
88 Pester
89 Python 3
90 Touch Events
91 TypeScript
92 Yeoman

暂缓
93 Handwritten CSS
94 JSF

语言和框架 接上页

Elixir 是一款构建在 Erlang 虚拟机上的函数式同像动态编程语言，其强大的宏系统使其成为构建特定于域的语言的理想之选。Elixir 具有多项出色功能，例如，其 Pipe 操作符能够让开发人员如同在 UNIX 命令行壳层中一样构建函数管道。共享字节代码让 Elixir 能够与 Erlang 互连，在利用现有库的同时支持多种工具，例如，Mix 构建工具、Iex 互动壳层和 ExUnit 单元测试框架。在实际操作中，这是构建 DSL 时 Erlang 的替代选择。

Julia 是一款动态过程同像编程语言，旨在满足高性能科学计算的需求。该语言的实施是围绕类函数和动态方法调度的概念进行组织。Julia 程序大多是可包含不同自变量类型组合的多个定义的函数。这些语言特性的组合和基于 LLVM 的即时编译器使 Julia 的性能达到很高水平。Julia 还支持基于消息传递的多处理器环境，允许程序在多个处理器上运行。这使编程人员能够基于任何并行编程模型创建分布式程序。

PowerShell 仍然被广泛用于在 Windows 机器上实现较低级别的自动化。**Pester** 是一款帮助实现 PowerShell 命令的执行与证实的测试库。Pester 具有功能强大的模拟系统，实现了在测试中设置存根和重复操作，从而简化了开发期间的脚本测试。Pester 测试还可以整合进持续集成系统，以预防衰退缺陷。

Python 3 在先前的 Python 2.x 基础上有了很大改变，这些更改与先前版本均不兼容。值得注意的是，Python 3 竟然通过除去一些语言特性，使其易用性和一致性得到提高，同时不牺牲其功能。这已对采用造成问题，原因是某些大家赖以使用的支持库未被移植，而 Python 开发人员又常常改变操作方式。尽管如此，此次改动简化了语言，这点值得称赞。如果您已积极使用 Python 进行开发，那么 Python 3 将让您耳目一新。

几经延后，W3C 现最终确定 Touch Events 推荐标准，主要延后原因是来自 Apple 的专利请求。与此同时，更新、更广泛、更丰富的 **Pointer Events** 标准也越发流行。我们建议考虑将 Pointer Events 用于必须跨不同输入方法工作的 HTML 界面。

TypeScript 是一种将新编程语言带进浏览器的有趣方法。TypeScript 可将新语言特性编译为标准的 JavaScript，但仍作为 JavaScript 的超集，而不是全新的语言。这并不是一种非此即彼的命题，也不是将 JavaScript 推到一个中继执行平台。众多语言特性取决于 JavaScript 计划好的未来扩展。

Yeoman 致力于通过简化脚手架、构建、活动和包管理等活动，来提高 Web 应用开发人员的生产效率。Yeoman 工具系列包括 Yo、Grunt 和 Bower，这些工具需要配套使用。

在使用 **JavaServer Faces (JSF)** 的过程中，团队不断遇到困难。我们建议您避免采用此项技术。团队选择使用 JSF 似乎只是因为这是一种 J2EE 标准，并没有真正评估过该编程模型是否合适自己。我们认为 JSF 有缺陷是因为 JSF 尝试将 HTML、CSS 和 HTTP 抽象化，完全颠覆了现代 Web 框架的工作方式。与 ASP.NET Web 窗体一样，JSF 尝试在无状态的 HTTP 协议上创建有状态性，最终导致了众多与共享服务器端状态相关的问题。我们知道与 JSF 2.0 相比，该模型已有长足进步，但是我们认为该模型从根本上是个失败。我们建议各团队采用简单的框架，认识并采用 HTTP、HTML 和 CSS 等 Web 技术。

参考资料

可触式互动
http://www.interaction-design.org/encyclopedia/tangible_interaction.html
http://www.computer.org/csdl/mags/co/2013/08/mco2013080070-abs.html
http://www.theverge.com/2012/9/21/3369616/co-working-robots-baxter-home
http://robohub.org/rethink-robotics-baxter-and-universal-robots-ur5-and-ur10-succeeding/

Web组建标准
http://www.polymer-project.org

Hystrix
https://github.com/Netflix/Hystrix/wiki
https://github.com/Netflix/Hystrix/tree/master/hystrix-dashboard
https://github.com/Netflix/Turbine/wiki

各语言的响应式扩展
https://github.com/blog/1107-reactivecocoa-for-a-better-world
http://facebook.github.io/react/
http://techblog.netflix.com/2013/02/rxjava-netflix-api.html
http://elm-lang.org/
http://www.flapjax-lang.org/

Pointer Events
http://www.w3.org/TR/pointerevents/
http://www.w3.org/TR/touch-events/
http://www.w3.org/2012/te-pag/pagreport.html
http://msopentech.com/blog/2013/06/17/w3c-pointer-events-gains-further-web-momentum-with-patch-for-mozilla-firefox

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, Germany, India, Singapore, South Africa, Uganda, the U.K. and the U.S.