技术雷达

由ThoughtWorks技术咨询委员会编写

2012年10月

thoughtworks.com/radar

ThoughtWorks®

有什么新动向？以下是本版所强调的趋势。

移动--由于移动正在迅速成为人们访问互联网的主要方式，这需要在新的企业应用战略、产品战略和实施中加以考虑--从 "移动优先 "设计一直到新的测试工具。

可获得的分析--大数据不一定等于大预算。开源工具和基于云的基础设施的组合提供了一个更容易获得的分析和可视化方法。

简单的架构--简单继续获得牵引力，包括构建和组成应用程序的技术，以及基于基础设施的技术，以实现简单的部署、故障转移和恢复。这个主题对我们来说是一个反复出现的主题，但我们还没有看到我们认为必要的使用转变。

可复制的环境--支持内部托管和公共云环境的开发、测试和生产环境的标准化、设置自动化和协调管理的**工具**在本版雷达上占据突出地位。

正确的数据持久性--随着 NoSQL数据库的成熟和被接受，对使用（和滥用）模式的理解变得非常重要。

简介

ThoughtWorkers对技术充满热情。我们建立它，研究它，测试它，开放它，写下它，并不断致力于改善它--为所有人。我们的使命是倡导软件的卓越性和革新IT。我们创造并分享ThoughtWorks技术雷达，以支持这一使命。ThoughtWorks技术顾问委员会，由ThoughtWorks的高级技术领导人组成，负责创建雷达。他们定期举行会议，讨论ThoughtWorks的全球技术战略和对我们行业有重大影响的技术趋势。

该雷达记录了技术咨询委员会的讨论结果，其格式对从首席信息官到开发人员等广泛的利益相关者提供了价值。这些内容旨在作为一个简洁的总结。我们鼓励你去探索这些技术的更多细节。雷达是图形化的，将项目分为技术、工具、平台、语言和框架。当雷达项目可以出现在多个象限时，我们选择了看起来最合适的一个。我们进一步将这些项目分成四个环，以反映我们目前对它们的立场。这些环是。

- **采纳。** 我们强烈地感觉到，行业应该采用这些项目。我们在我们的项目中适当地使用它们。
- **审判。** 值得追求。了解如何建立这种能力是很重要的。企业应该在一个能够处理风险的项目上尝试这种技术。
- **评估。**值得探索，目的是了解它将如何影响你的企业。
- **保持。** 谨慎行事。

自上次雷达以来，新的或有重大变化的项目用三角形表示，而没有变化的项目则用圆圈表示。每个象限的详细图表显示了项目的移动情况。我们感兴趣的项目远远多于我们可以合理地装入这么大的文件中的项目，所以我们淡化了上一次雷达中的许多项目，以便为新项目腾出空间。淡化一个项目并不意味着我们不再关心它。

关于雷达的更多背景，见**http://martinfowler.com/articles/radar-faq.html**

贡献者 ThoughtWorks技术顾问委员会由以下人员组成。

Rebecca Parsons（首席技术官）
Martin Fowler（首席科学家）
Badri Janakiraman
Darren Smith
Erik Doernenburg
Evan Bottcher
Graham Brooks
徐浩
Ian Cartwright
James Fischer
James Lewis
Jeff Norris
Mike Mason
Neal Ford
Pramod Sadalage
Ronaldo Ferraz

萨姆
-
曼
斯科
特-
肖
斯里
哈瑞
-斯
里尼
瓦桑
Thiyagu
Palanisamy
Wendy
Istvanick

技术

采纳

1. 健康检查页面
2. Windows基础设施自动化
3. 游击式用户测试
4. 在建工程限额
5. 自动部署管道
6. 过程中的验收测试
7. 高级分析法
8. 作为文件的集合体

审判

9. Polyglot Persistence
10. 性能测试为一等公民
11. 容器外的功能测试
12. 微服务
13. 开发工作站的基础设施自动化
14. 敏捷分析法
15. 作为数据的日志
16. 响应式网页设计
17. 移动优先
18. 声明性供应
19. 远程可用性测试
20. 语义监测
21. 边缘侧 包括用于页面组成的
22. DNS中的配置

评估

23. 部署和编写测试工具的脚本

保持

24. 基于数据库的整合
25. 特征分支
26. 测试记录仪
27. 详尽的基于浏览器的测试

工具

采纳

28. 基础设施即代码
29. 嵌入式servlet容器
30. 银背
31. 应用代码
32. 茉莉花与Node.js的搭配
33. 不变的服务器
34. 石墨

审判

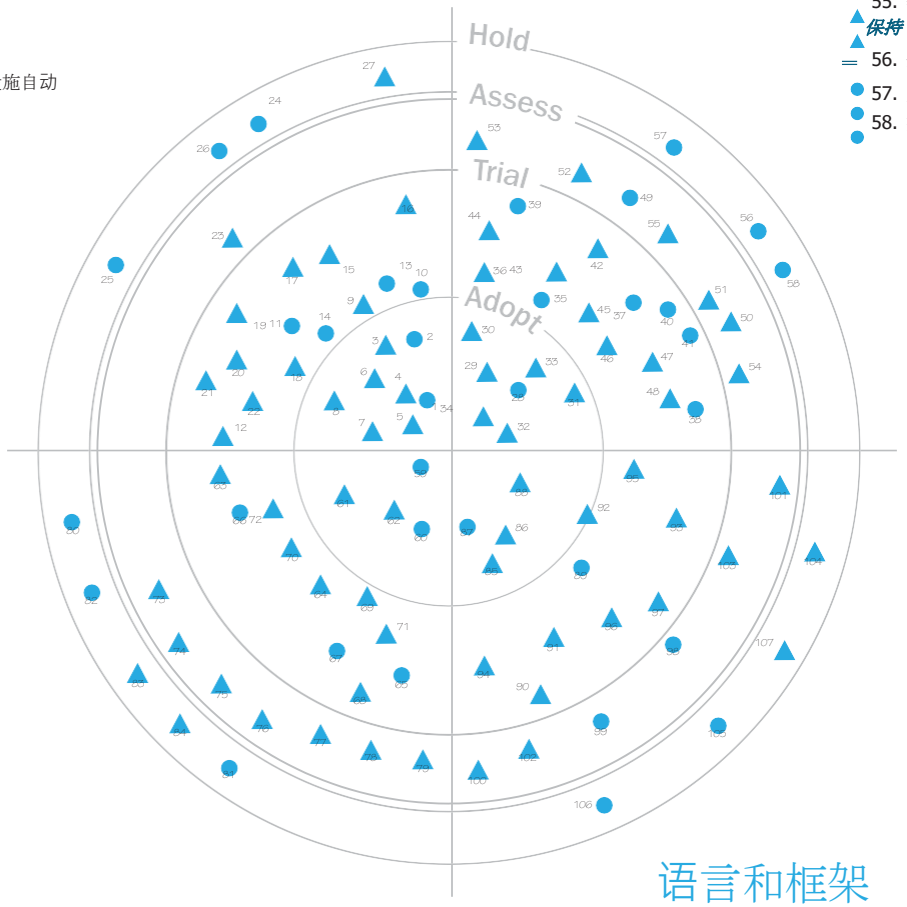
35. Vagrant
36. 渐进式
37. ÃÃÃ
38. 弗兰克
39. JavaScript微型框架
40. 翡翠
41. 淘宝网
42. 高级图表
43. D3
44. Apache Pig
45. SaaS性能测试工具
46. 依赖性结构矩阵
47. 蝗虫
48. Rake for Java & .Net

评估

49. 无逻辑标记
50. 疯狂的蛋
51. 兹普金
52. 西葫芦
53. 罐子
54. 灯台
55. 黎曼

保持

56. 企业服务总线
57. 具有隐性工作流的VCS
58. 雯雯



▲ 新的或移动的
● 无变化

平台

采纳

59. ATOM
60. 关心硬件
61. 移动支付系统
62. Neo4J

审判

63. Node.js
64. Riak
65. 特定领域的PaaS
66. Linux 容器
67. 私有云
68. 混合云
69. 梦之城_梦之城娱乐_梦之城国际娱乐_梦之城国际娱乐平台
70. 云中的持续集成
71. 辅导员（Couchbase
72. 具有异步I/O的单线程服务器

语言和框架

评估

73. 卡拉特拉瓦
74. 卫星导航系统(Datomic)
75. Vert.x
76. 蔚蓝
77. 开源IaaS
78. BigQuery
79. Windows Phone

保持

80. WS-*
81. Java门户服务器
82. 零代码包
83. 单一的基础设施
84. Meteor.js

采纳

85. Clojure
86. 领域分析
87. 关心语言
88. SASS, SCSS, LESS, 和Stylus

审判

89. 特定领域的语言
90. 划痕，爱丽丝，和科杜
91. 推特Bootstrap
92. 西纳特拉
93. AngularJS和Knockout
94. 要求.js
95. 滴水观音
96. 杰基尔
97. 用于离线应用的HTML5

评估

98. F#
99. ClojureScript
100. 吕阿
101. 润彩客网
102. 格雷姆林
103. 作为一个平台的 JavaScript

保持

104. Backbone.js
105. 存储程序中的逻辑
106. 谷歌镖局
107. 基于组件的框架

技术

我们看到，无论是在ThoughtWorks还是在更广泛的社区，**微服务**作为分布式系统设计的一种技术的采用正在上升。像Dropwizard这样的框架和像声明式供应这样的做法都表明了技术和工具的成熟性。避免通常的单体方法，并对单独替换系统部件的需要表示同情，这对系统的总成本有重要的积极意义。我们认为这对中长期的影响最大，特别是在两到五年的重写周期方面。

打破单体应用，用微服务构建系统，需要一个坚实的战略，将不同系统的输出整合成一个连贯的体验，为终端用户。在表现层使用**边际包含（ESI）进行页面整合**是一个实用而优雅的解决方案。这可以在你的环境中使用反向代理（如Varnish）或在更接近用户的内容交付网络（CDN）中进行。

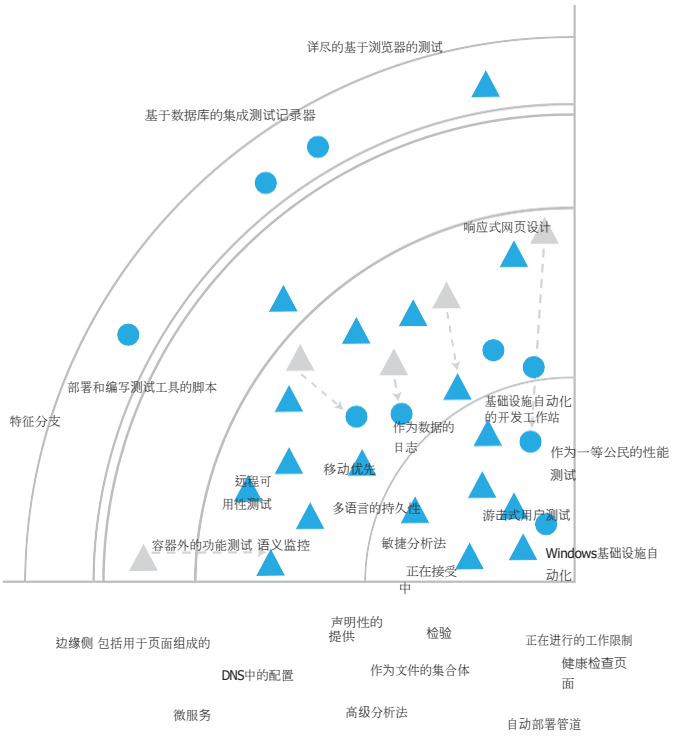
应用程序的部署经常受到过多的特定环境配置设置的影响，包括依赖服务的主机名。**在DNS中的配置**是一种有价值的技术，通过使用以下方法来减少这种复杂性标准主机名如'mail'或'db'，并让DNS解析到该环境的正确主机。这可以通过多种方式来实现，包括分界线DNS或配置搜索子域。要实现这一点，开发团队和IT运营之间的合作是必不可少的，但不幸的是，这在一些组织中仍然是困难的。

当设计一个领域模型时，聚合模式有助于增加结构和模块化。映射到关系型数据库中，聚合在表结构中是不可见的。文档数据库，如MongoDB，允许你在以下方面进行建模**聚合体作为文档**。这种1:1的映射意味着聚合根应该是从集合中加载的对象。

持续交付的采用意味着许多团队正在创建一个**自动化的部署管道**，该管道携带他们的代码一直到生产。管线允许复杂的构建链的可视化。部署活动。此外，它们还提供了以下能力可靠地跟踪构建工件，因为它们在通往生产的道路上通过每个阶段。一些供应商现在正在建立CI服务器，将流水线作为一个一流的功能来支持，而不仅仅是一个视觉元素。我们建议考虑到敏捷开发已经成为主流，我们提到这一点可能听起来很奇怪，但我们注意到团队正在重新发现和接受**工作进展的限制**。像看板这样的方法限制了飞行中的工作数量，迫使团队有更好的工作流程，并对瓶颈有更多的了解。

像Pallet这样的工具通过**声明式配置**为环境的创建和管理提供了一个引人注目的方法。通常，这是通过使用DSL声明你的环境拓扑结构--一些实例、操作系统、网络配置和应用程序，然后通过一个命令行工具自动创建整个环境。这种方法的不同之处在于实例创建和应用供应的解耦，以及增加了在多个盒子上声明特定领域应用级服务之间的依赖关系的能力。

随着部署自动化工具的成熟，包括Windows上的PowerShell，脚本越来越复杂，包含大量的逻辑。我们推荐**部署和脚本测试工具**，如PowerShell的Pester和Chef和Puppet的TOFT。围绕部署自动化的最重要方面进行良好的测试覆盖是至关重要的。



团队密切关注这些产品，以避免浪费时间，试图把管道塞进一个没有足够支持的工具中。

技术*继续*

我们正在迅速走向一个大多数消费者的互动都来自于移动设备的世界。**移动优先**通过设计首先针对移动设备的用户界面和服务器交互来拥抱这一趋势。移动优先的策略与那些假设有一个高能力的客户端设备连接到一个快速可靠的网络，然后降低体验以适应设备的限制的方法形成鲜明对比。

实现这一目标的技术之一是**响应式网页设计**。从基本的内容展示开始--通常是保持基本信息不变--增强体验以适应在更有能力的浏览器上检测到的功能。这通常采取基于屏幕尺寸的布局和格式变化的形式。

机器学习、语义分析、文本挖掘、定量分析和其他**高级分析**技术在过去的15年里已经稳步成熟。它们为预测、预报、识别可重复的模式和提供对非结构化数据的洞察力提供了难以置信的潜力。
历史上，我们存储和快速分析大量音频、视频和图像数据的能力一直受到严重限制。这对样本大小以及验证分析模型并将其投入生产所需的时间造成了限制。现在，利用一系列的新技术，如NoSQL、数据采集器、MapReduce框架和无共享商品服务器集群，我们有了
为真正有效地利用这些技术所必需的力量。再加上来自传感器、移动设备和社交媒体的全球数据的大量增加，我们认为这是一个具有巨大机会的领域。

由网络服务器、数据库、网络基础设施和后端系统产生的日志文件是企业运营和行为数据的一个宝贵来源。在过去，这些文件主要被视为故障情况下的诊断信息来源，但随着存储成本的降低，以及Splunk等工具对数百万事件的索引和检索，它们也可以成为一个来源。
的客户洞察力。将**日志视为数据**并存储完整的日志，而不仅仅是收集预定义的指标，这为回答企业以前无法预料的新问题提供了手段。

把用户带到一个受控的环境中进行正式的测试可能是一个缓慢而昂贵的提议。许多有用的、定性的反馈可以通过**游击式的用户测试**快速而廉价地收集起来--通过走到世界上，用普通公众的小样本进行测试。另一个选择是**远程可用性测试**，在那里你可以发送从线框到最终应用的所有东西，让全世界的人进行测试。Usabila, Loop11 和 Treejack
所有这些都提供了工具，你可以要求用户执行特定的任务，并捕获从完成任务所需的时间到用户在执行任务时的想法和感受等一切。

开发团队通常会产生指定和验证应用程序行为的测试，但一旦应用程序进入生产阶段就停止运行。这是一个错失的机会。**语义监控**使用您的测试来持续评估您的应用，将测试执行和实时监控结合起来。随着微服务和类似的细粒度架构方法的出现，在运行时测试它们的交互变得越来越重要。将消费者驱动的合同的验证纳入监控设施是一种方法。虽然仍在发展中，但我们看到了合并两个独立但重要的验证方案的巨大前景。

验收测试通常从 "外部 "锻炼系统，穿越整个网络堆栈，以保证锻炼整个应用程序的安全性。**过程中**的**验收测试**挑战了测试代码和被测试的应用程序必须在不同的过程中运行的概念，以实现这些好处。当使用嵌入式容器时，很容易设置系统，通过HTTP运行测试，并验证最终状态，而不需要部署到一个单独的容器并与之通信的相关设置成本。

我们以前谈到过在你的应用程序的适当层执行自动化测试。在这个雷达中，我们想说得很具体--我们建议不要进行**详尽的基于浏览器的测试**。像Selenium这样的网络浏览器自动化工具已经鼓励通过浏览器进行广泛的自动化测试。虽然这些测试在测试组合中仍有其位置，但大多数团队发现，通过浏览器执行大量的测试会产生一个缓慢而脆弱的测试套件。

工具

在我们项目中使用的所有构建工具和语言中，我们一直在寻找的是Rake。Rake是一种漂亮、简单而强大的构建语言，作为Ruby的内部特定领域语言来实现。Ruby能够在多个虚拟机平台上运行，这意味着Rake同样可用--同时为某些任务留出了利用更多特定语言工具的选项。找到一个类似的优雅和灵活性的组合是很难的，无论如何
你的平台，所以我们建议尝试**Rake for Java 和 .Net项目**。

有两件事让Ant和Maven等基于XML的构建工具产生了疲劳：太多的愤怒的尖括号和插件架构的粗糙性。虽然语法问题可以
我们认为，随着项目变得越来越复杂，插件架构严重限制了构建工具优雅成长的能力。我们觉得插件是错误的抽象层次，而更喜欢基于语言的工具，如**Gradle**和Rake，因为它们提供了更精细的抽象和更多的长期灵活性。

在JVM上运行的Ruby/Java混合应用程序中，需要处理软件包格式和依赖性解决方面的差异。通过提供一个与Ivy兼容的代理，将RubyGems打包成JAR并使用Ivy来解决Gem的依赖关系，**GemJars**整合并简化了真正的多语言代码库的构建。

云供应商所设定的先例现在正在改变企业数据中心内的期望。在云中，许多系统会自动扩展，以提供额外的可用性或应对需求的增加。对于管理不断增长的资产来说，**不可改变的服务器**，或 "凤凰服务器"，是企业寻求IaaS和PaaS的明智之举。相反，定制配置的 "雪花服务器 "增加了运营组的负担，并鼓励 "在我的机器上工作 "的心态。能够从头开始使用工具重新配置机器--硬的或虚拟的，例如
如Chef或Puppet可以大大降低管理大型服务器群的复杂性。加之软件是
设计成可以承受失败，这将导致更多可扩展和可靠的系统。

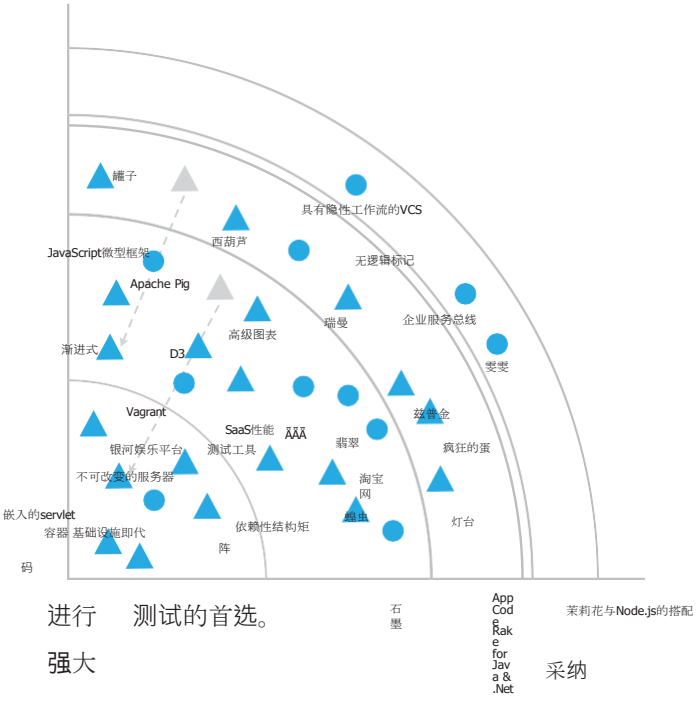
长期以来，我们一直认为JavaScript是一种一流的语言，并且一直热切地关注着测试的发展。
该领域的工具。浏览器外测试的精华目前是Jasmine。**Jasmine与Node.js配对**，是对客户端和服务器端JavaScript

当构建分布式应用以解决网络规模或大数据需求时，设置适当的监控工具成为一项非同寻常的工作。**Zipkin**是一个工具，它对基于服务的系统的不同组件进行检测，并使用 "类似Firebug "的视图，对通过多个服务的逻辑请求进行可视化分解。原始数据可以存储在Hadoop中，用于更高级的报告和数据挖掘。

Zucchini是一个测试框架，为iOS应用提供Cucumber风格的BDD测试。它使用CoffeeScript进行功能定义，在测试运行时进行截图，我们对它非常满意。

苹果的移动设备发展势头强劲，而本地应用程序是其成功的基石。自从JetBrains推出**AppCode**，一个用于iOS和OS X开发的IDE，复制了他们用于其他平台的IDE的优势，编写这些原生应用变得更加愉快和富有成效。

像大多数优秀的软件开发者一样，我们谨慎地选择我们的工具。我们特别热衷于有趣的脱离常规的做法，这就是为什么我们帮助支持**Light Table**的Kickstarter项目。虽然仍处于开发初期，但所承诺的交互性可与Smalltalk世界的精华相媲美，并带有现代色彩；我们急切地想知道将会发生什么。这个雄心勃勃的项目。



工具*继续*

Hadoop仍然是开发分布式数据处理应用程序的最流行的框架。虽然用Java编程Hadoop应用程序并不特别困难，但设计高效的MapReduce管道确实需要大量的经验。**Apache Pig**通过提供一种叫做Pig Latin的高级语言和一个执行运行时，简化了Hadoop的开发。Pig Latin是程序性的，提供了一个类似于SQL的界面来处理大型数据集。执行基础设施将Pig Latin编译成一个优化的MapReduce程序序列，在集群上运行。Pig Latin是可通过不同语言（如Ruby、JavaScript、Python和Java）的用户定义函数进行扩展。

有几个可用性测试工具与我们喜欢的 "游击队 "方法相匹配。在设计引人注目的用户界面时，眼球追踪一直是一种有用的技术，但是与之相关的设备和软件都很昂贵，而且通常需要使用专业公司。**疯狂的蛋**是一种更便宜的纯软件解决方案，它可以根据鼠标的移动产生热图。这种运动与注视有很强的关联性，可以作为一个合理的近似值使用。**Silverback**在测试期间不仅捕捉屏幕，而且还记录用户的脸和声音。这对于与更广泛的开发团队分享丰富的测试经验是非常宝贵的。

虽然有许多工具用于显示系统监控的图表，但**Graphite**最近已成为这一领域的明显领导者。它能够实时绘制指标图，具有一个轮回数据库，能够存储长时间的历史数据，同时还能以更高的保真度提供更多的最新信息。仪表盘上有许多配置选项，所生成的图表可以嵌入到网页中，以提高可见度。

Riemann是一个开源的服务器，它可以实时地聚合和转发事件。它用Clojure编写，并以Netty为基础，能够处理每个节点成千上万的并发连接。Riemann使用一个简单的Protobuf协议来处理事件，这使得它可以汇总从CPU和内存使用到下单到错误率的所有信息。它转发到Graphite等系统，触发电子邮件警报，并提供一个仪表板来监控这些指标。Riemann是将数据作为通用事件流进行实时处理的运动的重要组成部分，而不是针对不同类型的数据使用专门的系统。

性能越来越好的JavaScript引擎，加上对HTML中嵌入的SVG文档的广泛支持，使得基于纯JavaScript的客户端图形和可视化解决方案获得了很大的发展。**Highcharts**是我们遇到的最好的解决方案之一，它开箱即用，支持多种高度可配置的交互式图表类型，并且能够轻松渲染大型数据集。

D3是一个JavaScript库，用于将数据集绑定到DOM中，然后声明性地转换文档以创建丰富的可视化--从图表到热图。由于支持HTML、CSS和SVG，以及一个可扩展的插件模型，我们喜欢这个库使我们能够以更直观的方式传递信息。

我们非常赞成代码库的可视化技术。特别是，**依赖结构矩阵**（DSM）已经被证明是非常有用的，尤其是在支持进化结构和突发设计方面。对DSM的工具支持是很普遍的。

我们已经谈了很多关于**嵌入式Servlet容器的**问题--这些容器现在在我们的项目中被广泛采用。像SimpleWeb和Webbit这样的工具进一步采用了简单的嵌入式方法，提供了原始的HTTP服务器功能，而不需要实现Java Servlet。
规范。我们高兴地看到，利用这一点，测试代码的复杂性也相应减少。

我们坚信在线自动化性能测试，尽管到目前为止，这一领域的开源工具还很有限。**Locust**是一个深受喜爱的工具，它提供了用Python编写测试的能力，对运行多个注入器有很好的支持，还能生成基本的统计数据，以及一个有用的网络仪表板。它的网络负载测试方法更注重用户的模拟，而不仅仅是生成每秒的点击率。我们通常会推荐Locust，而不是像JMeter或Grinder这样的旧工具。

我们看到**SaaS性能测试工具**（如Blitz.io和Tealeaf）的兴起，而不是为性能测试的许可证和设置机器集群而纠结。这些服务使我们能够很容易地与大量不同地域的客户一起进行性能测试，而不需要在基础设施上投入大量资金。

平台

混合云结合了公共云和私人数据中心的最佳特点。它们允许应用程序在正常时期在私人数据中心运行，然后在公共云中使用租用的空间，以获得溢出容量。
流量高峰期。现在有一些基础设施解决方案，允许在混合云中自动和一致的部署，如Palette和RightScale。随着亚马逊、Rackspace和其他公司提供的强大产品，我们正在将混合云转移到本版雷达的 "试用 "中。

从几乎令人困惑的选项中选择合适的云计算供应商仍然是困难的。一种策略是采用**开源的IaaS**平台，如OpenStack或CloudStack。这允许你运行一个与公共云一致的私有云，并在需要时从一个云供应商迁移到另一个。更进一步，Apache的Deltacloud抽象出特定的供应商API，以提供跨云平台的一致体验。

谷歌的**BigQuery**将数据分析带到了云端。BigQuery允许你上传数据，并通过类似于SQL的特别查询来调查数据集，而不是将数据加载到昂贵的数据仓库中，并使用预定义的索引。这
是创建廉价概念验证甚至完整应用的好方法，因为数千台服务器对数百吉字节的数据的处理在几秒钟内就能完成。

微软的**Azure**云平台继续在追赶AWS等更成熟的云，但我们对微软如何应对市场需求印象深刻。与大多数微软解决方案一样，它仍然是一个竞争者，值得评估。

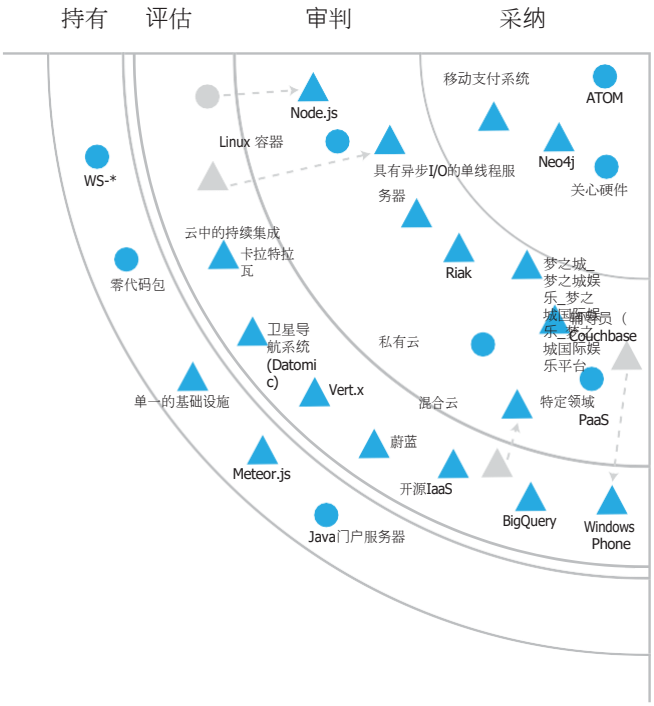
云中的持续集成是支持敏捷开发的那些事后可见的基础设施产品之一。由于没有本地软件和最小的配置，它只是工作。有了成熟的产品，严肃的开发者就没有借口回避这一重要的做法了。

尽管在全球北方有明显的阻力，但肯尼亚的M-Pesa等**移动支付系统**正在提供安全的无现金货币交易。随着该服务在非洲各地的推广，该系统为数百万拥有手机但缺乏传统银行网点的人打开了市场。Square等供应商正在慢慢改善这一状况，但北方地区仍然滞后。

对于符合文档数据库模型的问题，**MongoDB**提供了简单的可编程性、查询界面、自动故障切换的高可用性以及自动分片功能。它允许从RDBMS模型顺利过渡到NoSQL数据存储，并包含了熟悉的概念，如定义索引的能力。

图形数据库将信息存储为由命名关系连接的任意相互连接的节点，而不是表和连接。图式数据库没有模式，可扩展性强，是在复杂领域建立半结构化数据模型的绝佳选择。**Neo4j**是这个领域的领跑者--它的REST API和Cypher查询语言都支持简单而快速的存储和图的遍历。

Riak是一个分布式的键值存储，是无模式的，与数据类型无关的。它可以很好地用于写入量大的项目，以存储会话、购物车和流式日志等数据，同时它还保留了在全文搜索中执行复杂查询的能力。分布式集群可以在没有单一主站的情况下自我恢复，具有可调整的一致性。
和可用性设置，并能在需要时进行碰撞检测和解决--所有这些在高可用性环境中可能特别有帮助。



平台 *继续*

对数据库工作方式的根本性反思，**Datomic**是一个不可变的数据库服务器，具有迷人的交易和部署特性。敏捷项目中常见的头痛问题之一是管理数据库迁移，尤其是恢复以前的状态。Datomic使迁移的需求不复存在--每一个版本的数据（和模式）都是如此。是由数据库保存的。虽然仍在不断发展，但我们欣赏Datomic的大胆设想。

Couchbase是一个具有自动分片功能的持久性缓存，无主集群和复制数据以避免缓存错过。因为它支持Memcached协议，所以它可以直接替换基于Memcached的系统。

Vert.x代表了从传统的、独立的应用程序容器的又一次演变，它是一个连接同步和异步编程风格的应用程序框架。这让程序员可以选择用可扩展性和性能来换取简单性。与Node.js不同，Vert.x是一个库，可以从JVM上支持的各种语言中调用，包括Java、Ruby和JavaScript。

我们以前一直对可重用代码跨平台工作的说法持怀疑态度。我们对许多工具的经验
我们建议那些正在寻找这些类型的解决方案的客户谨慎行事。我们认为**Calatrava**是一个值得评估的移动应用开发方法，它能在这些危险的水域中谨慎地航行。该框架整齐地遵循业务和表现逻辑的分离，在有共性的地方最大限度地重用，在需要遵循速度或设备特定习惯的地方提供本地访问。

Meteor.js是一个客户端和服务器端的JavaScript应用框架，在网络浏览器内或Node.js容器中运行，并由MongoDB支持持久性。它使用 "智能包"--可以在浏览器中运行或作为云服务的一部分的小的代码包。它允许热代码部署和浏览器内的实时更新。我们认为这个想法很好，尽管这个框架还没有准备好投入使用。

尽管**Windows Phone**有一个很好的开端，但一个良好的
在所有的移动平台中，我们看到了微软及其合作伙伴在执行平台战略方面的一些挫折。这使我们对该平台的未来没有上次雷达时那么乐观。

有时，架构决策会导致你加入一些你只能负担得起的基础设施项目，如大型机或搜索设备。这是一个可怕的想法。它严重地限制了测试和部署的灵活性。
我们强烈支持你可以轻松设置和拆除的基础设施。**单子基础设施**属于过去被误导的供应商驱动的架构。

语言和框架

JavaScript正在走出浏览器，成为跨平台开发的一项重要技术。在Node.js、Meteor.js和Calatrava等移动框架所采取的代码重用方法中，它处于中心位置。伴随着最近可以编译成JavaScript的其他语言的激增，这让我们怀疑我们是否应该开始将**JavaScript视为一种平台**，而不仅仅是一种语言。

随着应用的不断扩大，许多JavaScript代码库的规模也在不断扩大。为了提高代码的模块化程度并帮助管理，我们看到团队正在接受**Require.js**这样的库。使用异步模块定义（AMD）格式，代码被分割成模块，简化了开发和维护，然后由一个优化工具对脚本进行组合和缩减，用于生产部署。

随着JavaScript开发的兴起，人们对可重复使用、可扩展的UI工具的需求越来越大。**Twitter Bootstrap**以该领域的最佳产品为基础，提供了一套强大的模式和组件，帮助开发者创建具有愉悦审美的响应式和自适应应用程序。

我们认为激发下一代人的热情是至关重要的。
技术专家。**Scratch、Alice和Kodu**是依靠视觉环境和积木作为教学设备的编程语言。它们为打算在学术界以外的环境中培养编程知识的教育项目和组织提供了令人兴奋的可能性。

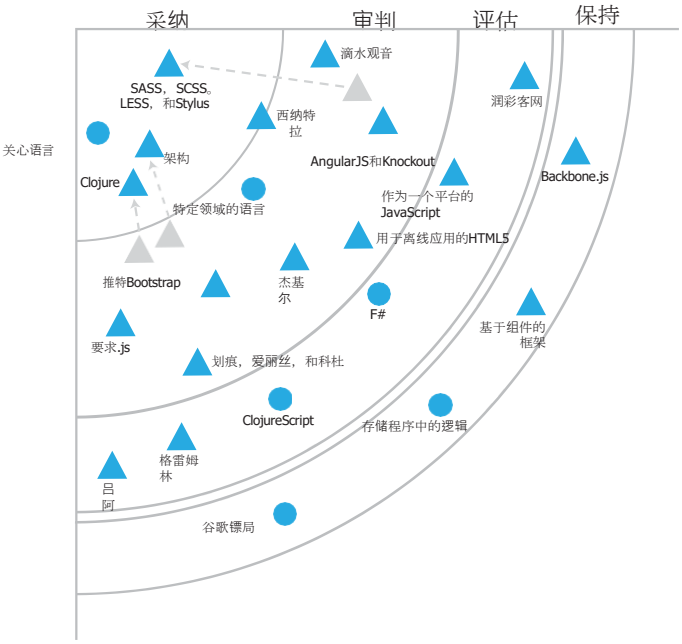
作为编程语言领域的一个不太可能的竞争者，**Lua**已经在各种行业中得到了大规模的应用。它被用作游戏开发和音乐创作的脚本平台；被嵌入到销售点设备和网络设备中；并被用于扩展具有安全执行语义的NoSQL数据库。我们期望在未来的时间里有进一步的增长。

微框架作为处理客户端和服务器端应用程序中日益增长的复杂性的一种方式正在出现。
Sinatra是服务端领域这一趋势的早期先驱之一，它暴露了一个轻量级DSL，以构建可以轻松组成的快速服务。Flask、Scalatra和Compojure分别是Python、Scala和Clojure的类似产品。

Dropwizard是几个轻量级的Java工具和框架的意见组合，其中许多工具和框架都值得提到它们自己的权利。该软件包体现了许多我们最喜欢的技术，包括一个嵌入式HTTP服务器，支持RESTful端点，内置的操作指标和健康检查，以及直接的部署。Dropwizard让你很容易做正确的事情，让你专注于一个问题的基本复杂性，而不是管道。

Gremlin是一种由多个图形数据库支持的命令式图形遍历语言。它的简明结构可以代替数据库的本地语言，从而导致更快的开发时间，在某些情况下，可以更快地执行。我们建议在简单的情况下将其作为一个很好的选择。

Jekyll代表了网络发布领域的框架的 "微观化"。虽然重点保持在做一件事上--以博客为特色的网站--尽可能地透明，但它也显示了通往更轻量级的未来的道路。我们喜欢的一个例子是，现在为你的软件项目发布有用的文档是非常简单的。



语言和框架*继续*

介绍一个用于开发iOS的Ruby编译器和工具链

在ThoughtWorks的开发社区中，**RubyMotion**不出所料地引起了很大的轰动。在那里在构建应用程序时，仍然需要了解底层的iOS APIs和一些Objective-C，但对于那些认为使用Ruby语言和工具更舒适的人来说，有明显的好处。

有一种趋势是将离线功能的需求等同于建立一个应用程序的需求。尽管标准化进程缓慢，但大多数HTML5功能现在已经在所有主要浏览器上实现。它的本地存储功能，全面支持移动和平板电脑浏览器--使**HTML5用于离线应用**成为非常合适的选择。

我们看到了一种创建单页网络应用的常见模式。这些应用程序不需要完整的页面刷新，而是从服务器上请求较小的数据集，并通过修改DOM来改变其页面的显示内容。为了使其更易于管理，已经开发了支持数据绑定、客户端模板和验证的JavaScript MV*框架。虽然轻量级的应用程序可能不需要一个框架，对于更复杂的场景，**AngularJS**和**Knockout**应该被认为是这个领域目前的领跑者。

Backbone.js是一个很好的例子，它的抽象性被推得太远。虽然我们一开始很喜欢它的易于连接性，但在实践中，它和所有这样的数据绑定框架（从WebForms到客户端/服务器工具）一样，存在着同样的问题。我们发现它过于模糊了框架和模型，迫使人们做出错误的架构决策，或者为了保持理智而精心制作框架黑客。

随着行业从桌面GUI开发转向网络，将最成功的模式和设计移植到新的范式上似乎是很自然的。经过15年的尝试，我们觉得仍然没有**基于组件**的**框架**成功地实现了这一点。我们建议不要试图将Web开发变成它根本不是的东西。现在是时候接受网络的页面和基于请求的本质了，并专注于支持--而不是反对--这些概念的框架。

参考文献

Alice http://alice.org
Apache Pig http://pig.apache.org
Calatrava http://calatrava.github.com
D3 http://d3js.org
Dropwizard http://dropwizard.codahale.com
光桌 http://kodowa.com
蝗虫http://locust.io
Lua http://lua.org
黎曼
http://aphyr.github.com/riemann 划痕**http://scratch.mit.edu** 银背
http://silverbackapp.com Zucchini
http://zucchiniframework.org

ThoughtWorks是一家全球性的IT咨询公司

ThoughtWorks - 定制软件专家。一个完全致力于定制软件的艺术和科学的公司。我们制造软件，并使我们的客户在这方面做得更好。我们的底线是快速、可预测地设计和交付软件。做企业规模的软件是很困难的，但对于那些能够按期交付的组织来说，回报是巨大的。

ThoughtWorks的产品部门通过其由Mingle®、Go™和Twist®组成的Adaptive ALM解决方案™，提供管理整个敏捷开发生命周期的工具。ThoughtWorks在澳大利亚、巴西、加拿大、中国、德国、印度、新加坡、南非、乌干达、英国和美国设有办事处，拥有2,100名专业人员，为客户服务。

我们在快速、可靠和高效的定制软件开发方面处于行业领先地位。
当你需要一个专家伙伴来帮助你在竞争中取得胜利并保持领先时，请联系我们。
http://www.thoughtworks.com/contact-us