Introduction

The ThoughtWorks Technical Advisory Board consists of a group of senior technical leaders within ThoughtWorks. They produce the ThoughtWorks Technology Radar to help decision makers understand emerging technologies and trends that affect the market today. This group meets regularly to discuss the global technology strategy for ThoughtWorks and the technology trends that significantly impact our industry.

The Technology Radar captures the output of these discussions in a format that provides value to a wide range of stakeholders, from CIOs to enterprise developers. With this in mind the content provided in this document is kept at a summary level, leaving it up to the reader to pursue more detailed knowledge as the need arises.

The goal of the radar is conciseness, so that its target audience understands it quickly. To that end, it is graphical in nature. However, terseness requires extra context; thus, there are some aspects that warrant further explanation. The first is the groupings (or quadrants) that radar items are placed within: techniques, tools, languages and platforms. In a number of cases a single radar item could appear in multiple quadrants, but we have tried to map each item to the quadrant that is most appropriate.

The titles given to each concentric circle also require clarification: hold, assess, trial and adopt. The placement of a radar item in one of these circles is intended to map our current position on the item.

Hold: when placed in this band, the item may be of interest to ThoughtWorks and others in the industry. However it is our opinion that the item is not ready to invest significant time and resources in which to build experience.

Assess: a technique, tool, language or platform that moves into the assess band of the radar is something that we believe is worth exploring with the goal of understanding how it will affect the technology impacted dimensions of your enterprise.

Trial: having established a radar item as something worth pursuing, it is important to understand how to build up this capability. Enterprises should look to trial the technology on projects that have a risk profile capable of taking onboard a new technology or approach.

Adopt: is the final stage that is of interest to us on the radar. Here we feel that the industry has begun to move beyond the trial phase and has found the proper patterns of usage for an item. An item may also appear in the adopt band if we feel strongly that the industry should be adopting a radar item now, rather than going through a more gradual adoption approach.

As we look at each quadrant in detail, we try to show the movement that each item has taken since we last compiled this information. Given that there is a gap of almost a year in capturing our positions on the radar, a number of items have come from off the radar and into the trial and adopt bands rapidly. We expect that this will occur less often as the radar is released more regularly.

Contributors

The ThoughtWorks Technical Advisory Board is comprised of

- Rebecca Parsons (CTO)
- Martin Fowler (Chief Scientist)
- Scott Conley (CSO)
- Ian Cartwright
- Erik Doernenburg
- Jim Fischer
- Neal Ford
- Ajey Gore
- Wendy Istvanick
- Mike Mason
- Cyndi Mitchell
- David Rice
- Pramod Sadalage
- Chris Stevenson
- Jim Webber
- Hao Xu

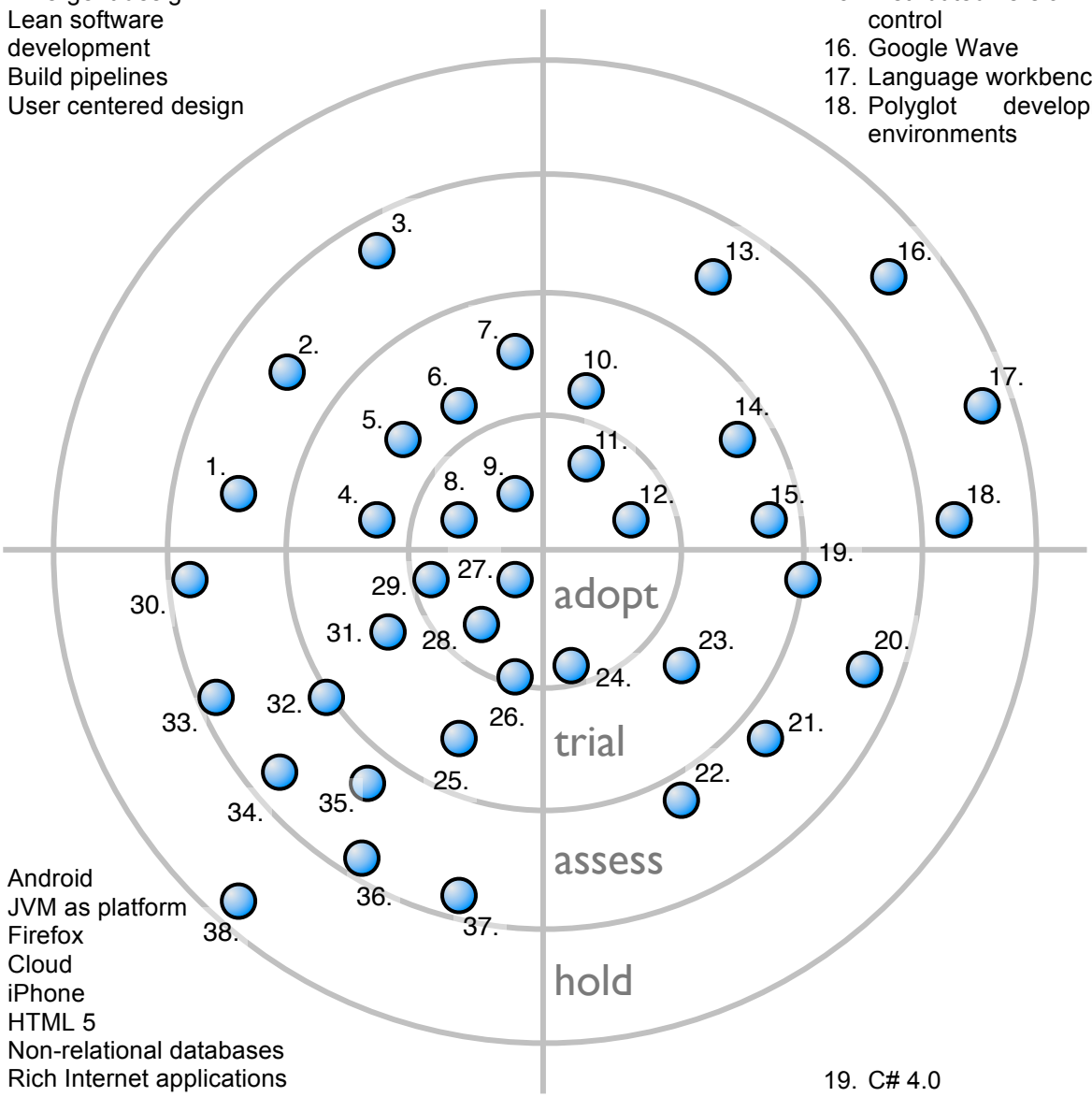
with technical assistance provided by Darren Smith.

techniques

1. Continuous deployment
2. Incremental data warehousing
3. Evolutionary architecture
4. Evolutionary database
5. Web as platform
6. Emergent design
7. Lean software development
8. Build pipelines
9. User centered design

tools

10. Visualization & metrics
11. IE6 end of life
12. ASP.NET MVC
13. Next-generation test tools
14. Subversion
15. Distributed version control
16. Google Wave
17. Language workbenches
18. Polyglot development environments

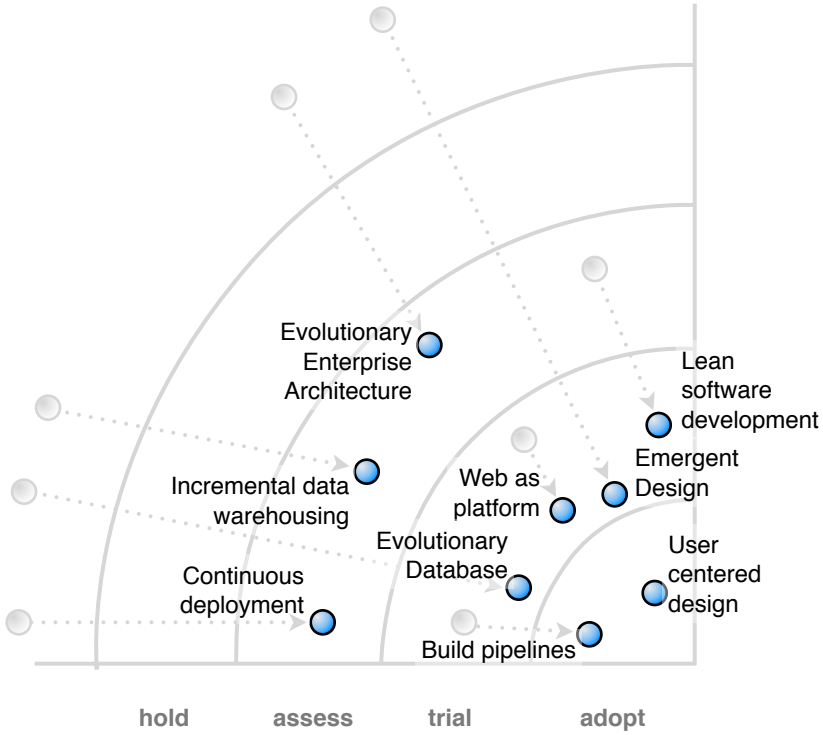


platforms

25. Android
26. JVM as platform
27. Firefox
28. Cloud
29. iPhone
30. HTML 5
31. Non-relational databases
32. Rich Internet applications
33. RDF & SPARQL
34. Google as corporate platform
35. Location based services
36. Chrome OS
37. Chrome
38. IE8

languages

19. C# 4.0
20. Java language end of life
21. Functional languages
22. Concurrent languages
23. DSL's
24. Javascript as a first-class language



Techniques

The past 2 years or more has seen a proliferation of continuous integration tools and platforms leading to substantial innovation in the build and release space. Distribution of builds is one such innovation and yet another is the way in which builds are now structured to make greater use of automation in various stages of the build. **Build pipelines** help to provide greater insight into the quality of each build and the environments to which they have been deployed. A natural expansion of the build pipeline meme is the adoption of **continuous deployment** techniques, where the intention is to extend the build pipeline into the production environment. This relies on automated deployment techniques and authorization mechanisms built into the continuous integration toolset. One of the key benefits is the ability to move new functionality into production rapidly and reliably.

We assist many of our clients in adapting enterprise software architecture practices to fit within an Agile software delivery approach. In the past year we have seen increased interest in **evolutionary enterprise architecture** and how service oriented architectures shape the boundaries between enterprise units. The value of an evolutionary approach to enterprise architecture is the creation of lighter weight systems that ease integration between disparate parts. By embracing this approach and the notion of the **web as an enterprise application platform,** we have reduced overall complexity of application architectures, increased quality and scalability, and reduced development costs.

The industry has seen significant changes to the way we use and store data over the past few years. Agile development practices have lead to greater emphasis on **evolutionary database** design, requiring new tools that support migration of schemas in line with changes to the domain model of an application. As storage space consistently becomes cheaper and data access speeds increase, many organizations are investigating the use of multiple schemas to hold data for different purposes, e.g. transactional and analysis schemas. **Incremental data warehousing** is becoming increasingly popular as the cost of moving data between a transactional data store and an analysis environment is less than the value of having access to near real-time reporting of critical business data.

As Agile practices move further toward mainstream adoption, we see significant benefits from the adoption of **Lean software development** practices as well. These practices have their roots in the Toyota Production System and complement much of our understanding of Agile software development to date. One topic that Lean has also given us greater insight into is that of set-based design. Set-based design leads us to implement similar solutions at the same time while the cost of doing so is constrained. This leads us into the area of **emergent design** and the ability to let experience shape our design decisions and defer key decisions until the last responsible moment.

The benefits of **user-centered design** are often understated. Gaining a broader understanding of data flows and users' goals simplify the overall architecture of a system while optimizing user interaction. In the past year we have seen a greater uptake of user-centered design in Agile software development practices as experts in both fields have established new ways of working together.

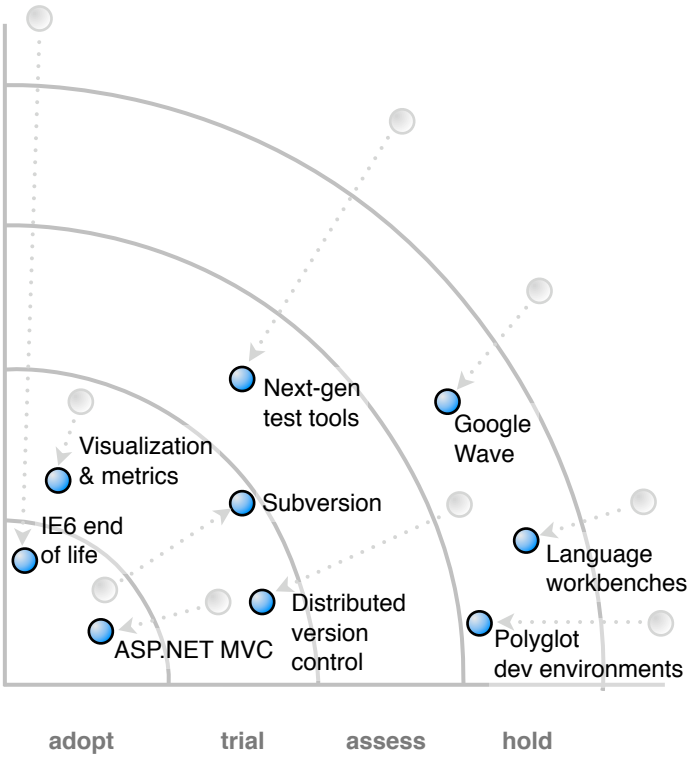
Tools

Evolutionary and emergent design of enterprise systems requires significant vigilance by development and architecture teams. Collecting **metrics** to capture development trends is a key part of understanding the stress points for a system under development. Assessing this information in its raw form is even more difficult than taking stock of a system at the source code level. To address this concern we have found a number of **visualization** tools and techniques to get what we refer to as the 1000ft view of the system and its internal quality. This 1000ft view allows us to identify visual patterns that help find and address issues more quickly.

Distributed version control systems such as Git and Mercurial have had significant exposure in the past year or more as open source projects move to this toolset en masse. The social networking aspect that GitHub and Bitbucket have brought to distributed version control has helped to propel these tools forward and into enterprises looking for ways to develop across multiple geographies. The move for many to a distributed version control system has resulted in a move away from tools such as **Subversion** and other centralized version control systems. As organizations assess and choose between these two different toolsets, we suggest that you evaluate both in relation to your team's specific needs. While we have seen widespread adoption of distributed version control tools within ThoughtWorks and beyond, we still advocate the use of continuous integration and limits to the amount of time that code is spent outside of the main branch.

Polyglot programming continues to gain widespread acceptance across the industry reflecting the reality that software developers have many languages and tools at their disposal. One area that we have yet to see take off is the creation of **polyglot development environments,** capable of satisfying multiple language needs of development teams. While Eclipse, IntelliJ, Visual Studio and others have some cross-language capabilities, their support for a wide range of languages is limited at best.

The Ruby language community is responsible for a number of innovations in the area of testing. The **next generation of testing tools** such as rspec and Cucumber are two such tools that have come out of this community. These tools, along with ThoughtWorks' Twist, provide a way to express tests in a more natural language syntax that captures the intent of the system in a way that end users can quickly grasp.



It is likely that test languages will continue to evolve with the assistance of **language workbenches**, tools that assist in the creation of domain specific languages. Tools such as Jetbrains' MPS and Intentional Software's offering are leading the industry in this area. Both provide ways of creating new languages to map business software more closely to the end user's domain language.

Google Wave has sprung up over the past few months and looks to be a promising platform for collaboration over the Internet. The platform is still in early beta and suffers from some stability issues. Some early developers have integrated with the Google Wave platform but commercial releases of software that utilize Google Wave will likely wait until the beta tag has been lifted from the product.

We have been tracking **ASP.NET MVC** since its early release candidates. This is an exciting development in the .NET space from Microsoft, both in the programming model and in the open source license under which Microsoft has released the library. ASP.NET MVC is similar to MVC frameworks on the Java platform and is a move away from the ASP.NET Web Forms approach to one that supports greater levels of automated testing.

Languages

While **JavaScript** first appeared in 1995, it is only in the past couple of years that libraries such as Prototype and JQuery have helped the language become more accessible to a wider developer audience. As developers continue to embrace JavaScript for developing rich user web applications, we increasingly hold JavaScript in the same level of esteem as any other production language, ensuring that scripts are adequately tested, refactored and maintained.

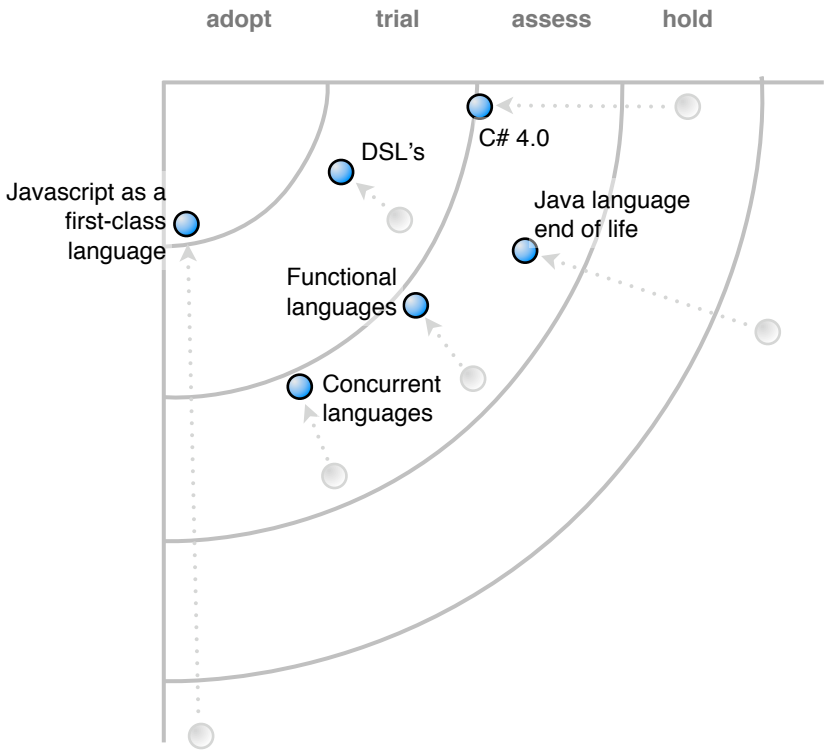
A significant amount of innovation occurred in the JavaScript space thanks to the Ruby on Rails community. This same community has helped to move both internal and external **DSLs** forward as a means for more closely mapping business requirements in code. Ruby's syntax lends itself easily to the creation of easily readable DSLs, while language tools such as ANTLR help to make the creation of new domain specific languages more accessible to interested developers.

When **C#** first appeared, many saw it as a direct competitor to the Java language's dominance in enterprise application development. This was often attributed to the syntactical similarities that the two languages shared. Since its introduction, however, C# has continued to move forward with the adoption of language features such as lambda expressions, extension methods, object initializers and automatic property setters and getters, all of which are available in the 3.5 release of the language. With the **4.0 release of C#,** we will see the introduction of a dynamic keyword and named and optional parameters, which will continue to bring C# more in line with languages such as Ruby and well ahead of the Java language.

As C# continues to surge ahead, the **Java language appears to be moving slowly** as the Java community waits for Java 7. Having waited for new language features to surface for almost 3 years, the Java community has begun to innovate in new languages that run on the Java Virtual Machine, languages such as Groovy, JRuby, Scala and Clojure. With the increase in number of languages available on the JVM, we expect enterprises to begin to assess the suitability of reducing the amount of Java specific code developed in their enterprise applications in favor of these newer languages.

The remaining two language types included on the radar are often grouped together. While functional and concurrent languages may be adopted in similar environments, their approaches are different. Functional programming focuses on expressing code in the form of mathematical functions that avoid maintaining state across multiple invocations. While **functional languages** such as Haskell have been around for a number of years, new functional (themed) languages such as Scala, F# and Clojure have sparked some interest in this paradigm. Due to the way in which functional languages manage state, interest in these languages has increased by many developers seeking to make the most out of multi-core processors.

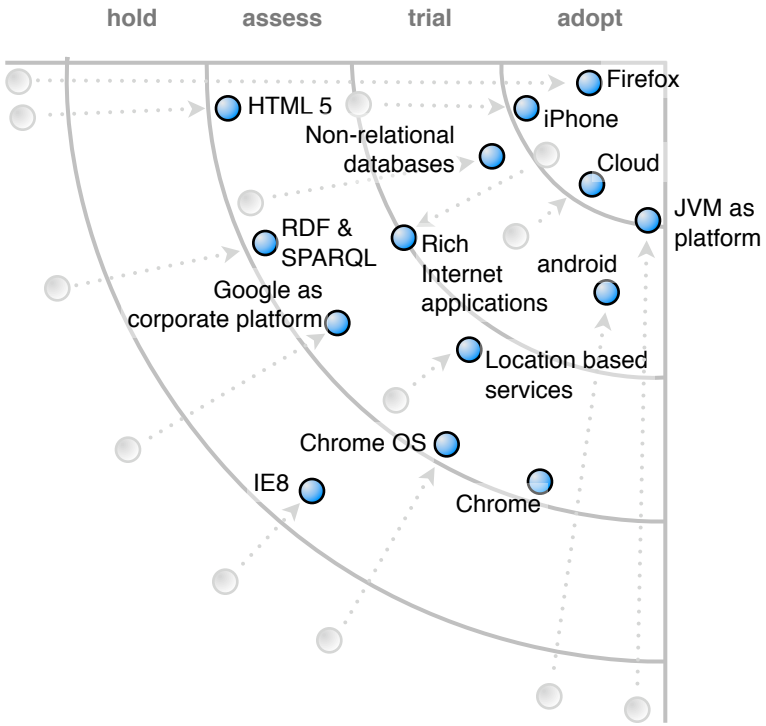
Many **concurrent languages** are also functional languages. The distinction lies in the emphasis on running operations in parallel. A number of such languages exist; Erlang is currently the most popular of these languages. Concurrent languages commonly provide some means for handling concurrency by using messages to communicate across multiple threads.



Platforms

Web browsers continue to evolve as they strive to keep pace with new specifications in HTML, CSS and JavaScript. Alas, many enterprises have yet to embrace the **end of life for IE6** and move to a newer and more standards compliant option. Of the browsers available today, **Firefox** and Opera provide support for the widest range of platforms. The Google browser, **Chrome**, brings new innovation to the browser space by splitting browser tabs into separate processes while providing a new implementation of JavaScript. These changes appear to give Chrome a significant performance boost over other browsers and have influenced the creation of a netbook OS called **Chrome OS**. While enterprises may look to move off IE6 and onto Microsoft's **IE8**, we remain concerned about IE8's current level of compliance to web standards.

The semantic web and its underlying technologies, including **RDF & SPARQL**, have been around for 8 years or more. Broader uptake of the **Cloud** and non-relational databases such Neo4j have helped move the semantic web into the reach of enterprise developers. Outside of the semantic web, **non-relational databases** are being adopted as alternatives to relational databases in a number of situations. Leveraging these technologies will require new approaches to architecture and development that suggest widespread adoption will only occur over a number of years.



The **iPhone** and **android** operating systems have rapidly become key players in the mobile platform marketplace. Apple's app store and Google's open source operating system have helped both companies leapfrog the competition in capturing developer mindshare.

While the radar has called out the possibility of the Java language nearing its end of life, the **JVM** is demonstrating its resilience as a general-purpose virtual machine for other languages such as Ruby, Groovy, Scala and Clojure.

Our position on **Rich Internet Applications** has changed over the past year. Experience has shown that platforms such as Silverlight, Flex and JavaFX may be useful for rich visualizations of data but provide few benefits over simpler web applications. Given that these toolsets have limited support for automated testing, it would suggest that a more traditional web application stack provides greater value for enterprise development. We recommend only using RIA platforms for rich visualizations incorporated into web applications, not as comprehensive development targets.

At the start of October, ThoughtWorks became a customer of Google Apps. Although we have heard a wide range of opinions about the user experience offered by Google Mail, Calendar and Documents, the general consensus is that our largely consultant workforce is happy with the move. The next step that we as a company are looking to embrace is **Google as a corporate platform** beyond the standard Google Apps; in particular we are evaluating the use of Google App Engine for a number of internal systems initiatives.

Google App Engine, Amazon EC2 and Salesforce.com all claim to be **Cloud** providers, yet each of their offerings differ. The Cloud fits into a broad categorization of service offerings split out into Infrastructure as a Service (e.g. Amazon EC2 and Rackspace), Platform as a Service (e.g. App Engine) and Software as a Service (e.g. Salesforce.com). In some cases, providers may span multiple service categories, further diluting the Cloud as a label. Regardless, the value of infrastructure, platform and software in the cloud is difficult to question and although many offerings have hit bumps in the road, they certainly have earned their position on the radar.

References

Jez Humble, Chris Read, Dan North. "The Deployment Production Line." *Proceedings of the conference on AGILE 2006*. 2006. http://tr.im/GtOt

Neal Ford. "Evolutionary architecture and emergent design: Investigating architecture and design." *developerWorks*. February 24, 2009 http://tr.im/GtOF

Neal Ford. "Evolutionary architecture and emergent design: Emergent design through metrics." *developerWorks*. June 30, 2009 http://tr.im/GtOX

M. Poppendick & T. Poppendick. "Implementing Lean Software Development: From Concept to Cash." *Addison-Wesley Professional*. 2006

S. W. Ambler & P. J. Sadalage. "Refactoring Databases: Evolutionary Database Design." *Addison-Wesley Professional.* 2006

Frederick Cheung. "Migrations." *RailsGuides*. 2008 http://tr.im/GtP2

Dave Robertson & John Johnston. "Agile methods and user centered design." *Infoq*. February 02, 2009 http://tr.im/GtP8

Interview with Erik Doernenburg. "Erik Doernenburg on Software Visualization." *Infoq*. October 19, 2007 http://tr.im/GtPg

Martin Fowler. "Feature Branch." *martinfowler.com*. September 3, 2009. http://tr.im/GtPI

Martin Fowler. "Language Workbenches: The Killer-App for Domain Specific Languages?" *martinfowler.com*. June 12, 2005 http://tr.im/GtPp

Scott Guthrie. "ASP.NET MVC Framework." *weblogs.asp.net.* October 14, 2007. http://tr.im/GtPx

Ben Parr. "Google Wave: A Complete Guide." *Mashable.* September 29, 2009. http://tr.im/GtPI

Douglas Crockford. "The World's Most Misunderstood Programming Language Has Become the World's Most Popular Programming Language." *crockford.com.* March 3, 2008. http://tr.im/GtPQ

"C# 4.0 Language Specification." *Microsoft Corporation.* March, 2009.

Michael Calore. "Norwegian Websites Declare War on IE 6." *Wired.* February 19, 2009. http://tr.im/GtQ0

Phillip van Hoof. "Introduction to RDF and SPARQL." *Replicating Memes*. July 14, 2009. http://tr.im/GtQ9

J. Ellis. "NoSQL Ecosystem." Rackspace Cloud Blog. November 9, 2009. http://tr.im/JwXJ

S. Pichai & L. Upson. "Introducing Google Chrome OS." *Google Blog.* July 7, 2009. http://tr.im/GtTy

Geva Perry. "Application Lifecycle in the Cloud." *Thinking Out Cloud.* November 10, 2009. http://tr.im/GtTE