

Technology Radar

Prepared by the ThoughtWorks
Technical Advisory Board

April 2010

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 reflects our current opinion on the item.

Hold: the item may be of interest to ThoughtWorks and others in the industry, however we think an enterprise should not yet invest significant time and resources to build experience with the item.

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: 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. With a large amount of new items for the radar, we have opted to differentiate new items (triangles) from existing items (circles).

Contributors

The ThoughtWorks Technical Advisory Board is comprised of

Rebecca Parsons (CTO)
 Martin Fowler (Chief Scientist)
 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. Build pipelines
2. Emergent design
3. Evolutionary DB
4. Web as platform
5. Automation of technical tests
6. Service choreography
7. Lean software development
8. Continuous deployment
9. Visualization and metrics
10. Polyglot programming
11. Incremental data warehousing
12. OAuth
13. Scrum certification

Tools

14. ASP.NET MVC
15. Subversion
16. Squid
17. Message buses without smarts
18. Next-gen test tools
19. Neo4j
20. mongoDB
21. Distributed version control
22. NoSQL
23. RDF triple stores
24. Restfulie
25. Visualizations for business data
26. GitHub
27. Cross mobile platforms
28. ESB
29. Language workbenches



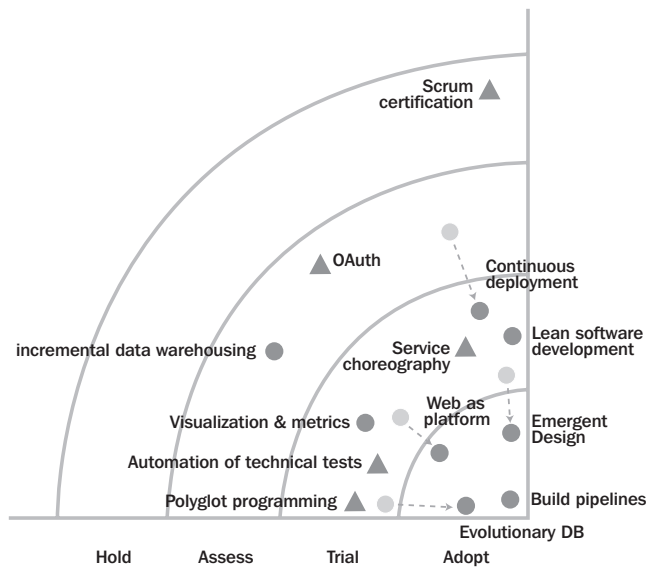
Platforms

40. IE6 end of life
41. Platform roadmaps
42. ALT.NET
43. iPhone
44. Cloud
45. JVM as platform
46. android
47. EC2 & S3
48. Location based services
49. Large format mobile devices
50. Facebook as business platform
51. Google as corporate platform
52. Application appliances
53. Google App Engine
54. mobile web
55. Rich Internet Applications
56. Azure
57. GWT
58. IE8
59. WS-* beyond basic profile

Languages

30. C# 4.0
31. Ruby/Jruby
32. Javascript as a first-class language
33. DSLs
34. Groovy
35. Java language end of life
36. F#
37. Clojure
38. Scala
39. HTML 5

Techniques



The world over, larger and larger organizations are looking to Agile methodologies to lead them out of the wasteful spending that plagues large scale software development projects. These corporations have the difficult challenge of re-skilling their people to take advantages of these approaches. To do so, a large number look to training and certification to plug the gap that experience has yet to fill.

Scrum was one of the founding approaches to Agile software development, and continues to provide a worthwhile core for the management side of software development. **Scrum Certification** schemes have proven counterproductive, granting only a veneer of competence, which often misleads teams into a distorted experience of agility.

Integrated business processes now routinely span multiple systems and even enterprises. This raises the question of how these processes should be coordinated. In our experience centralized orchestration solutions are costly to implement and often fail to deliver the promised benefits, which has lead us to prefer **service choreography** as an approach.

The Web is a global data structure that enables us to share information. However not all data is meant to be shared by everyone and it's important to be able to share information on the Web in a disciplined and governable manner without requiring massive centralized infrastructure. **OAuth** provides a way of sharing resources on the Web responsibly and securely. It is a Web protocol (for Web browsers or machine-to-machine interactions), which allows federated authorization of access to Web resources. What's interesting is that OAuth is a simple protocol to implement and utilize and yet its design goals match many common enterprise authorization problems. OAuth remains in the assessment category, however, because it has fragmented, and the IETF has not yet drawn the community back together under an Internet RFC.

Our understanding of the Web has matured to the point where we believe it is a viable platform for building distributed systems. RESTful techniques have advanced past pretty URIs + JSON towards hypermedia systems that project business protocols over the Internet and support seamless business process and service composition. The Web provides a powerful capability for scale, resiliency, and ease of implementation with commodity infrastructure like caches and Web servers with commodity protocols (like HTTP, AtomPub, and OAuth). Moving from trial to adopt is indicative of our position that the Web is ready for primetime, not just for Internet-facing systems but as a practical base for enterprise systems delivery.

Significant advances in the tools for **automating functional testing** haven't been replicated in the technical testing space. Data management for performance, load and soak testing is a particular issue. However, the tools are improving and increased visibility for these tools supports the early and often technical testing that we advocate.

Tools

IT departments are increasingly striving to liberate data from disparate systems. A broad set of approaches have been promoted under the generic term Service Oriented Architecture (SOA). This has led to confusion about what the term and approach actually means. We believe businesses do not need the complex enterprise service bus products advocated by vendors. **ESBs** actively undermine the reasons for choosing the bus approach: low latency, loose coupling, and transparency.

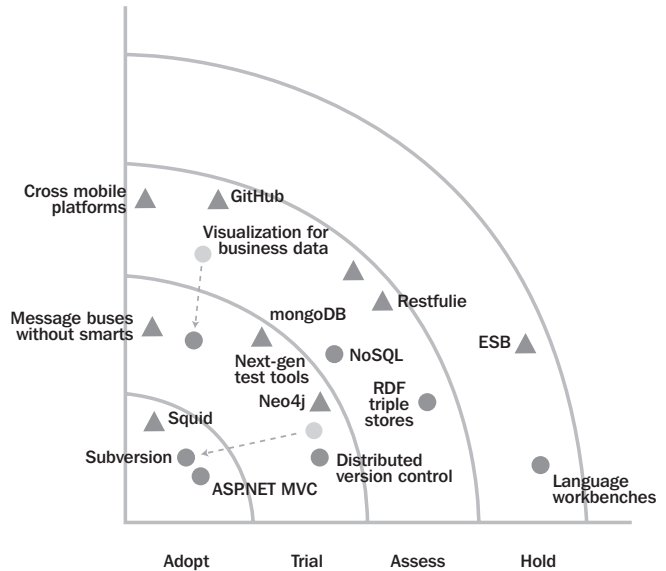
In contrast we have seen considerable success with **Simple Message Buses** where the integration problems are solved at the end points, rather than inside a vendor ESB system. The most well known Simple Message Bus approach is one based on the principles of REST and leveraging the proven scalability of the web. However organizations that have already invested in ESB infrastructure can leverage the useful parts of that infrastructure (reliable messaging etc) while still using a Simple Message Bus approach and performing integrations at the edges of the system.

The hypermedia constraint from REST is now understood as critical in sharing business protocols over the Web. Unfortunately many frameworks for building computer-to-computer systems on the Web are ignorant of this constraint and tend towards simple CRUD systems.

Restfulie is the first of a new generation of frameworks that natively support hypermedia, for Ruby, Java, and .NET. In Restfulie, business protocols are implemented using DSLs and exposed across the Web through hypermedia representations; clients drive those protocols through a similar declarative mechanism, consuming server-generated representations as they work towards a business goal. As the first framework of its kind, Restfulie is opinionated and provides strict “training wheels” in order to bootstrap newcomers. However, it is an empirical proof that the Web and hypermedia can be used to orchestrate complex business activities.

There continues to be significant activity in the **non-relational database** space. There are many different models, and widespread understanding of the capabilities of the different approaches is lacking. Thus, we still believe that this technology needs to remain in the assess ring, although we anticipate that this will change by the time of the next radar.

Graph databases store information as interconnected nodes with arbitrary relations rather than tables and nameless relations. Graph databases are an excellent choice for complex domains with semi-structured data since they're schema-less and highly extensible. **Neo4j** is the front-runner in the graph database space being an embedded Java component, which supports fast storage and search of graphs for Java solutions (including server applications). The Neo4j community is highly active and now has a basic REST API enabling it as more general-



purpose database engine. Neo4j moving into the trial category is representative of our experience trialling it in real-world scenarios and the early successes we've achieved.

Document-oriented databases treat each record as a document with the ability to add any number of fields of arbitrary size. A relatively large amount of the attention that has been directed at document databases has landed on **mongoDB**, a highly scalable option with support for querying, indexing, replication and sharding. Beyond its enterprise feature set, its popularity is aided by its driver support for Java, Ruby, PHP, C#, Python and a number of other languages.

Semantic Web W3C standards, and the tools implementing them, are at last worthy of real attention. **RDF** and RDFa allow anyone to say anything about anything in a sharable, structured format. This proves a much more powerful means of linking and structuring data from disparate sources than the strictness of RDBMS, or the mess that is unstructured Web data. Correspondingly SPARQL is the query standard that allows information to be mined from RDF marked-up data.

Subversion moves back into the Adopt section of the radar because it is a solid version control tool suitable for most teams. We consider Subversion's features to be the basic standard for a modern version control tool. ThoughtWorkers continue to embrace and recommend **Distributed Version Control** tools such as Git and Mercurial, but we caution that these systems often require deeper understanding to get the most out of them. New to the radar is **GitHub**, a “social coding” tool supporting both source code hosting and social networking. GitHub is arguably one of the main reasons Git has become the leading DVCS tool, and GitHub's collaboration features are often used by enterprises that need to support distributed teams.

Languages

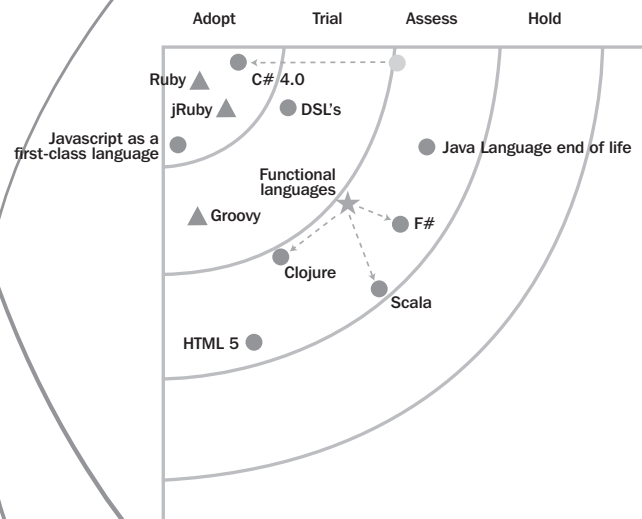
The past year or more has seen renewed interest in the development of new programming languages and the expansion of existing language feature-sets.

In the previous radar, we lumped functional languages together in a group. For this version, we've exploded that group and started calling out the ones interesting to us. Of the current crop of functional languages, the one we like the most is **Clojure**: a simple, elegant implementation of Lisp on the JVM. The other two that we find interesting are **Scala** (a re-thinking of Java in functional form) and **F#**, the OCaml derivative from Microsoft that now appears "in the box" in Visual Studio 2010.

Functional languages have a wide range of practical uses, including simulation, computational finance, computational science, large scale data processing and parsing. These fields benefit from functional programming techniques that simplify concurrent execution and the expression of complex mathematical functions concisely. Functional programming requires a shift in thinking for enterprise developers experienced in object oriented development. Moving to an often terse syntax for solving complex problems may initially be intimidating to many. As with all forms of programming languages, syntax is just one aspect of the language itself. In functional programming another significant aspect is the use of common idioms. These idioms speed code comprehension and increase overall maintainability.

This might not be news to all, but it is worth noting that dynamic languages are long ready for adoption and trial. **Ruby**, particularly when deployed on **JRuby**, is ready for adoption. ThoughtWorks uses Ruby and JRuby extensively in both its Services and Product work. **Groovy** is ready for trial and could prove more accessible than Ruby/JRuby in a Java shop. For the right type of applications, Ruby, JRuby, and Groovy prove far more effective, expressive, and productive than Java and C#.

As we have discussed previously, 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. This is not to say that enterprises should outright abandon Java as a programming language, we do however suggest that you look for alternatives that may be more fit for purpose in the area that new development is taking place.

HTML 5 offers a large number of improvements over HTML 4 and XHTML 1.0. Many of these improvements are focused on providing support for developing complex web applications, and improving integration of rich content such as audio and video in standard ways. Features such as client-side storage, web sockets and offline use will further establish the position of the web browser as a viable enterprise application platform.

Platforms

While we have found that many of the components necessary for scalable web enterprise architectures are well known and stable, our focus in the platforms space tends to the consumer side of the web. Web browsers and mobile devices are each undergoing significant changes that are likely to drive the creation of new service offerings for consumers and enterprises alike.

Internet browsers such as Google Chrome, Safari, Opera and Firefox, have made serious inroads in the implementation of the HTML 5 specification. With these advances it is now possible to experience many of the improvements that HTML brings. Unfortunately so far Microsoft has lagged on implementing these new standards. We recommend that organizations favor standards compliant browsers over **IE8**.

IE6 is a significantly deficient browser with many documented security holes and should be phased out as soon as possible. Browsers such as Firefox and Chrome can be installed alongside IE, allowing the user to choose which one to use. We recommend that organizations with intranet applications that require IE6 consider using it only for those specific applications, and install one of these alongside for general use.

While .NET has proven itself as a solid platform, many practitioners are dissatisfied with many of the default Microsoft tools and practices. This has led to the growth of the **Alt.NET** community, which champions techniques that we find more effective along with (usually open-source) tools that better support them.

Large format mobile devices, such as the Apple iPad and Amazon Kindle, provide a new model of ubiquitous computing. Their long battery life, simple interfaces and easy connectivity have the potential to change the way we interact with computers. Apple's new user interfaces discard the familiar desktop metaphors of files and folders

that have been standard since the introduction of the Macintosh in 1984.

Almost every enterprise has "legacy systems" that are expensive to operate and upgrade. Often a system will become legacy over the course of several years, through neglect or atrophy. We recommend using **platform roadmaps** to maximize the value of a systems portfolio and plan for the upgrade and eventual retirement of systems.

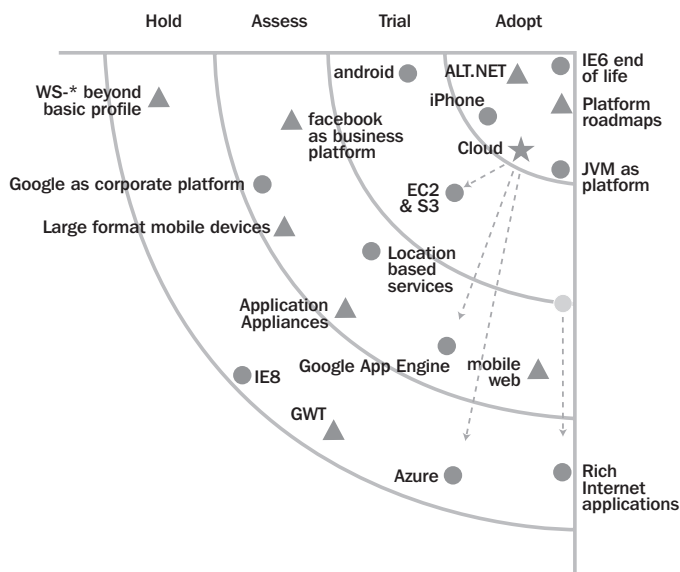
Facebook has become popular in part due to its rich API and explosion of third-party applications. ThoughtWorks is now starting to see our clients consider **Facebook as a business platform**. In addition to having a Facebook presence, businesses are building Facebook applications that are tightly integrated with their own services and offer useful functionality to Facebook users.

Web services are now widely used as an enabler for service oriented architectures as well as for the integration of existing applications. We see mature tools and largely interoperable implementations for web service standards covered by WS-I Basic Profile, but we remain skeptical about the proliferation and value of **WS-* standards beyond Basic Profile**.

Rich Internet Applications (RIA) are a popular topic, driven by the effort and marketing of big vendors pushing their offerings. RIA is useful for complex visualizations but ill-suited for other programming tasks because it doesn't fully support the engineering hygiene we require for our tools: testing is difficult and application partitioning is cumbersome. These frameworks also don't support common elements we take for granted in applications hosted in a browser: bookmarking, addressability, browser controls, and other aspects. We're not entirely critical of these tools, but think that their sweet spot is rich visualizations, not building traditional data entry CRUD applications.

Google Web Toolkit (GWT) offers an interesting premise: write Swing-like Java code and generate unit testable JavaScript widgets and user interfaces. From a practical standpoint this doesn't work well. First, using code-gen to produce the artifacts is time consuming, artificially extending build times and requiring manual changes to obtain optimal package layout. Second, if the JavaScript doesn't behave exactly as you want you will have to hack the generated code. Third, using Java to generate JavaScript means that you can't take direct advantage of the powerful features of JavaScript or numerous libraries such as JQuery. Finally, the JUnit support is quite limited, for example code using reflection cannot be tested.

The **Cloud** continues to be of interest to us, with Software as a Service the most mature cloud component. Platform and Infrastructure as service offerings have reached different levels of maturity, and we reflect that in our placement of **EC2**, **Google App Engine** and **Azure**.



References

R. C. Martin “Scrum / Agile’s inherit shortcomings? Uncle Bob Martin’s 7 theses.” Scrum Users Yahoo! Group. Feb 3, 2010. <http://bit.ly/ca6wdF>

M. Fowler. “Should there be a certification program for agile methods?” martinowler.com. April 30, 2004. <http://bit.ly/bwbbmp>

“Service Choreography.” Wikipedia. April 4, 2010. <http://bit.ly/boGefH>

J. Humble & D. Farley. “Continuous Delivery: A handbook for building, deploying, testing and releasing software.” Addison-Wesley Professional. April 9, 2010. <http://bit.ly/cssJu8>

R. Fielding. “Architectural Styles and the Design of Network-based Software Architectures.” University of California, Irvine. 2000. <http://bit.ly/blbUY6>

J. Webber, S. Parastatidis & I. Robinson. “REST in Practice: Hypermedia and Systems Architecture.” O’Reilly Media. March 2010. <http://oreil.ly/ccBtNK>

M. Fowler. “Richardson Maturity Model: steps toward the glory of REST.” martinowler.com. March 18, 2010. <http://bit.ly/cmrXMJ>

E. Hammer-Lahav, Ed. “The OAuth 1.0 Protocol.” IETF. February 5, 2010. <http://bit.ly/bwLMDm>

E. Sachs, A. Jain & P. Madsen. “Overlap of identity technologies.” Google OAuth & Federated Login Research. March 13, 2009. <http://bit.ly/9upnqU>

M. Fowler. “Version Control Tools.” martinowler.com. February 17, 2010. <http://bit.ly/9a4NmU>

S. Drobi. “Stuart Halloway on Clojure and Functional Programming.” InfoQ. March 12, 2010. <http://bit.ly/daScIN>

A. van Kesteren. “HTML5 differences from HTML4.” W3C. March 4, 2010. <http://bit.ly/bs0KDy>

K. Ballinger et al. “WS-I Basic Profile Version 1.1.” Web Services Interoperability Organization. April 10, 2006. <http://bit.ly/9XDUGL>

J. Miller. “What is ALT.NET?” MSDN. March, 2008. <http://bit.ly/c8XWNg>

M.Fowler. “AltNetConf.” martinowler.com. October 9, 2007. <http://bit.ly/atCex6>

ThoughtWorks is a global IT consultancy

We deliver custom applications and provide consulting grounded in reality; we help organizations become efficient through Agile and Lean practices and principles. By hiring exceptional people, we can solve our clients’ biggest and most pressing problems. All of our services are offered both on and offshore, and are delivered with pride and passion.