WS-* vs. REST: Mashing up the Truth from Facts, Myths and Lies

Sanjiva Weerawarana, Ph.D.

Founder, Chairman & CEO, WSO2
sanjiva@wso2.com

QCon San Francisco – November 8, 2007

About me ..

- Original author of WSDL 1.1, editor of WSDL 2.0
- Created Apache SOAP
- Longtime contributor to Apache Axis, Axis2
- Created Apache WSIF
- Co-author of BPEL4WS, WS-Addressing, WS-Eventing, WS-Policy and others
- Bean Markup Language (98), BSF (99)
- Author of “Web Services Platform Architecture”, Prentice Hall, 2005
- Member of Apache Software Foundation, ex-Board Member OSI, Founder & Director of LSF

A bit of history ..

- Why were “Web services” (aka WS-*) created?
  - 1998-9: Lots of people were building “e-commerce” applications using XML & HTTP
  - Everyone invented their own way to do security, reliability, transactions
    - E.g.: RosettaNet, ebXML
  - Not good if you want to be a middleware provider to multiple vertical industries
  - Needed a common way to do common things
- Cynical view:
  - RPC between .Net and Java

Web services design rationale

- World is not all about HTTP
- World is not all about XML
- World is not all about XML Schema
- Not all interactions are request/response
- Full security, reliability etc. are needed but not all the time: composability of features is key

Lie:
Web Services Need WS-*

Um, no.

- WS-* is just overhead unless you have something in your SOAP headers

```
<s:Envelope>
  <s:Header/>
  <s:Body>
    <RealXMLPayload/>
  </s:Body>
</s:Body>
```

- If HTTP(S) + XML is enough for the problem, more power to you

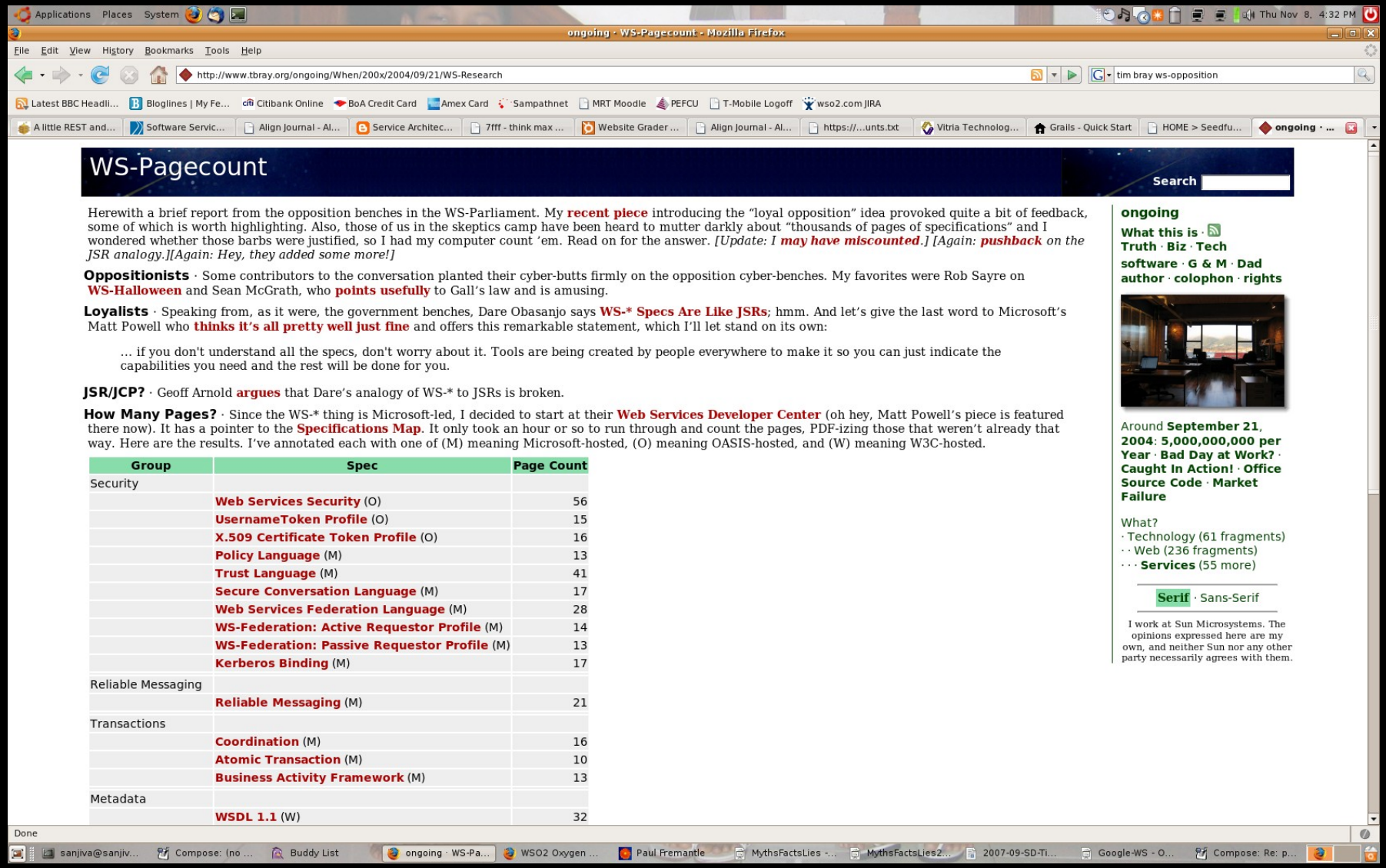
Lie:
Web Services Don't Need WS-*

Sure, let's all go back to 1998!

- There is no commonly accepted, aka interoperable, REST model for:
  - Message Signing / Non-repudiation
  - Reliable Messaging
- REST-* on its way! ARGH!!
  - HTTPR, anyone?
- You say “who needs them?”
  - Just listen to the next talk by Pete Lacey ;-)

Myth:
WS-* is Complex

Tim Bray ...



WS-Pagecount

Search

Herewith a brief report from the opposition benches in the WS-Parliament. My **recent piece** introducing the "loyal opposition" idea provoked quite a bit of feedback, some of which is worth highlighting. Also, those of us in the skeptics camp have been heard to mutter darkly about "thousands of pages of specifications" and I wondered whether those barbs were justified, so I had my computer count 'em. Read on for the answer. *[Update: I **may have miscounted**.] [Again: **pushback** on the JSR analogy.][Again: Hey, they added some more!]*

Oppositionists · Some contributors to the conversation planted their cyber-butts firmly on the opposition cyber-benches. My favorites were Rob Sayre on **WS-Halloween** and Sean McGrath, who **points usefully** to Gall's law and is amusing.

Loyalists · Speaking from, as it were, the government benches, Dare Obasanjo says **WS-* Specs Are Like JSRs**; hmm. And let's give the last word to Microsoft's Matt Powell who **thinks it's all pretty well just fine** and offers this remarkable statement, which I'll let stand on its own:

> ... if you don't understand all the specs, don't worry about it. Tools are being created by people everywhere to make it so you can just indicate the capabilities you need and the rest will be done for you.

JSR/JCP? · Geoff Arnold **argues** that Dare's analogy of WS-* to JSRs is broken.

How Many Pages? · Since the WS-* thing is Microsoft-led, I decided to start at their **Web Services Developer Center** (oh hey, Matt Powell's piece is featured there now). It has a pointer to the **Specifications Map**. It only took an hour or so to run through and count the pages, PDF-izing those that weren't already that way. Here are the results. I've annotated each with one of (M) meaning Microsoft-hosted, (O) meaning OASIS-hosted, and (W) meaning W3C-hosted.

Group	Spec	Page Count
Security		
	Web Services Security (O)	56
	UsernameToken Profile (O)	15
	X.509 Certificate Token Profile (O)	16
	Policy Language (M)	13
	Trust Language (M)	41
	Secure Conversation Language (M)	17
	Web Services Federation Language (M)	28
	WS-Federation: Active Requestor Profile (M)	14
	WS-Federation: Passive Requestor Profile (M)	13
	Kerberos Binding (M)	17
Reliable Messaging		
	Reliable Messaging (M)	21
Transactions		
	Coordination (M)	16
	Atomic Transaction (M)	10
	Business Activity Framework (M)	13
Metadata		
	WSDL 1.1 (W)	32

ongoing

What this is · **Truth** · **Biz** · **Tech** **software** · **G & M** · **Dad** **author** · **colophon** · **rights**

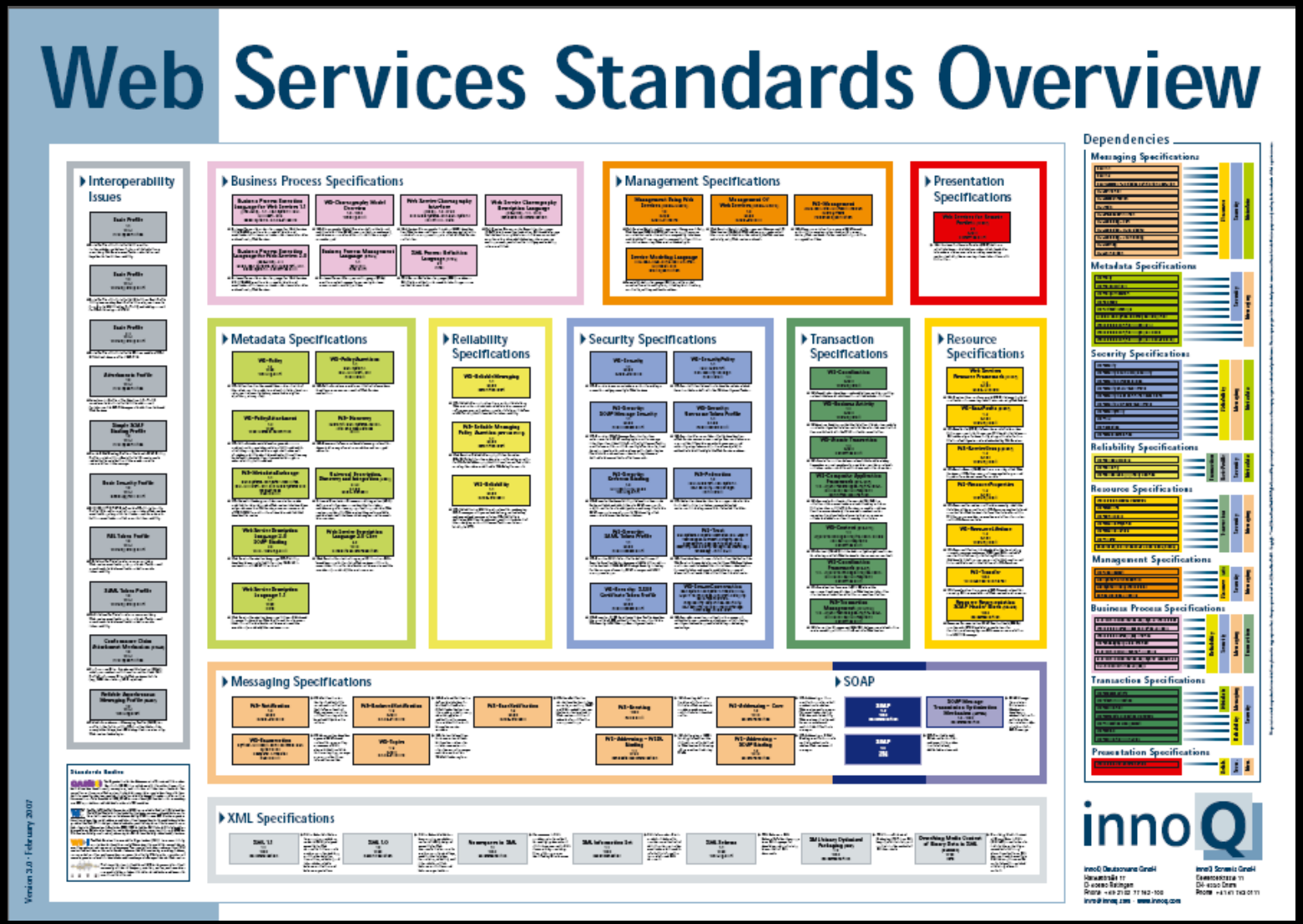
Around **September 21, 2004**: **5,000,000,000 per Year** · **Bad Day at Work?** · **Caught In Action!** · **Office Source Code** · **Market Failure**

What?
· Technology (61 fragments)
· · Web (236 fragments)
· · · **Services** (55 more)

Serif · Sans-Serif

I work at Sun Microsystems. The opinions expressed here are my own, and neither Sun nor any other party necessarily agrees with them.

InnoQ.com ...


Web Services Standards Overview
Interoperability Issues
Business Process Specifications
Management Specifications
Presentation Specifications
Metadata Specifications
Reliability Specifications
Security Specifications
Transaction Specifications
Resource Specifications
Messaging Specifications
SOAP
XML Specifications
Dependencies
Messaging Specifications
Metadata Specifications
Security Specifications
Reliability Specifications
Resource Specifications
Management Specifications
Business Process Specifications
Transaction Specifications
Presentation Specifications
innoQ
Version 3.0 - February 2007

Is WS-* really complex?

- For the middleware implementor- yes, quite
- For the application developer, NO!
  - If implementing services you should focus on dealing with the payload and let the middleware do the rest
  - Or find better middleware!

- WS-* programmers need to understand XML, XML Schema, WSDL and WS-Policy
  - If they tell you otherwise, find better software

Analogy

- Is TCP/IP complex?
  - For the stack implementor- yes, quite
  - For the application developer, NO!
- Is HTTP complex?
  - For the server implementor or client implementor- yes, quite
    - Not convinced? See Sam Ruby's ETech 2005 presentation: “Just” Use HTTP [http://intertwingly.net/slides/2005/etcon/]
  - For the application developer, NO!

Lie:
SOAP is RPC

Reality

- 1999: SOAP 0.9 – RPC, HTTP only
- 2000: SOAP 1.1 – RPC support, not HTTP only
- 2003: SOAP 1.2 – Messaging format with RPC pattern supported

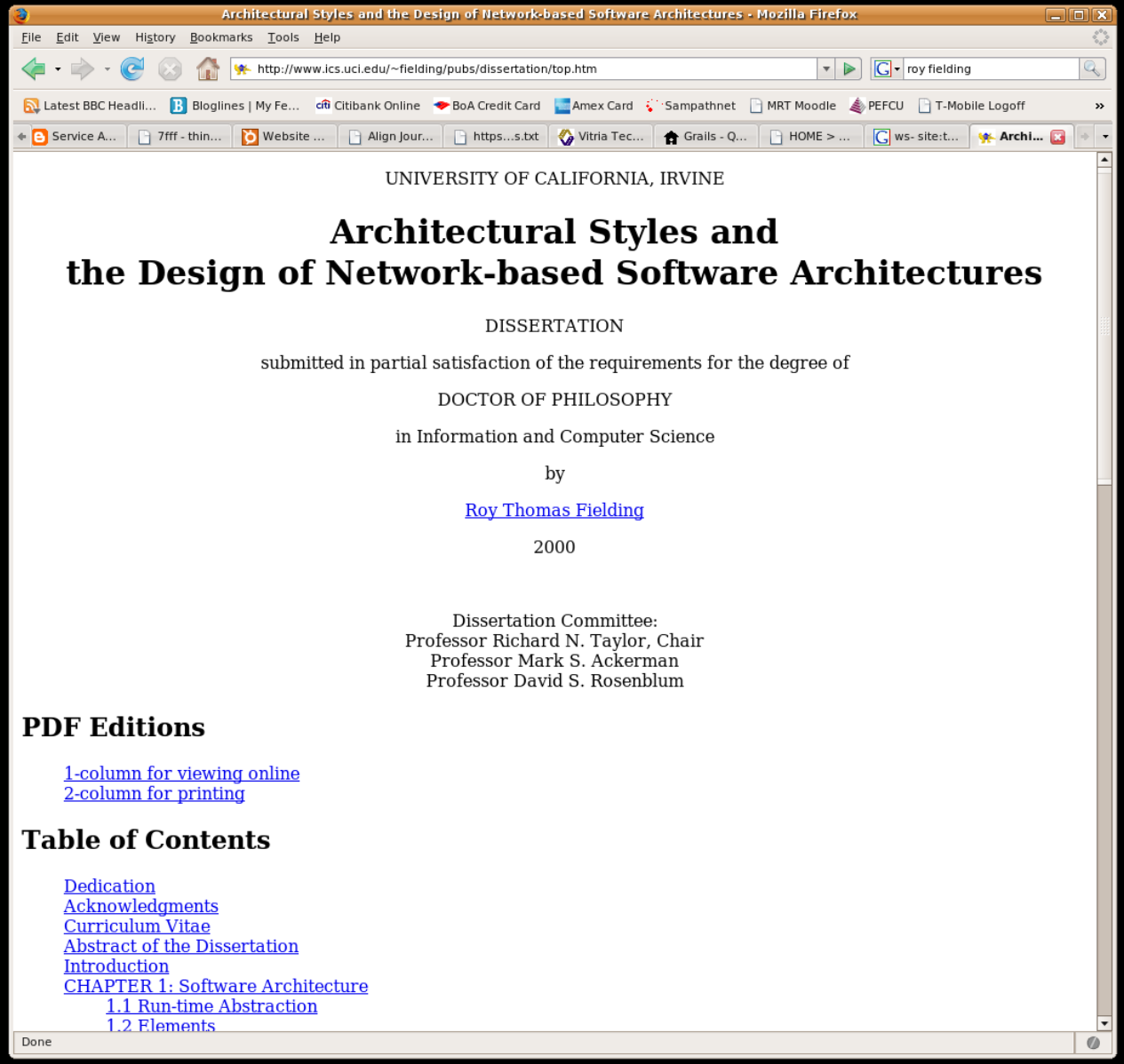
> “SOAP is fundamentally a stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns by combining such one-way exchanges”
>
> - SOAP 1.2 Primer, W3C

Myth:
REST is Easy to Learn

Really?

- HTTP 0.9/1.0/1.1, PEP, HTML, XHTML
- Media Types, MIME, S/MIME
- JSR 311 – JARWS
- POST Once Exactly
- SSL/TLS
- URL, URI, URN, IRI
- WebDav, DeltaV
- XForms, XML, XML Schema, XPath, XSLT, CSS
- JSON
- WebAPI, XMLHttpRequest, AJAX, Comet
- RDDL, Microformats, GRDDL, etc…
- Atom, Atom Publishing Protocol, GData, etc…
- RFCs 1945, 2068, 2069, 2109, 2145, 2169, 2227, 2295, 2296, 2518, 2616, 2617, 2774, 2817, 2818, 2935, 2936, 2964, 2965, 3143, 3205, 3229, 3230, 3310, 4130, 4169, 4229, 4236, 4387, 4559, 4918

Don't forget the bible:



UNIVERSITY OF CALIFORNIA, IRVINE

Architectural Styles and the Design of Network-based Software Architectures

DISSERTATION

submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

Roy Thomas Fielding

2000

Dissertation Committee:
Professor Richard N. Taylor, Chair
Professor Mark S. Ackerman
Professor David S. Rosenblum

PDF Editions

1-column for viewing online
2-column for printing

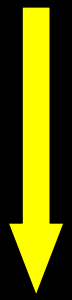
Table of Contents

And the new testament:

Lie:
REST is Simple

Um, no.

- REST is an “architectural style”
  - implementation
  - architecture
  - architectural style



Increasing levels of abstraction

- An analogy
  - implementation: code (say Java or C++)
  - model: UML
  - meta-model: MOF

And Model Driven Architecture was supposed to rule us all.

Reality

- True REST is still an art form
  - Example: AtomPub, the poster-child of RESTfulness, took a lot of effort by a lot of *really* smart people
    - (and apparently they didn't get it right .. Web3S)
  - And that's just one RESTful application
- Are you smart enough to build a RESTful application?
  - I know I'm not
- Are average developers & architects able to design RESTful systems correctly?

Lie:
REST is Easy

Hypermedia as the Engine of Application State

Fact:
REST is Full of Subtleties

The little details

- Method safety
  - GET, HEAD, OPTIONS, TRACE will not modify anything
- Idempotency
  - PUT, DELETE, GET, HEAD can be repeated and the side-effects remain the same
- Caching
  - Correct use of Last-Modified and ETag headers
- Uniform interface
  - In practice, most servers don't do PUT/DELETE

Lie:
REST Doesn't Need WSDL

Reality

- How do I know what query parameters I can include in a GET or POST?
- Developers need tooling- ability to find out what data a GET on a URL will give at development time is critical – text/xml isn't enough
- To make HatEoAS work, you need to really understand the media type and have *a priori* code written that know where to look for links that capture application state

Lie:
Content Negotiation, the Savior!

Content negotiation has failed

HTTP content negotiation was one of those "nice in theory" protocol additions that, in practice, didn't work out. The original theory of content negotiation was worked out when the idea of the web was that browsers would support a handful of media types (text, html, a couple of image types), and so it might be reasonable to send an 'accept:' header listing all of the types supported. But in practice as the web evolved, browsers would support hundreds of types of all varieties, and even automatically locate readers for content-types, so it wasn't practical to send an 'accept:' header for all of the types.

- Larry Masinter, April 11, 2006.

Myth:
REST Programmers Eat the Payload Directly

Reality

- WS-* tools have made programmers lazy by introducing “data binding”
  - Duh, what a mistake
- Programming XML in Java *still* sucks
- RESTfulness won't remove programmers' urge to look for restfulness

Myth:
WSDL is Wildly Popular

“Dev: (Reads WSDL spec). I trust that the guys who wrote this have been shot. It’s not even internally consistent. And what’s with all this HTTP GET bindings. I thought GET was undefined.”

- Pete Lacy, “S Stands for Simple”, Nov 15, 2006

Reality

- Any damned fool could come up with a better description language than WSDL!
- But .. you just have to get the whole world to accept it.
  - Good luck – see you at QCon 2015!

Myth:
WSDL can't describe RESTful Services

Reality

- True, WSDL 1.1 was terrible at it
- WSDL 2.0 can describe any RESTful service!
  - Example of an APP description
- WSDL 2.0 and WADL are basically the same, except inverted in thinking
  - WADL: start with resources and show what operations you can do against them
  - WSDL: start with operations and say which resources you can apply them to

Myth:
HTTP, the One True Protocol

Reality

- Enterprisey
  - JMS, SMTP, TCP, IIOP, MQSeries
- Cool
  - Jabber/XMPP, YahooIM, SIP

Also ...

- HTTP's uniform interface is the greatest – until we need just a tad more
  - WebDAV & DeltaV: a whole bunch more
  - PATCH: just one more to get APP right

Myth:
REST is Multiprotocol

.. was one of those "nice in theory" .. that, in practice, didn't work out

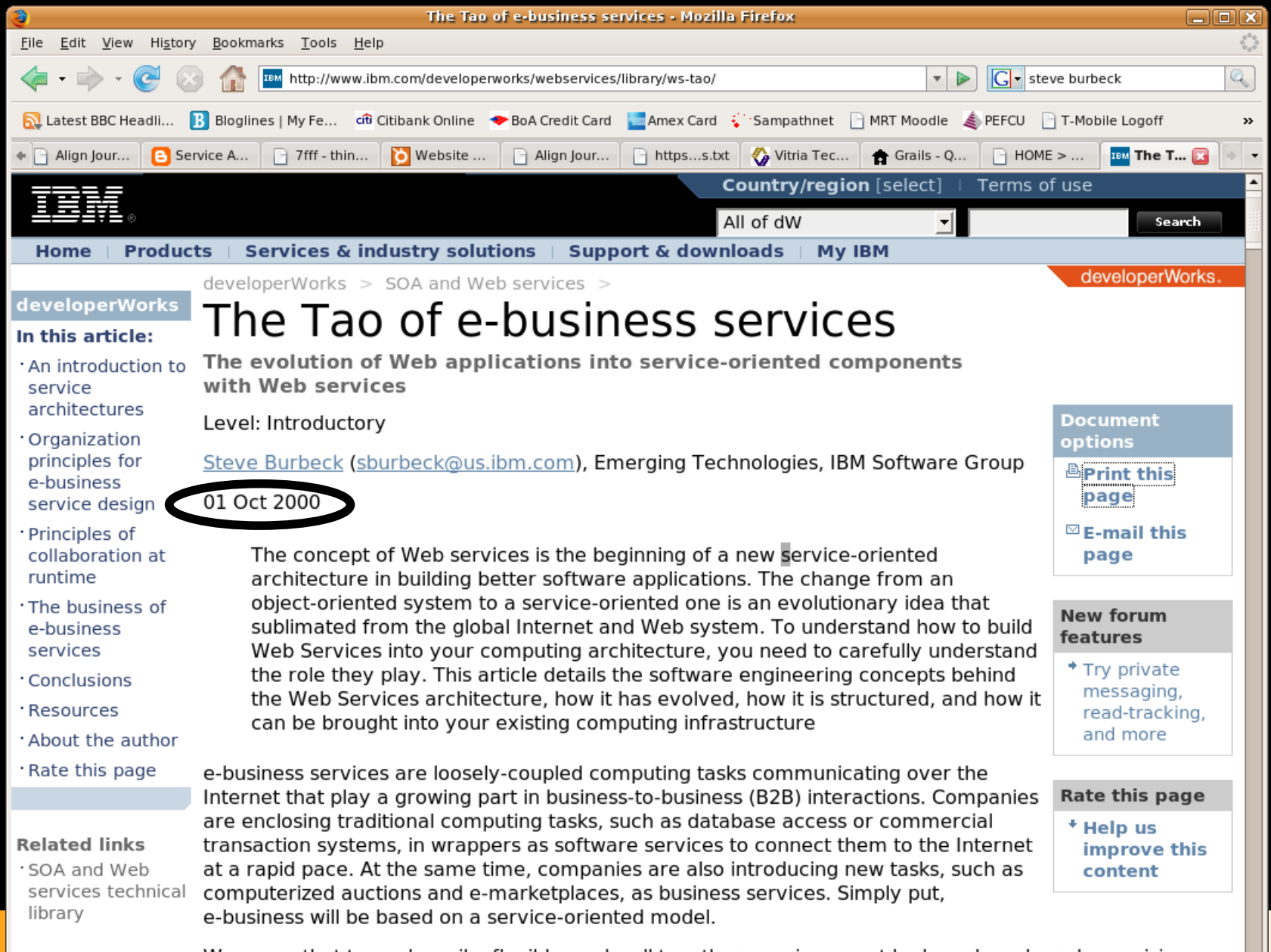
Myth:
REST is Scalable

Reality

- True, very true ..
  - .. as long as you don't want security
- Oh- you want both caching and security? Sorry, we don't do that here.

Myth:
SOA was a Response to REST

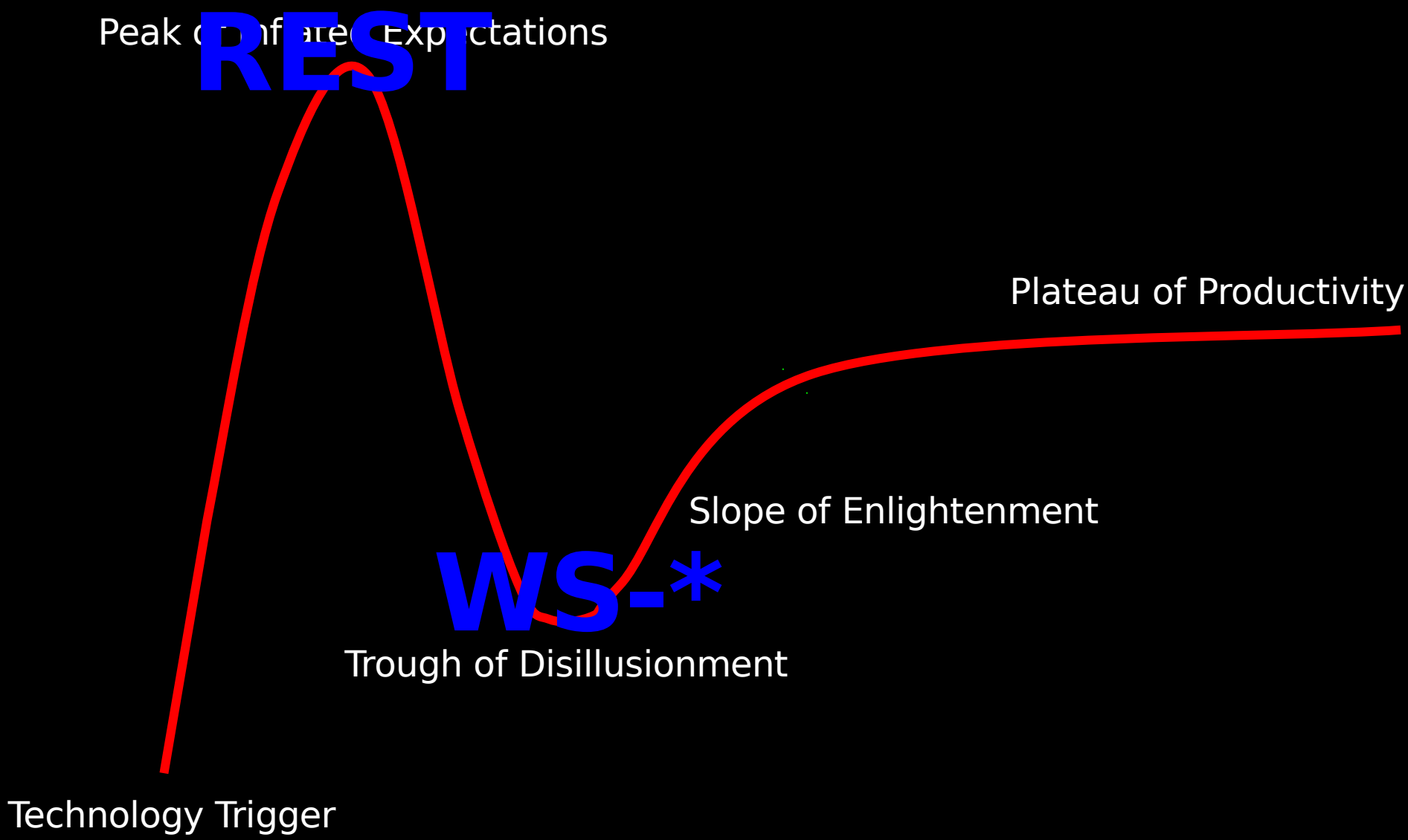
Huh?


The Tao of e-business services - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.ibm.com/developerworks/webservices/library/ws-tao/
steve burbeck
Latest BBC Headli... Bloglines | My Fe... Citibank Online BoA Credit Card Amex Card Sampathnet MRT Moodle PEFCU T-Mobile Logoff
Align Jour... Service A... 7fff - thin... Website ... Align Jour... https...s.txt Vitria Tec... Grails - Q... HOME > ... The T...
IBM
Country/region [select] | Terms of use
All of dW
Search
Home | Products | Services & industry solutions | Support & downloads | My IBM
developerWorks.
developerWorks > SOA and Web services >
developerWorks
In this article:
An introduction to service architectures
Organization principles for e-business service design
Principles of collaboration at runtime
The business of e-business services
Conclusions
Resources
About the author
Rate this page
Related links
SOA and Web services technical library
The Tao of e-business services
The evolution of Web applications into service-oriented components with Web services
Level: Introductory
Steve Burbeck (sburbeck@us.ibm.com), Emerging Technologies, IBM Software Group
01 Oct 2000
The concept of Web services is the beginning of a new service-oriented architecture in building better software applications. The change from an object-oriented system to a service-oriented one is an evolutionary idea that sublimated from the global Internet and Web system. To understand how to build Web Services into your computing architecture, you need to carefully understand the role they play. This article details the software engineering concepts behind the Web Services architecture, how it has evolved, how it is structured, and how it can be brought into your existing computing infrastructure
e-business services are loosely-coupled computing tasks communicating over the Internet that play a growing part in business-to-business (B2B) interactions. Companies are enclosing traditional computing tasks, such as database access or commercial transaction systems, in wrappers as software services to connect them to the Internet at a rapid pace. At the same time, companies are also introducing new tasks, such as computerized auctions and e-marketplaces, as business services. Simply put, e-business will be based on a service-oriented model.
Document options
Print this page
E-mail this page
New forum features
Try private messaging, read-tracking, and more
Rate this page
Help us improve this content

SOA

- IBM Emerging Technology group (under Rod Smith) was thinking about SOA from late 90s
  - This paper was a culmination of that work!
- First “SOA Platform”, IBM Web Services Toolkit, was first released in early 2001
  - Original version done by my group in IBM Research :-)
- SOA was *NOT* an industry response to REST

Fact:
REST is HOT, WS-* is NOT!


REST
Peak of Inflated Expectations
Plateau of Productivity
Slope of Enlightenment
WS-*
Trough of Disillusionment
Technology Trigger

Damned Lie:
Its easy with REST or WS-*!

Reality

- Distributed computing is hard no matter what!

My Advice

- Don't get caught up in hype
- REST and WS-* both have strengths and weaknesses; neither is the silver bullet
- If writing services, write them so you can offer either a RESTful interface or a WS-* one
  - Similar to what POJOs did for J2EE .. focus on your part and let the environment provide the rest
- Building scalable, interoperable distributed systems is still hard

Nahh. Just switch to Erlang.

Wow, it really is so clear up here!

Questions?