REST as a better web service paradigm

Norman Gray
VO-TECH / Uni. Leicester / Uni. Glasgow

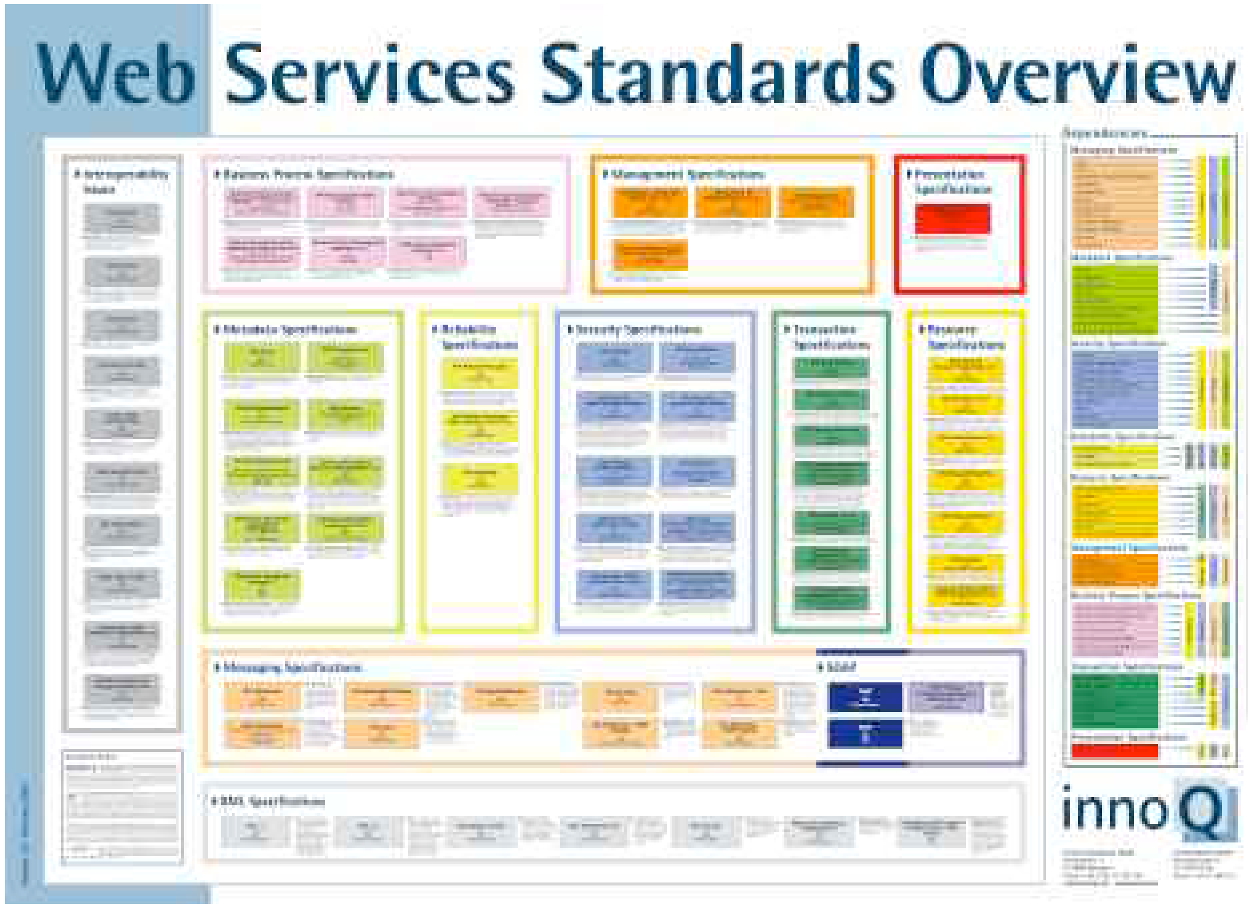
IVOA Interop, Beijing, 2007 May 15

overview

Plan:

- First, a bit of SOAP-bashing, just to get warmed up.
- What is REST supposed to be?
- What does a REST service look like?

soa is simple…



http://www.innoq.com/soa/ws-standards/poster/

soa is friendly...

- Guns don't kill people, the SOA WS-* stack kills people.
- The first rule of SOA is you do not talk about SOA.
- SOA actually stands for SOA Oriented Architecture.
- Saddam didn't have WMD, he had SOA. But SOA is so powerful, they went with the WMD angle instead to quell fear.

`http://www.soafacts.com/`

soa is useful…

Quoth Mark Nottingham, one-time chair of the WS-Addressing WG:

> Show me the interoperable, full and free implementations of WS-* in Python, Perl, Ruby and PHP. You won't see them, because there's no intrinsic value in WS-* unless you're trying to suck money out of your customers. Its complexity serves as a barrier to entry at the same time that it creates "value" that can be sold.

http://www.mnot.net/blog/2006/05/10/vendors

this is not supposed to be a theological matter

Image: burning at the stake

it’s supposed to be engineering

- Claim: REST is a better impedance match to the web
- Claim: it’s worked for 15 years so far (compare CORBA and HTTP: which protocol’s endpoints do you most often see on the sides of busses?)
- Claim: it’s much less brittle than RPC
- Compare unix character streams

rest is not the same as http

…though they're often conflated.

- HTTP is RFCs and Apache and stuff – web architecture
- REST is a design pattern – ‘representations’, ‘state’
- HTTP 1.1 was designed with REST in mind, which is why the link is natural
- You can talk about REST in purely CRUD terms
- …HTTP is just the ‘transfer’ part.

tool support

‘Traditionally’ done without tool support, not because it's necessarily *easy*, but because the technology isn't the hard bit. However:

- Restlet `http://www.restlet.org/`: RESTful servlets
- JSR-311 `http://jcp.org/en/jsr/detail?id=311`: JAX-RS – The JavaTM API for RESTful Web Services
- WADL `https://wadl.dev.java.net/`: mostly client support (Matthew)

there are a few restful koans, though

Ommm…

use http as an application protocol

…rather than merely a transport protocol.

This is probably the most instructive one. The idea is that, with the principal HTTP verbs `GET`, `PUT`, `POST` and `DELETE`, you can describe all the important changes to your application state.

And if you really can't, you might ask yourself whether your application should be on the web.

A bit like CRUD in databases.

name states rather than actions

. . . or Nouns Not Verbs!

If you get this right, *then* the previous point is easy.

A name can be passed around straightforwardly on busses (diesel or memory) with less chance of everyone getting confused. You can't do this so easily with a verb/message: whom can I send this message to?, when?, can I replay it?, are *you* allowed to?, can I store/duplicate/discard it? A name's just a name.

URIs name *states* of the applications, and you retrieve *representations*.

rest is and is not like o-o

You know you’ve ‘got’ O-O, when you can see why objects are named with nouns and methods are named with verbs.

A REST-inspired design is the same, except that (in effect) the method names are chosen for you, and the set of names is (in effect) your API. Thus the choice of names becomes a weightier design decision.

The upside is that this forces you to ask yourself some very useful questions about what it is you're designing, and pushes you towards a design that's simple and powerful.

exploit http – everyone else does

A large chunk of RFC 2616 is taken up with discussing when things may and may not be cached. Hint: `GET` is idempotent.

Optimisations are a function of the strength of the assertions you can make about a system.

There are probably more HTTP status codes and headers than you recall.

HTTP will change on a vastly slower timescale than your private SOAP schema, thus removing a mass of brittleness immediately.

avoid fallacies of distributed computing

- The network is reliable
- Latency is zero
- Bandwidth is infinite
- The network is secure
- Topology doesn't change
- There is one administrator
- Transport cost is zero
- The network is homogeneous

what does a rest service look like?

Guy Rixon's Universal Worker Service (roughly):

- `POST` to `http://example.org/uws`; get back `201 Created` with a `Location` header `.../uws/job123`
- `GET .../uws/job123/expirytime`, and `PUT .../uws/job123/expirytime` to extend the deadline
- `GET .../uws/job123/results`; get `200 OK`, `500 Error` or `304 Not Modified` with a `Retry-After` header
- Bored? `DELETE .../uws/job123`

pointers

`http://www.ietf.org/rfc/rfc2616.txt` The biz.

Roy Thomas Fielding , *Architectural Styles and the Design of Network-based Software Architectures*, 2000, `http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm`

REST polemic Stefan Tilkov, 2006 `http://www.bejug.org/confluenceBeJUG/display/PARLEYS/REST+-+The+Better+Web+Services+Model`

quoth...

To give and not to count the cost; / To fight and not to heed the wounds; / To toil and not to seek for **rest**; / To labour and not ask for any reward / Save that of knowing that we do Thy will.
— Loyola, on SOA

It is a far, far better thing I do, than I have ever done; it is a far, far better **rest**, that I go to, than I have ever known.
— Dickens, on REST