SOA Governance for WS-* using REST

Paul Fremantle, CTO, WSO2
paul@wso2.com

Paul Fremantle

- Co-founder and CTO, WSO2
    - Open source SOA middleware
- Chair, Apache Synapse PMC
- Co-Chair, OASIS WSRX TC
- Previously STSM at IBM Hursley Lab
    - IBM WebServices Gateway, WSIF, JSR110, etc

Contents

- Understanding SOA and Metadata
- Requirements for an SOA Registry
- Resources and REST design
- Applying this to SOA Metadata
- Atom Publishing Protocol
- REST design issues
- How does this apply to WS-*
- "Governance" – what is it, what does it mean?
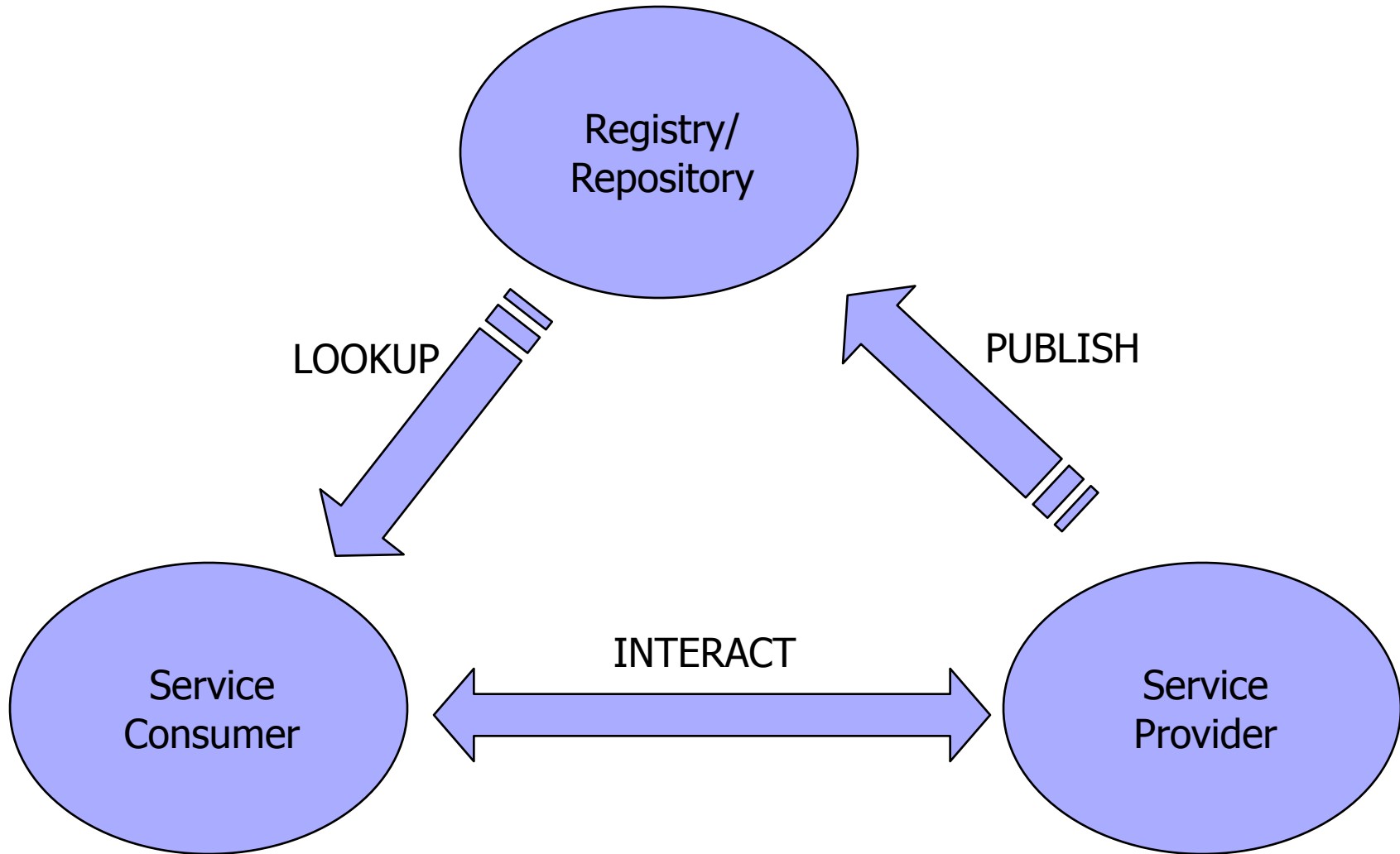
What is SOA Governance?

- Understanding what services are available
- Does this service meet enterprise guidelines?
- What point in the lifecycle are they?
  TEST->SYSTEM TEST->PRODUCTION->DEPRECATION
- Is this service available? 24x7?
- Who is the business contact? Technical contact?

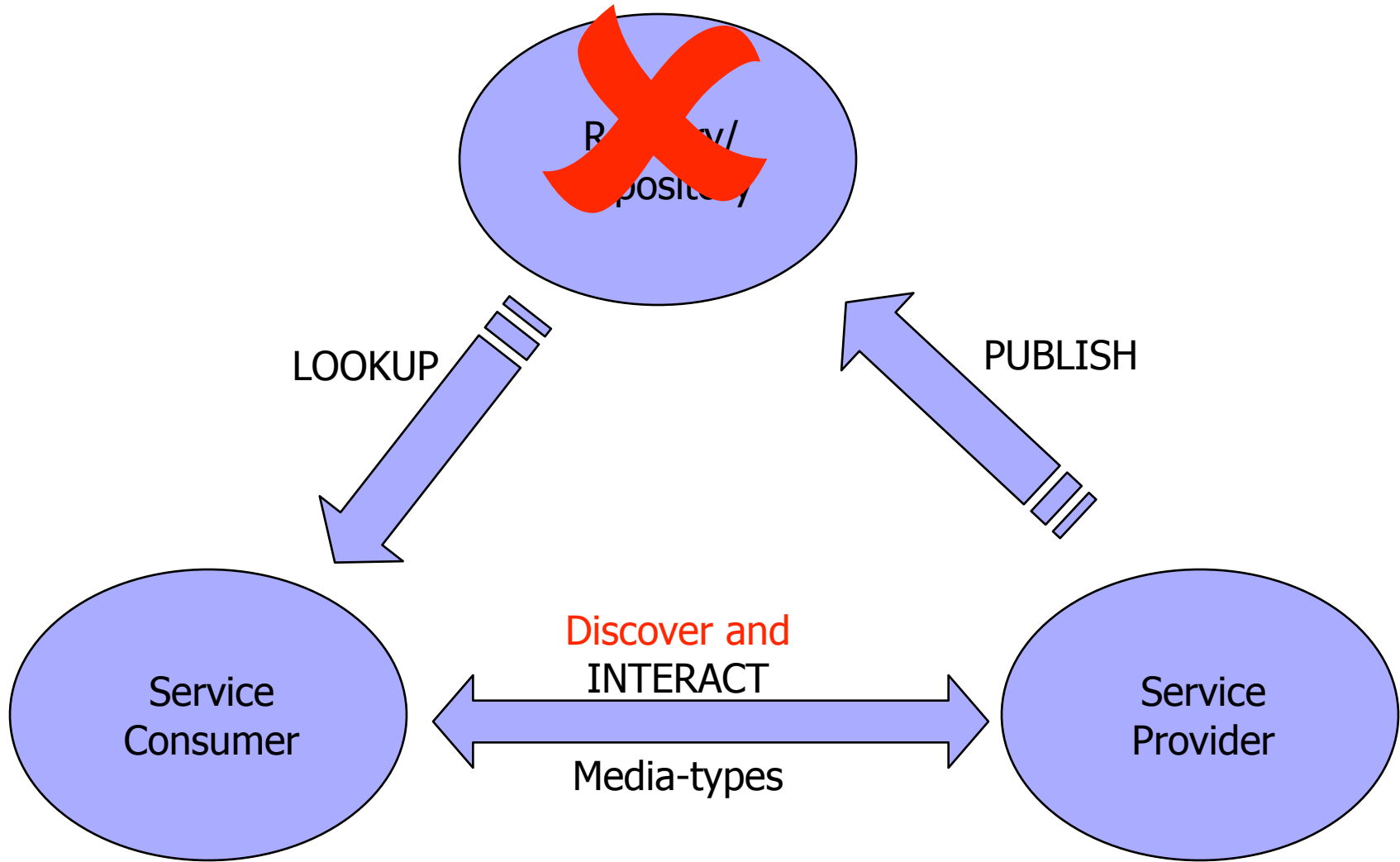- Governance is managing and implementing corporate policies in an effective way

What is SOA Governance?

- Governance *is* implementing a Registry/ Repository
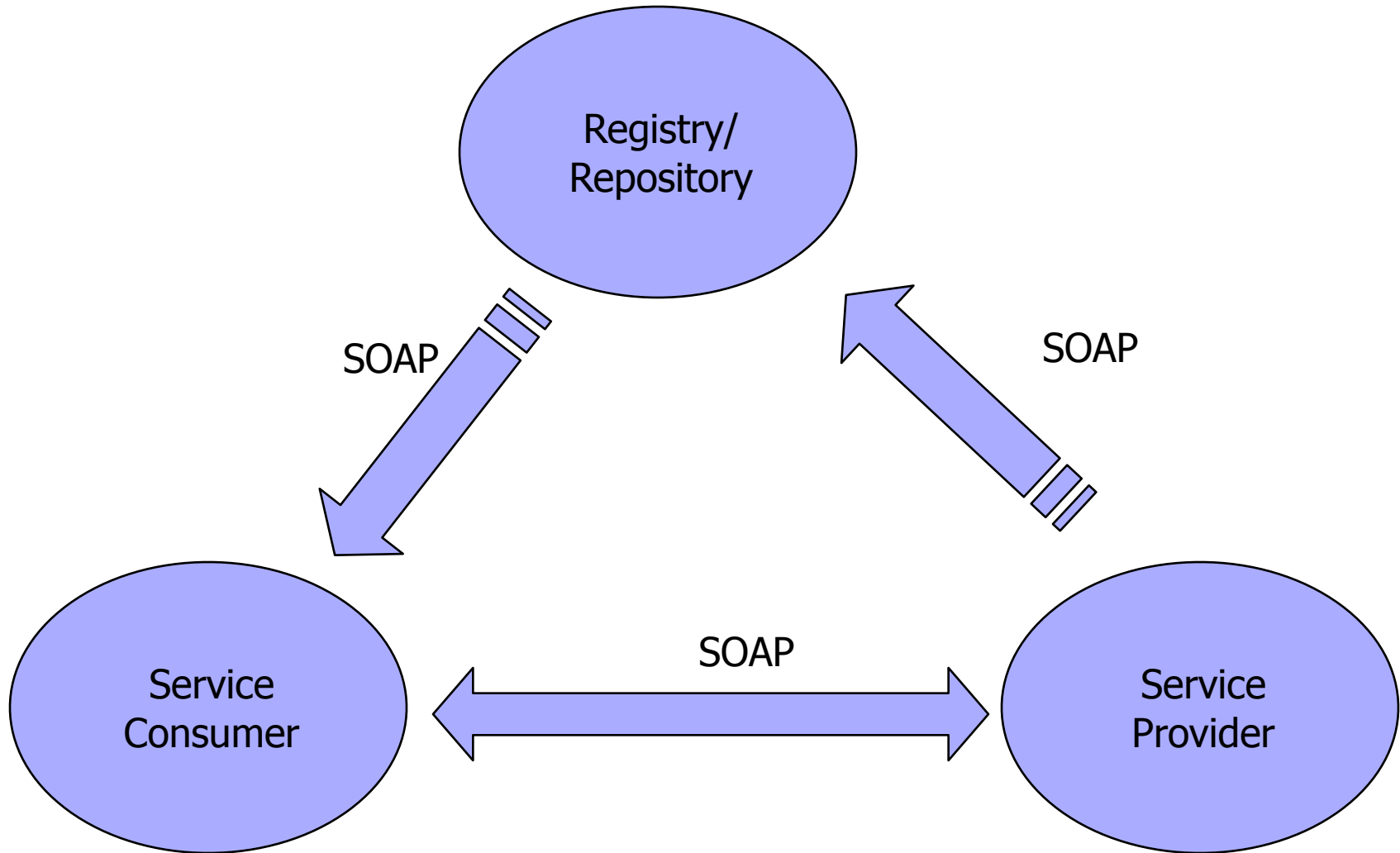 - with the right lifecycle, policies, metadata and rules
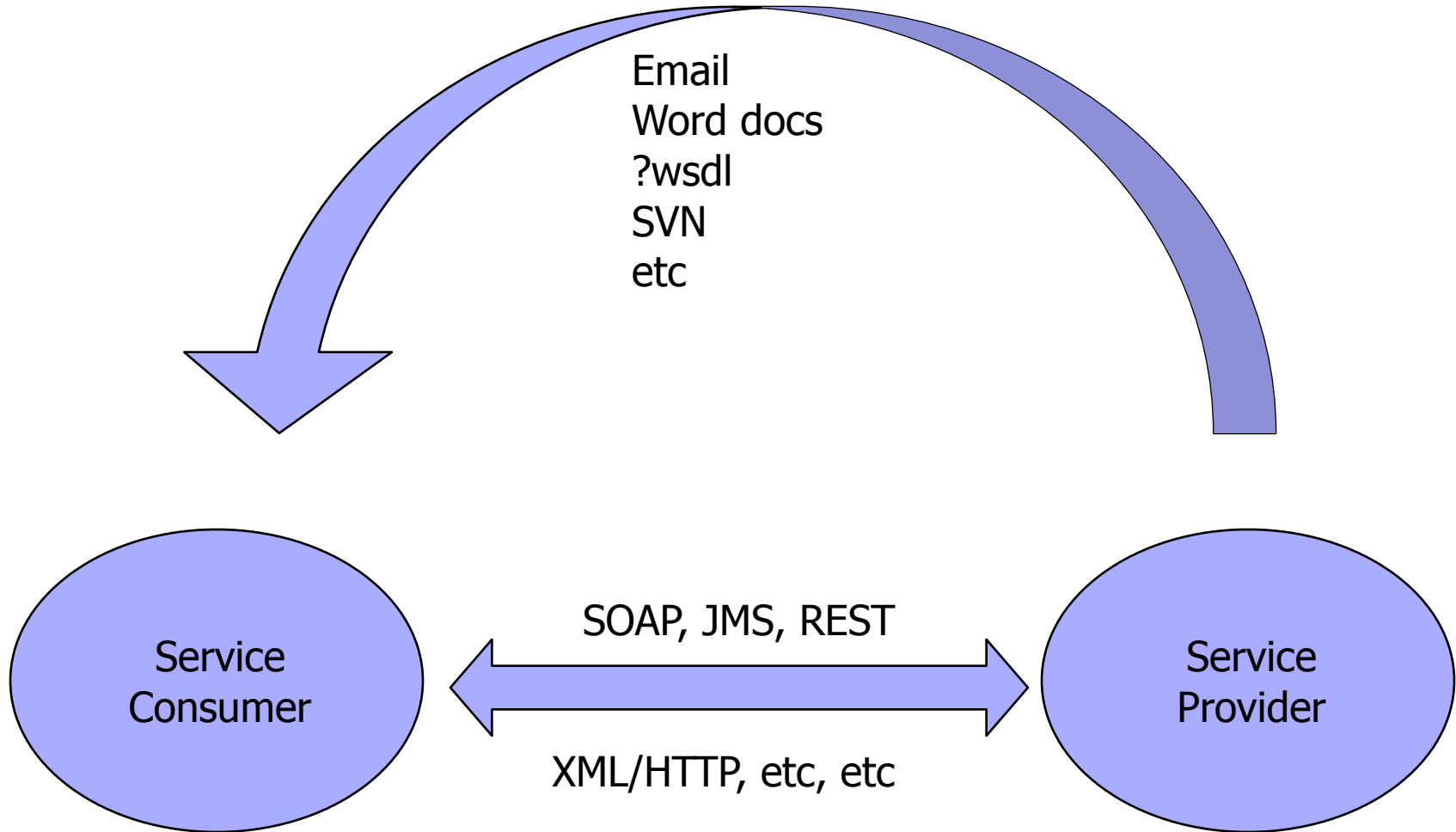
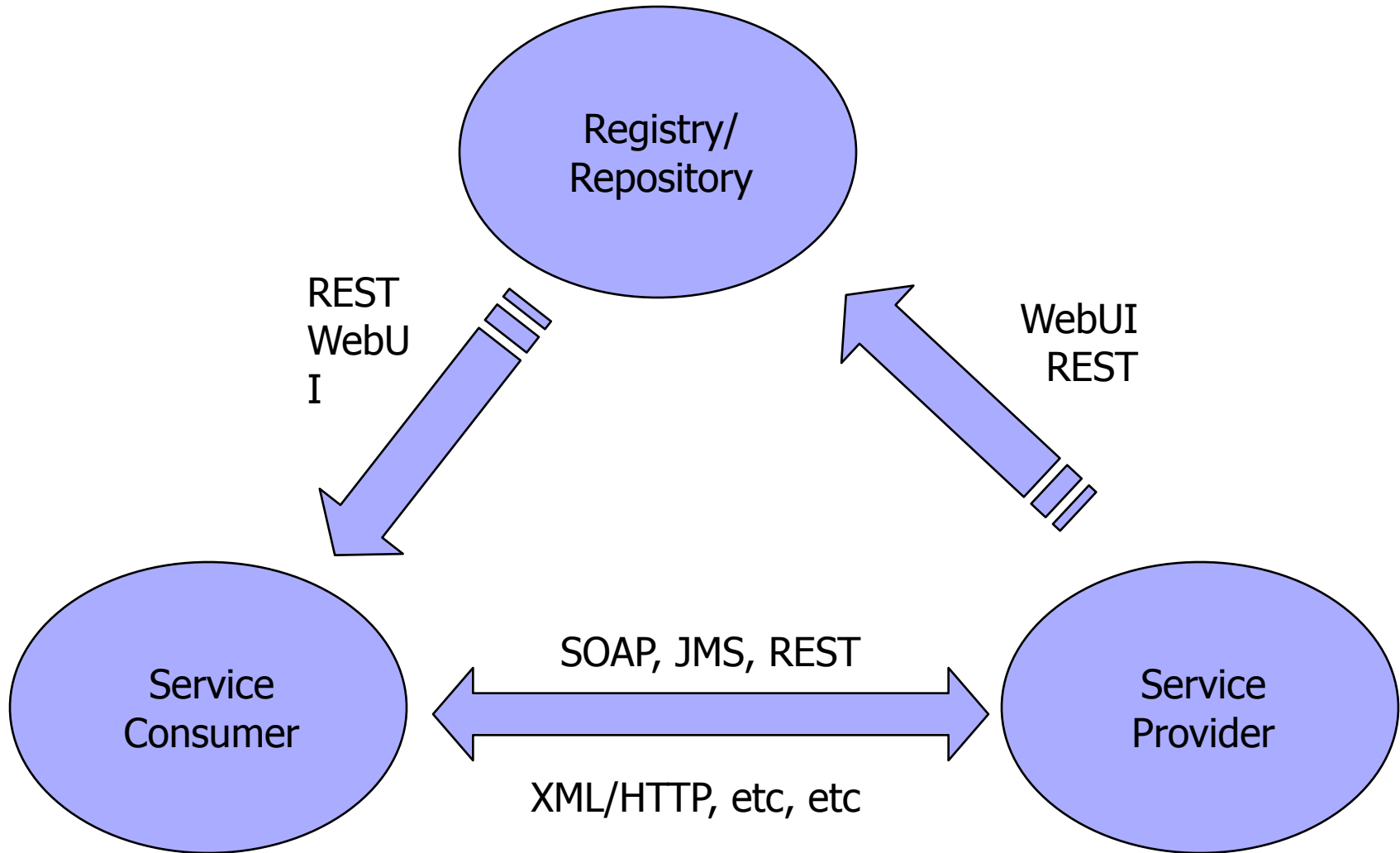# The oldest SOA picture of all

One strong REST view

One problem with UDDI

The Reality of SOA

Email
Word docs
?wsdl
SVN
etc

Service
Consumer

SOAP, JMS, REST

XML/HTTP, etc, etc

Service
Provider

Our view

Where did UDDI come from?

- Publish, categorize and search Web Service definitions
- Designed with "homogenous" thinking
  - Assumed that everyone will work to the same set of interfaces
  - Based on strict criteria, systems will automatically find service instances that offer a given interface
- Fundamentally based on the same model as Windows Registry
  - Long UUIDs - tModels
  - Lots of interlinking

This is a valid set of requirements

SOA Developers can publish WSDLs and WS-Policies and search for service definitions

The system shows dependencies between services, schemas and other dependent artifacts

But only a small part of the requirements

SOA Developers can publish WSDLs and WS-Policies and search for service definitions

The system shows dependencies between services, schemas and other dependent artifacts

Registry characteristics/requirements

Business users feel happy to create and document 'domains'

Developers can comment on what works and doesn't, best practice, hints and tips

Using my favourite blog reader I can subscribe to comments on my services

---

SOA Developers can publish WSDLs and WS-Policies and search for service definitions

The system shows dependencies between services, schemas and other dependent artifacts

---

Using simple APIs, content handlers can be written to perform dependency analysis, extract useful data and validate against policies.

Simple metadata properties allow the lifecycle of services to be managed.

Standard APIs allow systems to publish and consume metadata without understanding complex standards

---

Every change is versioned and I can rollback at any point to a previous revision

Security controls allow me to configure exactly who can read, write, delete and manage authorization for each resource

The system can be run in a highly-available load-balanced cluster

Social Governance

Business users feel happy to create and document 'domains'

Developers can comment on what works and doesn't, best practice, hints and tips

Using my favourite blog reader I can subscribe to comments on my services

REST design

- Everything is a Resource, identified by a URI
- Everything has a Uniform Interface (PUT, POST, GET, DELETE)
- The representation you get is based on Content-Type
  - e.g. text/xml, image/jpeg
- Interactions are stateless
- Links are key
  - "Hypermedia as the engine of application state" (HATEOAS)

REST design (continued)

- Ideally the "site" and the "api" are the same
  - Based on Accept headers each client gets the representation they like
- In reality very few sites work like this
  - Many sites are not stateless – use sessions
  - But not so good for APIs
- Navigational context is easy for people to figure out
  - No simple technical description of HATEOAS

How to apply this to SOA metadata?

Building an SOA Registry with REST
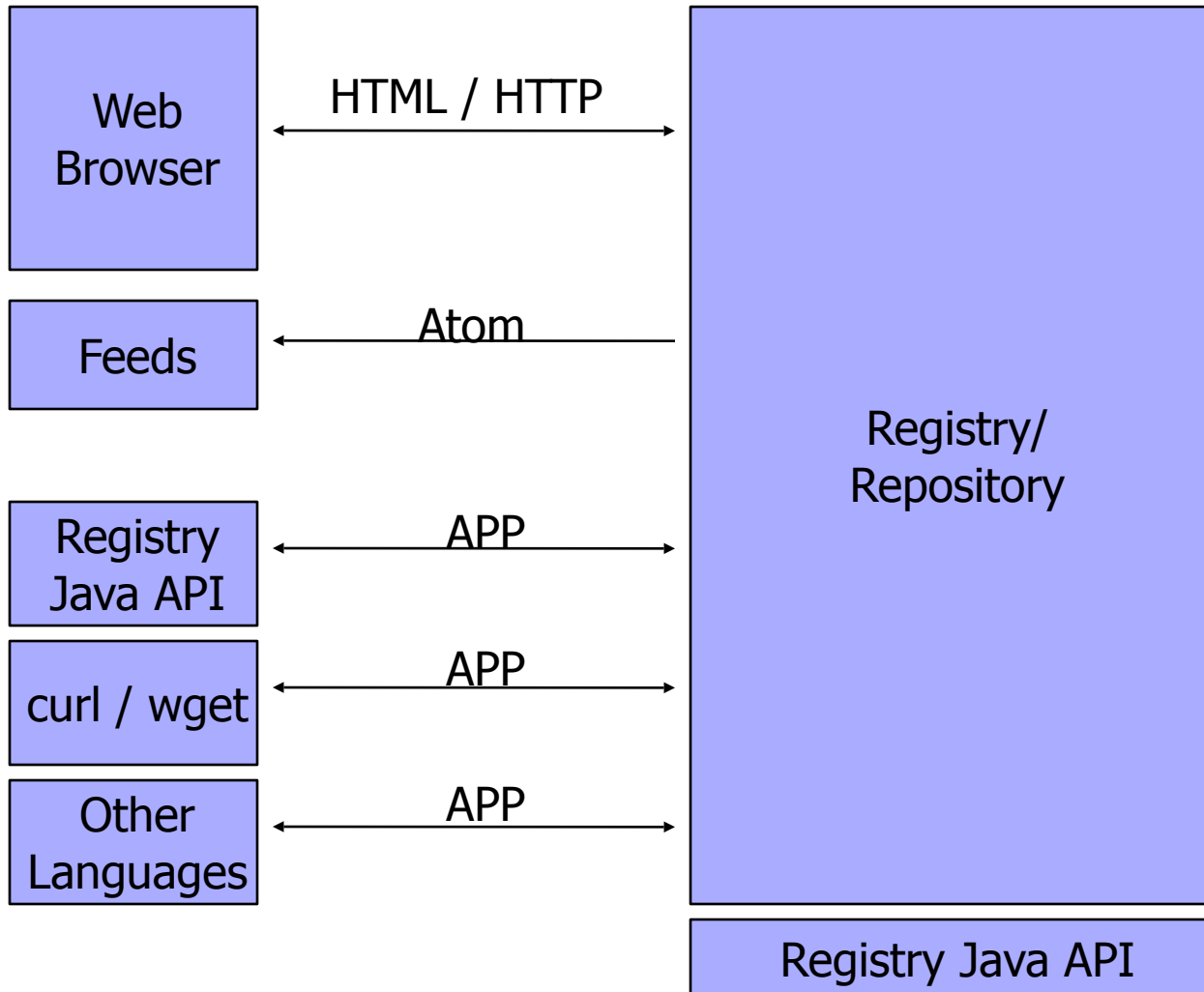
WSO2 Registry

An open source project that has tried to think about human and community issues as it tackles Enterprise SOA

- http://wso2.org/projects/registry
- Apache 2.0 license
- Open mailing list, wiki, JIRA, etc

Simple Atom Feed

```xml
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
 <title>Registry Blog</title>
 <link href="http://pzf.fremantle.org/registry/blog/"/>
 <updated>2008-02-07T15:15:02Z</updated>
 <author>
    <name>Paul Fremantle</name>
 </author>
 <id>blog-6003063374827736283.post-4039376056255567566</id>
 <entry>
    <title>Social Enterprise</title>
    <link href="http://pzf.fremantle.org/registry/blog/2"/>
    <id>blog-687787243798723.post-342798273498734</id>
    <updated>2008-02-07T15:15:02Z</updated>
    <content>
       <html>…</html>
    </content>
 </entry>
</feed>
```

The benefit of Atom

- You can "subscribe" with your Atom Feed Reader to *ANYTHING* in the Registry
  - When new versions of this service are deployed
  - When people comment on my service
  - When new services tagged "finance" are deployed

Atom and AtomPub

- Standard "feed" reading and writing capability
- AtomPub (Atom Publishing Protocol)
  - RFC 5023

- Service (1..1)
  - Workspace (1..n)
    - Collection (1..n)
      - Entries / Media Entries (1..n)

More on AtomPub

- Clear definition of behaviour of
  - POST, GET, PUT, DELETE
- For example, when you POST a resource to a collection
  - Specify a "Slug" header that defines the proposed name
  - The response 201 Created + Location header of new URI
- Benefits
  - A well-defined protocol
  - With interoperability, multiple clients, tools
  - But also accessible with curl, wget, etc
  - Does exactly what we needed (almost)
- Issues
  - There is some ambiguity about how to create a new collection
  - No definition of queries

AtomPub isn't just for Atom

- The AtomPub team defined clearly how you can create collections of Atom entries
- But also they define what happens if you POST other "stuff"
  - Other stuff == "Media Resources"
- Well defined behaviour when you post a Media Resource
  - Creates an Atom Entry with the metadata
  - Plus a link to the real resource

HATEOAS

- Atom has well defined link model
- An example:

```xml
<?xml version='1.0' encoding='UTF-8'?>
  <feed xmlns="http://www.w3.org/2005/Atom" xmlns:ns="tag:wso2.org,2008:foo">
    <parentPath xmlns="http://wso2.org/registry">/</parentPath>
    <link href="http://localhost:8000/wso2registry/atom/stuff" />
    <link href="http://localhost:8000/wso2registry/atom/stuff" rel="self" />
    <entry>
      <link href="http://localhost:8000/wso2registry/atom/stuff/
flatpackmediator.jar" />
      <title type="text">/stuff/flatpackmediator.jar</title>
      <updated>2008-03-13T11:19:39.512Z</updated>
      <link href="http://localhost:8000/wso2registry/atom/stuff/flatpackmediator.jar"
rel="self" />
      <link href="/stuff/flatpackmediator.jar" rel="path" />
    </entry>
  </feed>
```

How we defined our URLs

- Base URL
  - http://server/wso2registry/
- "Intermediate" paths
  - base/web
  - base/atom
  - base/resource
- Examples:
  - http://localhost:8080/wso2registry/web/services/finance/invoice.wsdl
  - http://localhost:8080/wso2registry/atom/services/finance/invoice.wsdl
  - http://localhost:8080/wso2registry/resource/services/finance/invoice.wsdl
- Three different views of the same resource
  - Note we didn't use the Accept model

How we defined our URL scheme

- /tags
  - Collection of all tags in the system
- /tags/[mytag]
  - Collection of all resources tagged *mytag*
- /resource/r1;tags
  - Collection of tags on resource r1
- /resource/r1;comments
  - Collection of comments on r1
- etc

Versions

- Every time a resource is updated we create a new version
- We keep track of dependencies between resources (e.g. WSDL <- Schema)
- Access versions
  - /resource/r1?v=4
  - /resource/r1;version
    - Collection of pointers to versions

Creating Collections

(or why Microsoft didn't use AtomPub – until they did)

- Not defined in AtomPub
- Spec says:
  - *This specification does not specify any request semantics or server behavior in the case where the POSTed media type is "application/atom+xml" but the body is something other than an Atom Entry. In particular, what happens on POSTing an Atom Feed Document to a Collection using the "application/atom+xml" media type is undefined.*

Queries

- ## Still work in progress
  - □ We want our backend to be flexible, but we haven't yet created our own Query Language

- ## Our current solution:
  - □ Store the backend specific query (e.g. SQL) as an entry in the Registry
  - □ Execute the query with parameters passed as HTTP GET parameters

Full definition

http://wso2.org/wiki/display/registry/Registry+Protocol

Java API

Registry reg = **new** RemoteRegistry(**new**
URL("http://localhost:8000/wso2registry/atom"), "admin",
  "admin");

Resource resource = reg.get("/services/finance/
  invoice.wsdl");
Object wsdl = resource.getContent();

Resource newCollection = **new** Resource();
newCollection.setDirectory(**true**);
newCollection.setAuthorUserName("admin");
reg.put("/stuff", newCollection);

paul@wso2.com

http://localhost:8080/wso2registry/web/

WSO2 Registry   (Untitled)

Resources   People   Activity   Sign out

WSO2 Registry

Good afternoon admin!

Advanced Search

About

GET HELP

User guide
Registry user guide

GET INVOLVED

Reporting Problems
Issues can be reported using the public JIRA.
Registry jira

NEWS

Online training **Introduction to WSO2 Registry.** Register now

Resource

<root>/

Created:	1h 14m ago	**Author:**	system
Last Updated:	0m ago	**By:**	admin
Media Type:	Unknown		
Permalink:	http://localhost:8080/wso2registry/web/?v=6		
Versions:	View versions		
My Rating:	☆☆☆☆☆		
Rating:	☆☆☆☆☆ (0.0)		
Properties:	⊕		
Description:			

Tags

Comments

Entries

Name	Created Date	Author	Rating		Action
system	1h 14m ago	system	☆☆☆☆☆ (0.0)		
todo2			1h 12m ago	admin	★★★☆☆ (3.0)	
Paul	1h 9m ago	admin	★★★☆☆ (3.0)		

What about WS-*?

- Focus on
  storing, searching, managing
  WSDL, Schema, WS-Policy

- Issues
  - Dependency links
    - WSDL imports Schema and Policy
  - Validity - is this WSDL valid? is it WS-I compliant?
  - Does it meet my corporate guidelines?
  - What stage of its lifecycle?
    - Test, System Test, Production, Deprecation
  - WS-* metadata isn't enough for the real world
    - Comments, Tags, Properties and Ratings add some simple real-life annotations that augment this

Content Handlers

- Whenever you POST or GET a WSDL we can intercept and run stuff
- For example, when we import WSDL
  - Also import the Schemas
  - Create internal dependency mapping
    - WSDL dependsUpon Schema
    - Schema isDependedUponBy WSDL
  - We are extending this to run WS-I validation
- We also support URL handlers
  - Allow you to extend the REST model of the Registry

Lifecycle handling

- **Version 1.0**
  - ☐ Properties

- **Version 1.1**
  - ☐ Better specification
  - ☐ Configure your lifecycle phases
  - ☐ Run handlers when lifecycle changes occur

So, what do I think about REST?

- I like REST a lot…
- but I'm still skeptical about REST
  - Even in this – the most obvious possible scenario – there are too many design choices to be made
  - Even after you subset to Atom/AtomPub there are still lots of non-standard design choices to be made
  - Still needed very smart people
- But this has worked out very well
  - In terms of building the Human Interaction and Social aspects
  - Unification of the human interface with the machine interface
  - Atom feeds

Get involved!

- Home page
  - http://wso2.org/projects/registry/
- Mailing List
  - registry-dev@wso2.org
- SVN
  - https://wso2.org/svn/browse/wso2/trunk/registry/
- Issue tracker
  - https://wso2.org/jira/browse/REGISTRY

Questions