CICS & SOA

What you don't know might help you

Will Yates – CICS Test Architect

#SHAREorg

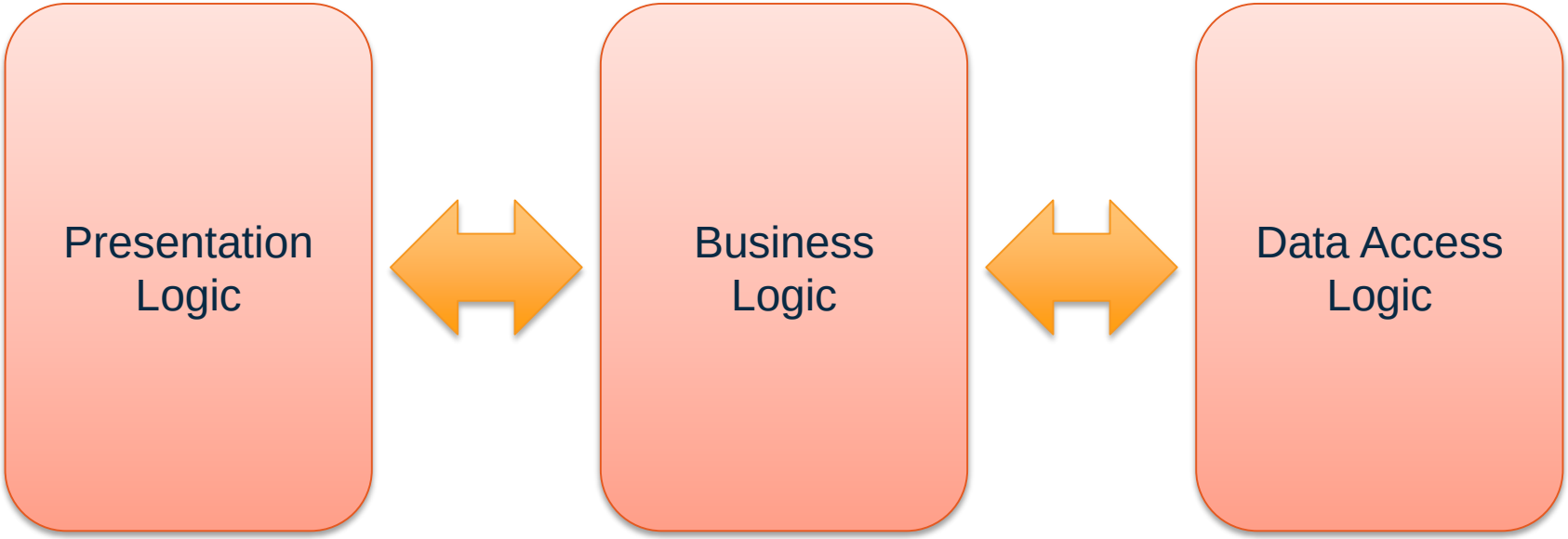
SHARE is an independent volunteer-run information technology association that provides **education**, professional **networking** and industry **influence**.

Copyright (c) 2015 by SHARE Inc. Except where otherwise noted, this work is licensed under http://creativecommons.org/licenses/by-nc-sa/3.0/

SOA is not just web services

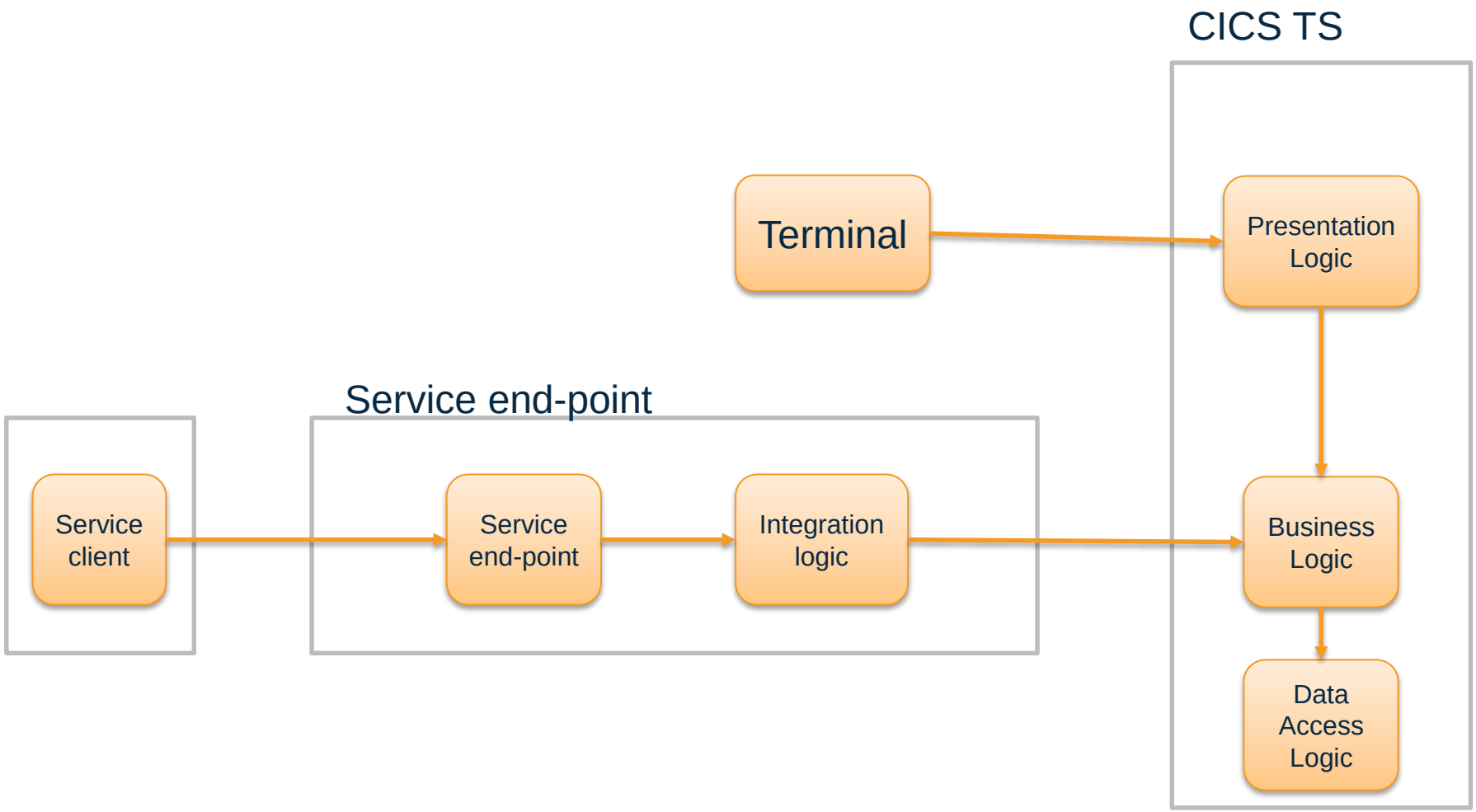
A service-oriented architecture (**SOA**) is an architectural pattern in computer software design in which application components provide services to other components via a communications protocol, typically over a network. The principles of service-orientation are independent of any vendor, product or technology.

CICS essential design concepts

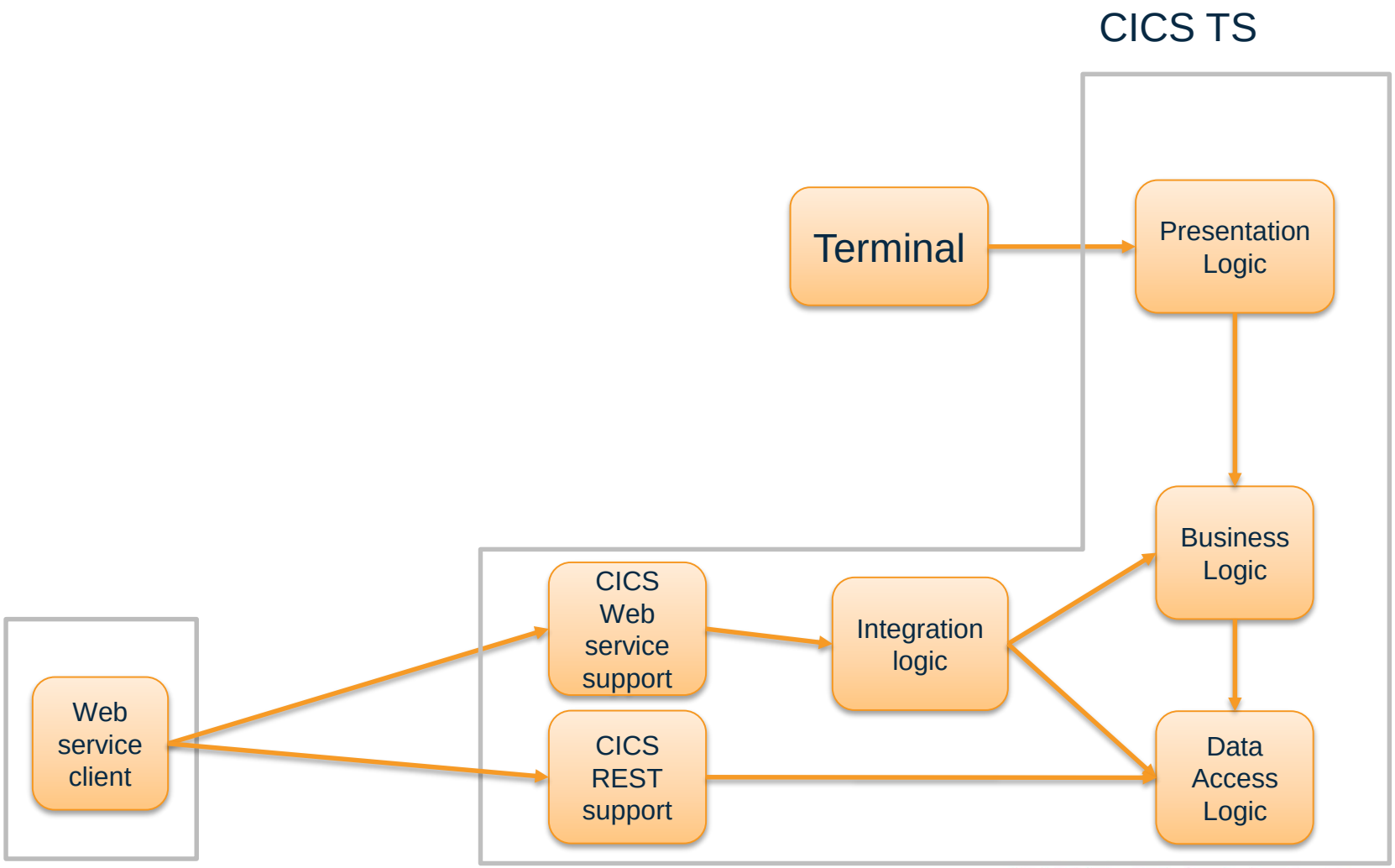


- Pseudo – conversational programming
- Communication Area – COMMAREA or Channels & Containers
- Program to program communication

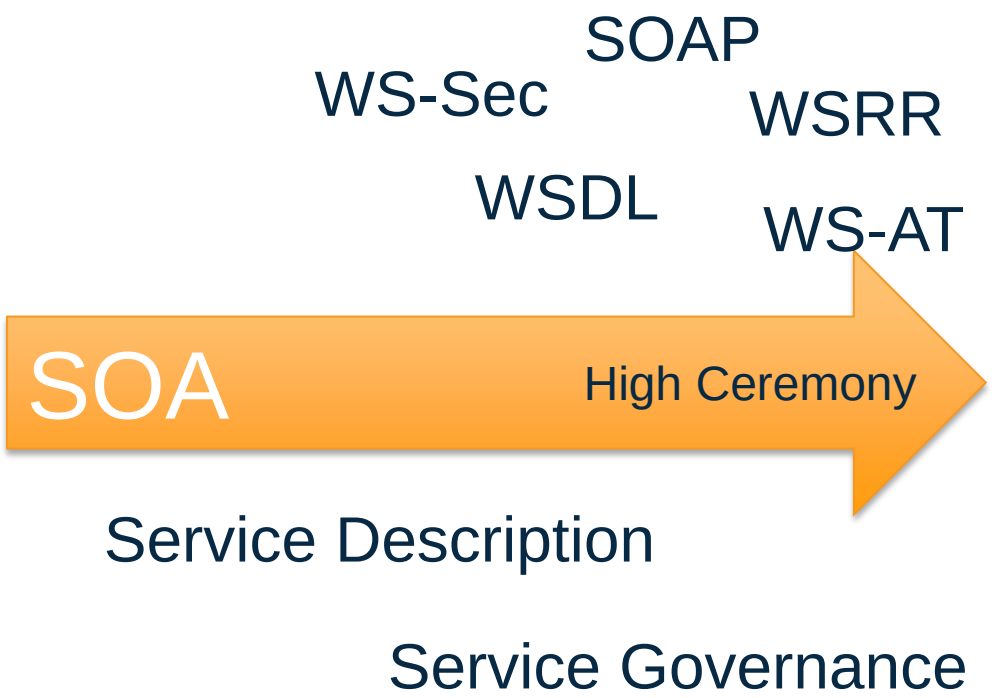
Two models of CICS SOA Integration



Two models of CICS SOA Integration



SOA Story: Enterprise SOA



SOA Story: Restful SOA

REST

HTTP

PHP

ATOM

SOAP

WS-Sec

WSRR

WSDL

WS-AT

High Ceremony

SOA

Situational Apps

Mobile Enablement

Resource Orientated

Service Description

Service Governance

SOA Story: CICS TS v3 era

REST

HTTP

PHP

ATOM

SOAP

WS-Sec

WSRR

WSDL

WS-AT

High Ceremony

SOA

Situational Apps

Mobile Enablement

Resource Orientated

Service Description

Service Governance

CICS TS v3

SOA Story: CICS TS v4 era

CICS TS V4.2
Feature Pack for Mobile Extensions

REST
HTTP
PHP
ATOM

SOAP
WS-Sec
WSRR
WSDL
WS-AT
High Ceremony

SOA

Situational Apps
Mobile Enablement
Resource Orientated

Service Description
Service Governance

CICS TS v4

SOA Story: CICS v5 era

- Language mapping enhancements to WSBIND

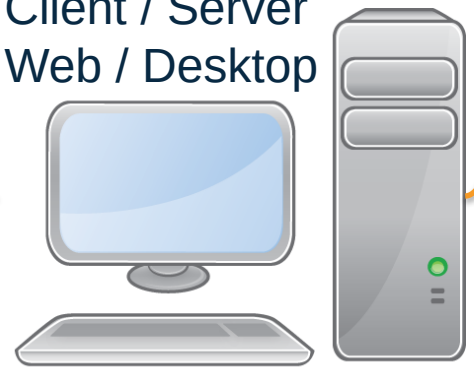
- New Base Function
  - Mobile Extensions Feature Pack

- New enablement function
  - Websphere Liberty Profile embedded into CICS runtime
    - Web enabled servlets
    - JAX-RS
    - JAX-WS

Mobile: Another stage in computing history

Host / Mainframe

Client / Server
Web / Desktop



Mobile / wireless / cloud

<u>Mobile is different:</u>

- Ubiquitous computing
- Mixed models
- A different type of app
- Revolutionary to business models

Mobile internet usage (UK 2014)

68% of adults used the internet ‘on the go’

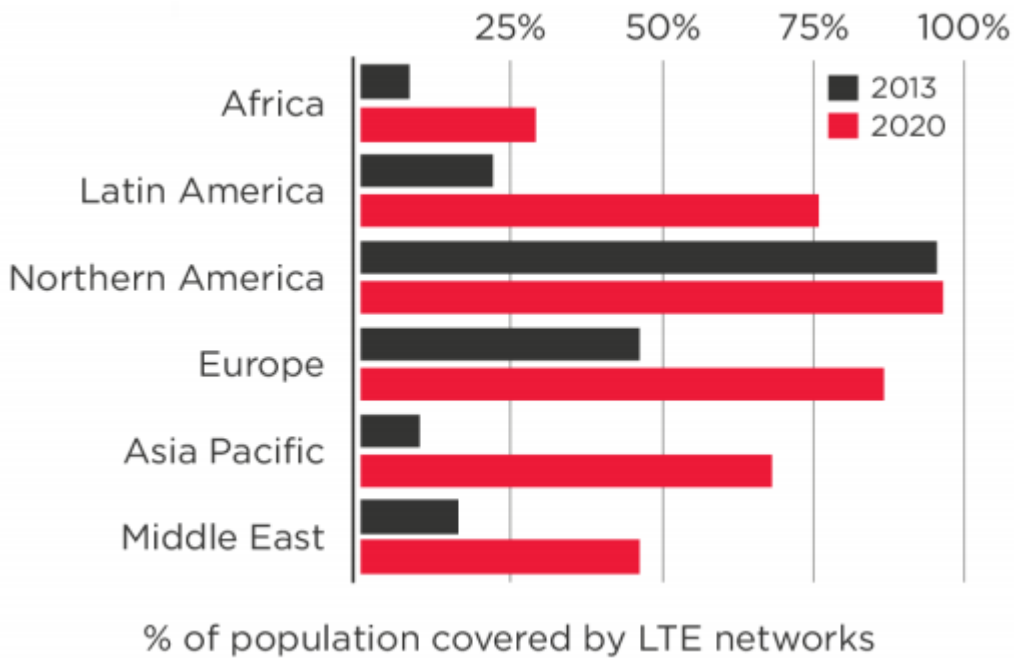
58% used a mobile phone or smartphone

43% used a portable computer

15% used an Ebook reader or other handheld device

Source: UK Office for National Statistics

Worldwide 4G network coverage



64%

Of the worlds population will be covered by 4G-LTE networks by the nd of 2020

Source: GSMA Intelligence

OK I get it mobile is huge!
But how does that affect which SOA technology I pick?
What choices do I have?

SOA pick and mix

JSON | SOAP | XML | HTML

Language Protocol

REST | WS - *

Service invocation

HTTP | HTTPS | Message Queue | CTG

Transport

SOA pick and mix

JSON | SOAP | XML | HTML

Language Protocol

REST | WS - *

Service invocation

HTTP | HTTPS | Message Queue | CTG

Transport

SOA pick and mix

JSON | SOAP | XML | HTML

Language Protocol

REST | WS - *

Service invocation

HTTP | HTTPS | Message Queue | CTG

Transport

SOA pick and mix

JSON | SOAP | XML | HTML

Language Protocol

REST | WS - *

Service invocation

HTTP | HTTPS | Message Queue | CTG

Transport

Language formats

- heavyweight, human readable, text based format for data
- More complex with custom types
- Requires dedicated parser

```
<SOAP><xml/></SOAP>
```

```
{
  "JSON": "REST"
}
```

- Lightweight, human readable, text based format for data
- Simple Structure
- Native JavaScript support
- Widely adopted by the industry

Not just JSON or SOAP

- Neither technology demands a particular data format
  - Raw XML
  - HTML
  - Binary Data
    - PDF etc

- HTTP media type can inform the client of the response data

- What can your client consume – who is the customer?

SOA pick and mix

JSON | SOAP | XML | HTML

Language Protocol

REST | WS - *

Service invocation

HTTP | HTTPS | Message Queue | CTG

Transport

Service invocation (REST or WS-*)

- The key motivator of a RESTful SOA is **simplicity** and **ubiquity**
  - Deliver content in the simplest possible way
  - HTTP is available everywhere; it's like the air around us
- With Enterprise SOA (WS-*), it's not the body (user data) that distinguishes the QoS, it's the ***richness*** of the WS-* headers
  - WS-Security is about choice in the decision of encryption, identity tokens and digital signatures
  - WS-Addressing is about transport-neutral mechanisms of describing addresses
  - Even the WS-I standards are about allowing maximum flexibility within a fully agreed-upon framework of standards
- You might want to choose WS-* where you don't have overall control of connectivity.
  - WS-ReliableMessaging and SOAP over JMS are about choice in how you obtain qualities of service

REST: Accessibility for Developers

- A RESTful Web service is formed like a sentence – it simplifies how developers access services
  - **Verb** = HTTP Action (GET, POST, PUT, DELETE)
  - **Noun** = the URI of the Service (the document)
  - **Adjective** = MIME type of the resulting document

verb	noun	adjective
GET PUT POST DELETE	http://server.com/users/100210	

REST: Accessibility for Developers

- A RESTful Web service is formed like a sentence – it simplifies how developers access services
  - **Verb** = HTTP Action (GET, POST, PUT, DELETE)
  - **Noun** = the URI of the Service (the document)
  - **Adjective** = MIME type of the resulting document

verb | noun | adjective

GET http://server.com/

```
<operation>
        <function>getCustomerDetails</function>
        <customerNumber>100210</customerNumber>
</operation>
```

JSON vs SOAP/XML web services

- JSON is lightweight?
  - Complexity of message structures
    - A complex message in JSON representation will be more costly to process than a simple message in XML representation
    - HTTP(S) connectivity base-line cost is equivalent

JavaScript and JSON work together well by design out of the box

XML is very powerful but needs requires a dedicated parser

- Versioning of a service can be costly with SOAP/XML
  - Rolling out changes to back-level clients can be costly
    - Regenerate and deploy WSDL
- JSON schemas are more forgiving e.g. default values fields

Merging Enterprise SOA & RESTful SOA

- Developing for both Enterprise SOA and RESTful SOA enables the exploitation of distinct content pools:
  - Services generated inside your enterprise
  - Services generated outside the enterprise
- Utilizing the best of both worlds allows you to take advantage of all the communities served by your business

Attributes and Advantages of a RESTful SOA

- **Simplicity**
  - Many decisions pre-made, constrained choices
    - Protocol (HTTP), encryption model (HTTPs)
    - Identity token exchange (Basic-Auth, or standard HTTP schemes)

- **Ubiquity**
  - Use the HTTP infrastructure and other technologies like JavaScript that already exist

- **Effortless use of services**
  - Single, well-understood programming model (Javascript)
  - Lots of examples on the web
    - Copy-cut-and-paste programming to use services
  - Someone should be able to use a RESTful SOA Service without knowing they're doing it!

- **Cacheability, Scalability, Testability, Secureability, Navigability**

Summary

- SOA is here to stay
  - Exposing services not just functions
  - SOAP like JSON and REST is just an enablement technology
- Importance of choosing a technology to expose your services that can be consumed by the client
- CICS implements key technologies to allow you to ‘take your pick’ to build your SOA strategy