

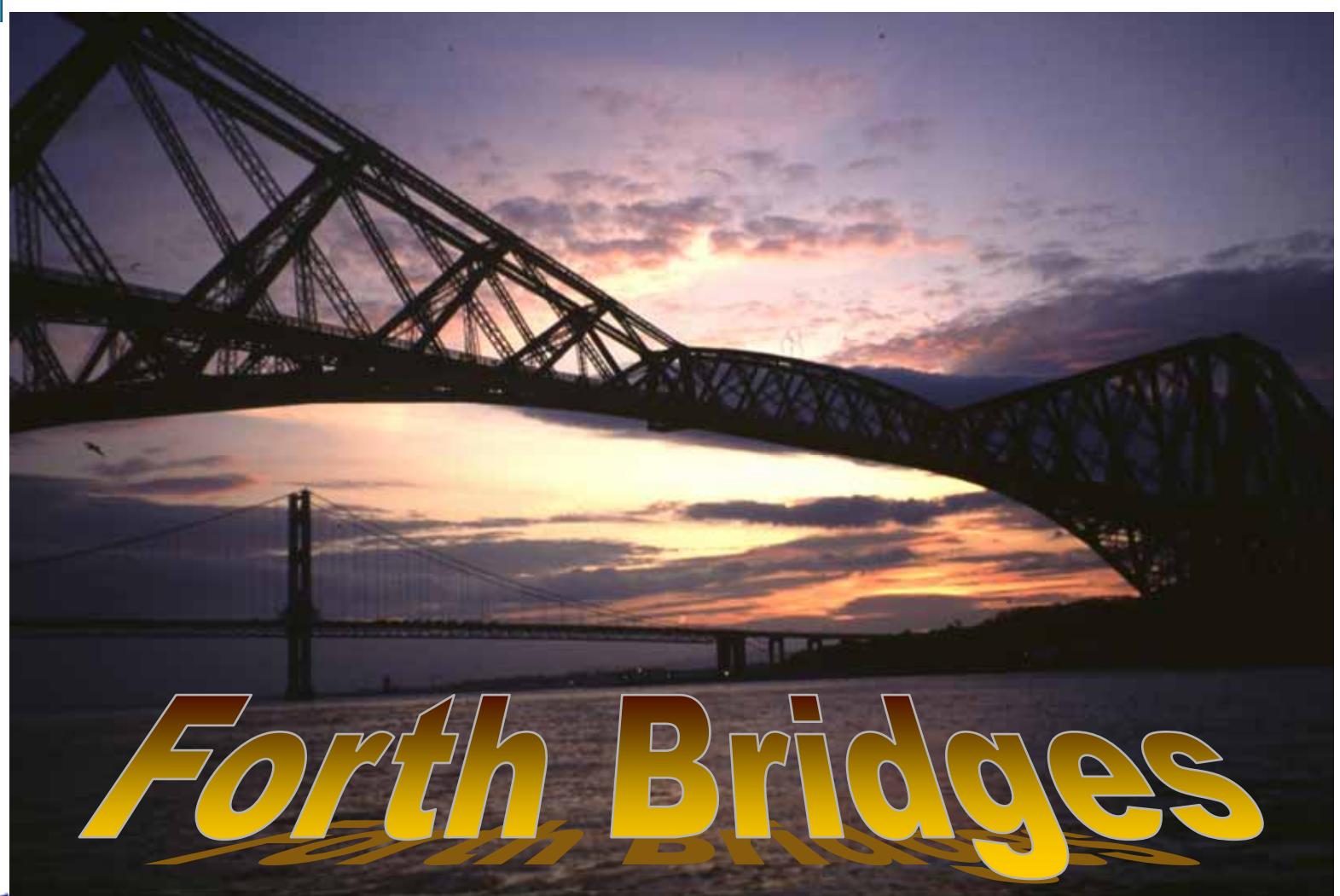


# Web Services & Grid Services

Presented by David Fergusson  
Slides by Richard Hopkins  
Training Outreach and Education  
Edinburgh e-Science

[rph@nesc.ac.uk](mailto:rph@nesc.ac.uk)





# *Forth Bridges*

# Grid-Related Standards

## W3C – World-Wide Web Consortium

- Industry non-profit organisations individuals.
- Best known for **fundamental web standards**

## DMTF: Distributed Management Task Force

- Industry
- management standards and integration technology.
- Best known for standards that address **system management** in enterprise and Internet environments

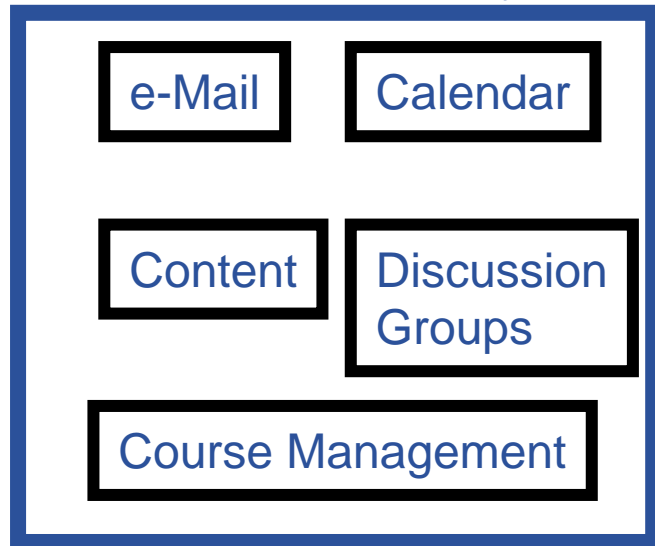
## OASIS: Organization for the Advancement of Structured Information Standards

- Vendors users academics governments;  
Organizations individuals industry groups
- Best known for **e-business** standards that address real world business requirements

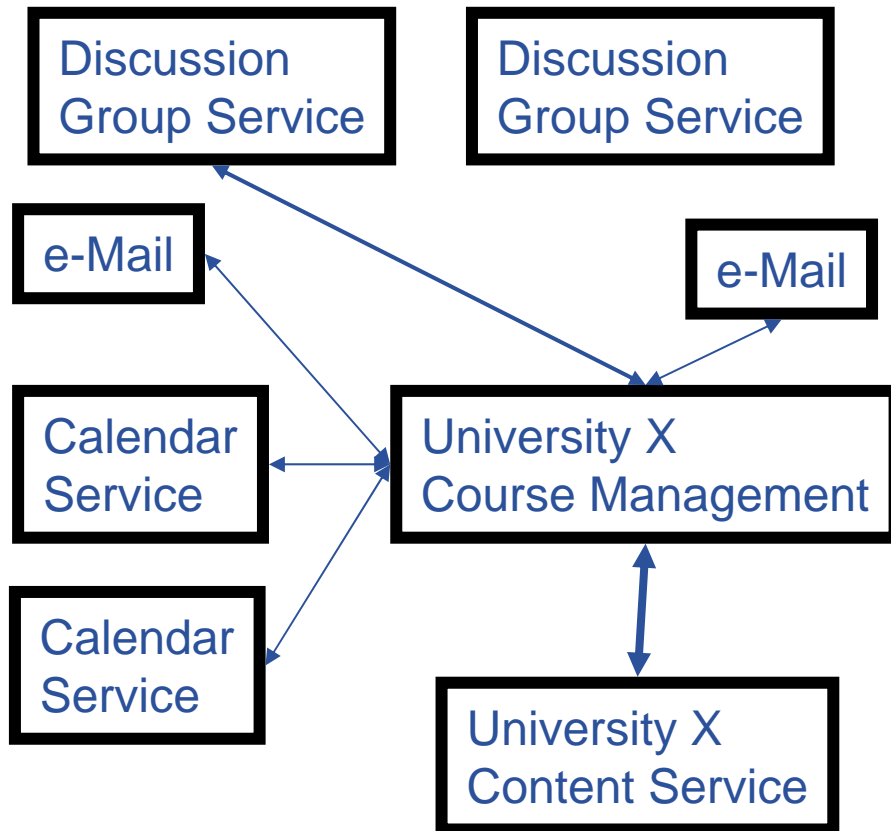
## GGF : Global Grid Forum → OGF : Open Grid Forum

- International community leading the global standardization effort for grid computing.
- Founded in 2000
- Members include
  - users, developers, and vendors.
  - Industry, academics, research laboratories
- **Best known for standards and architectures for Grids, including:**
  - OGSA (Open Grids Services Architecture)
  - GridFTP
  - BytelO
  - JSDL, BES (Basic Execution Service)
  - HPCP (High Performance Computing Profile)
  - ...

## University X Staff/Student Support System



Bundled collection of Proprietary  
Web-based Applications –  
Lock-in  
Jack-of-all-Trades



Loosely coupled collection of Services  
Core competency focus

- **Web Services are software components that are..**
  - Accessible across a network
  - Defined by the messages they receive / send
  - Loosely coupled
    - Web services framework supports Autonomous Evolution
      - *So can change service implementation without changing interfaces*
      - *Can evolve interface with forward & **backward compatibility***
  - **Interoperable: each service has a description that is accessible and can be used to create software to invoke that service - WSDL**
- **... and based on standards**
  - Built on (extensions of) standards made ubiquitous by the Web: http(s), XML, ... and for which tools are therefore built.
  - Developed in anticipation of new uses
  - central - XML SOAP WSDL
- **Briefly - A service is a software system designed to support interoperable machine-to-machine interaction over a network**

## The killer argument for not bothering with grids

- It's easier just to use my own cluster

The killer argument for not bothering with commercial airlines

- It's easier just to buy my own jet

The killer argument for not bothering with scheduled train services

- It's easier just to run my own train

The killer argument for not bothering with public roads

- It's easier just to build my own roads

Works for some VIPs

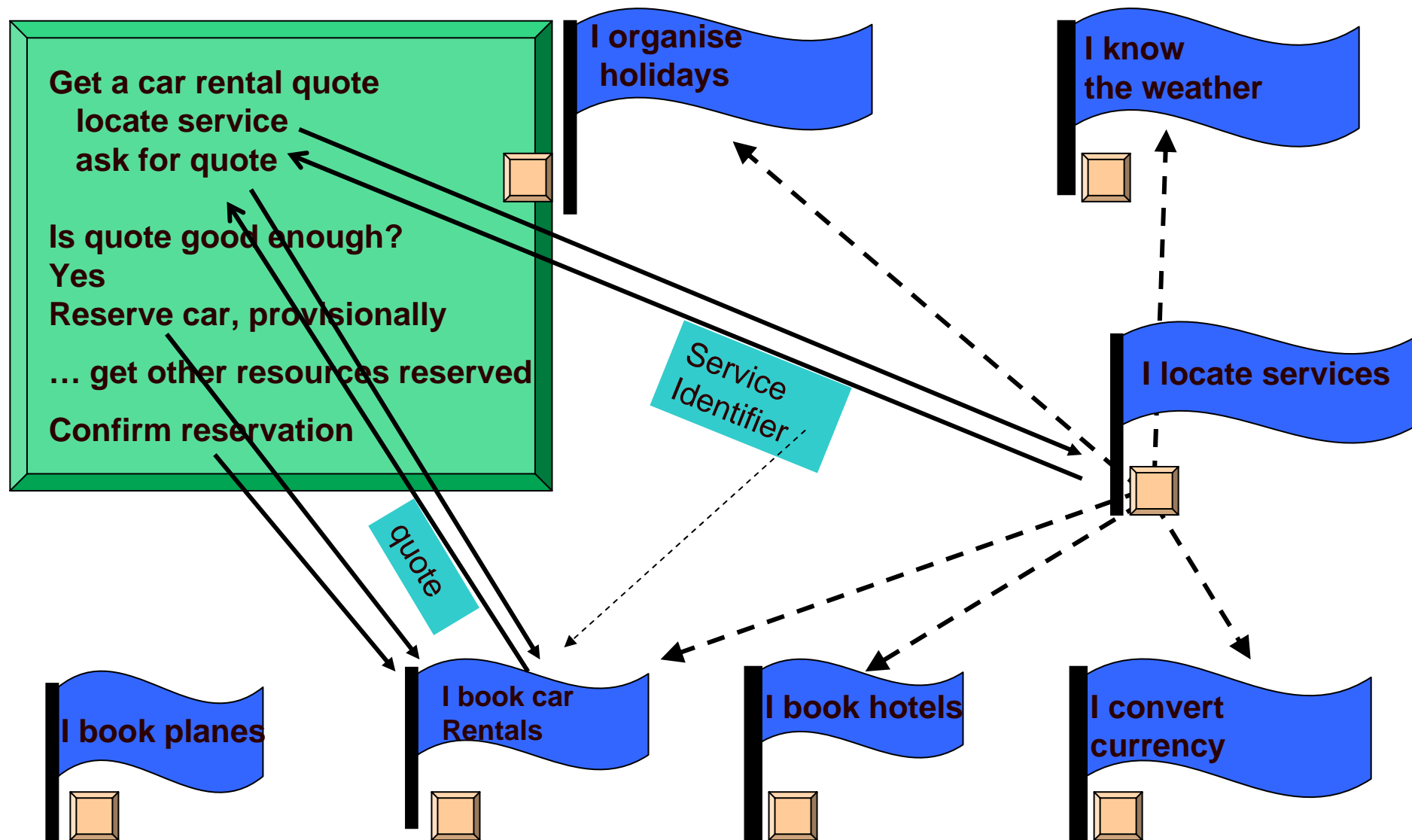
Works for one V-VIP

Works for nobody -  
you can't build a road  
to everywhere  
you might want to go

**FUNDAMENTAL INFRASTRUCTURE MUST BE SHARED**

- **Orchestration of services**
  - *Can be heterogeneous*
  - *Are owned by different organisations*
  - *geographically distributed*
  - Data
  - Instruments
  - Applications
- **Processor Power**
  - if your peak need is much higher than your average need
- **Collaboration**
  - if you want to jointly develop and use applications

- **History –**
  - the next stage in Electronic inter-enterprise interaction
  - 0. e-mail – human interaction at both ends
  - 1. Web browsing – human interaction at client end
    - 1.1 Static web pages
    - 1.2 Pages with dynamically generated content
  - 2. Web Services – human interaction at neither end
    - Applications that are invoked by software clients
- A model is automated web browsing - Human travel agent provides “organise holiday” service by surfing the web to look for and invoking services – book a hotel; book a plane; book a car hire; ....; confirm bookings of best options to meet client needs.
- The aspiration of Web services is to provide a framework that allows that same model to be used in writing an application –
- which is itself becomes an “organise a holiday” service, finding and using useful services







Fundamental abstraction –

- Object
- Service

– **Where are they the similar /different**

• Motivations

- **Encapsulate Implementation**
- **Low design coupling**
- **Autonomous Evolution**
- **Component re-use**

• Structure - class / instance / inheritance

- **Objects – central to the model**
- **Services –**
  - weak notion of class/instance (“porttype” corresponds to class)
  - Inheritance only as add-on



- Universe-unique identifier that can be communicated
  - **URL / Object reference**
- Life-cycle pattern
  - **Object**
    - Created by factory, on user request
    - Exists till garbage collected
    - Formal class/instance/inheritance structure
  - **Service**
    - Created out-of-band
    - Life-cycle autonomy (can't do garbage collection)



- Interaction model
  - **Both have operation invocation, but -**
    - Objects exchange object references – everything is an object
    - Services exchange documents (particularly XML)
- Data Format
  - **Object – data format is encapsulated**
    - Can only be accessed by its Home S/W; gives either
      - *Keep home S/W at home*
        - Fine granularity
        - Tightly-coupled communication
      - *Or*
        - Shipping the code (APIs)
        - Compatibility issues
  - **Service – data format is standardised for communication**
    - Less robust
    - Have extensibility features



- Need to achieve effective cooperation even though
  - **the different services are produced by different organisations, without any design collaboration, on different platforms (interoperability)**
  - **the services are autonomously evolving**
- Loose Design Coupling – minimum prior shared information between the designer of the two components of an interaction
  - **Dynamically accessible Machine processable Meta data**
    - Self-describing data in standard format – **XML** documents
    - Description of structure of communications – **SCHEMAS** (types)
      - *With extensibility*
    - Service description – **WSDL**
  - **Means for obtaining it – from a repository, using UDDI standard**
  - **Communication protocol that supports this – SOAP**
  - **Everything is a SCHEMA-described XML document – soap message, WSDL definition, schemas themselves (meta-schema)**

- Universal form for information exchange – now the ABC of computing
- Textual
  - No proprietary data encoding
  - Human readable (in a decent editor) – particularly for de-bugging
- Self-describing – meta-data

**<Sparse\_Matrices:TwoDMatrix>**

**<Sparse\_Matrices:TwoDElement x="7" y="10">**

**<Variable\_Precision\_Reals:Number> 1234.469 </>**

**</>...**

**</>**

**</> end-tag is personal short-hand**

- Bloated - but can do attachments
- Extensibility - mixed languages; >1 namespace in one document :
  - **Sparse\_Matrices – s/h for unique namespace URI**
  - **Variable\_Precision\_Reals - – s/h for unique namespace URI**



- An XML document needs a machine-processable syntax definition –
  - Schemas – assumed in web services
  - DTDs – deprecated
- Well-formed vs. Valid – validating against a schema
- A schema is itself an XML document
- Schema defines a type structure –
  - **as in a programming language. Slightly strange**
- Geared to document parsing – e.g. white-space facet for simple types
- Verbose
- Used in standards specifications –
  - **Most standards involve some data structure defined by a Schema**
- Has wildcards for extensibility
  - **For SparseMatricies Schema - TwoDMatrix consists of**
    - 1 or more twoDElements, each having
      - *x and y integer attributes for its location*
      - *A variable-precision real number for its value*
    - Followed by anything else

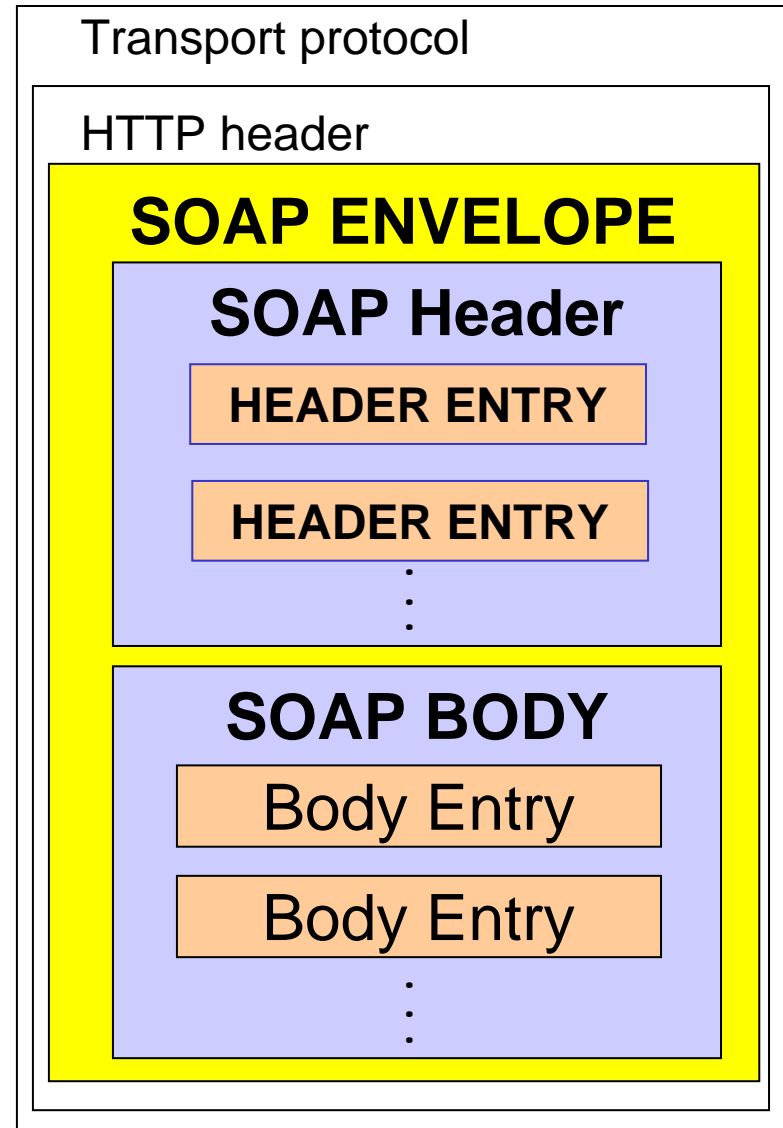
- An XML document describing the exposed interface of a service— what the consumer sees
- Constitutes a contract with the client - provides a specification of what is offered by the service provider which can be relied on by the service consumer
- What – the service interface, defined in terms of message exchanges –
  - **Syntax of interactions, e.g.**
    - Message 1 from consumer to service – service request
    - Message 2 from service to consumer – response
    - Similar to a remote procedure call
  - **Syntax of message**
    - XML schema
- How – the representation of the messages on a particular protocol
- Where - URL

- **Name**
  - **Originally – Simple Object Access Protocol**
  - **Temporarily – Service Oriented Architecture Protocol ?**
  - **Now (SOAP 1.2) – Not an acronym**
- **Purpose**
  - A extensible protocol to enable the exchange of
    - structured and typed information
    - between peers
    - in a decentralised, distributed environment
- **Status**
  - SOAP 1.2 – <http://www.w3.org/TR/soap12-part0>
    - W3C recommendation, June 2003
  - **SOAP 1.1** – <http://www.w3.org/TR/NOTE-SOAP-20000508>
    - W3C submission May 2000 – but that’s what people use currently

- **XML based**
- **Higher order Protocol –**
  - Built on some underlying protocol - binding
    - Extensibility – can define binding for any underlying protocol
    - Usually HTTP – a specific standard extension
- **Single Message Protocol**
  - Defines standard for a single message communication
  - Multi-message conversations require a means to associate one message with another
    - Via underlying protocol (e.g. use of same connection)
    - Via the application (specific message-id information as part of the soap message)
- **Multi-stage message processing –**
  - The soap Processing model
    - Different headers targeted at different processing stages

## Each SOAP message will have:

- **Outer layer(s) for underlying protocols**
- **Envelope (XML root element)**
- **Header (optional)**
  - Multiple header blocks/entries
  - For different purposes – factorisation
  - For different processing stages
    - Actors
- **Body (mandatory)**
  - The payload
  - Zero or more XML elements
  - May be a Fault element
    - Specific fault reporting standard



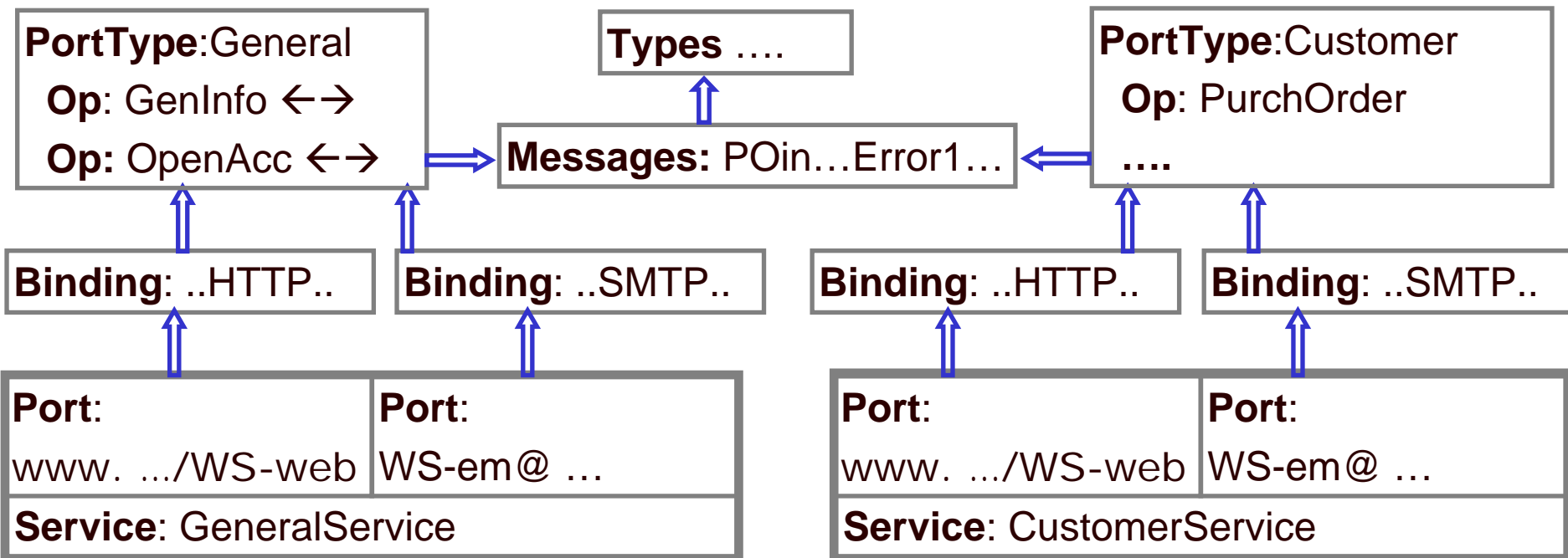
- **Review**
- **SOAP**
- **WSDL – Details** ←
- **Tools**
- **Practicals**



- An XML document describing the exposed interface of a service– what the consumer sees
- Constitutes a contract with the client - provides a specification of what is offered by the service provider which can be relied on by the service consumer
- Supports separation of concerns to allow common definition and re-combination
- What – the service interface, defined in terms of message exchanges
  - Syntax of interactions, e.g.
    - Message 1 from consumer to service – service request
    - Message 2 from service to consumer – response
    - Similar to a remote procedure call
  - Syntax of message
    - XML schema
- How – the representation of the messages on a particular protocol
- Where - URL
- Service Invocation requires ability to send/receive the defined messages
  - Interoperable
  - No application specific APIs – generated from WSDL

- **Example**
- **Company Provides two types of Service (PortTypes)**
  - General Service
    - Get general information (GenInfo)
    - Open an Account (OpenAcc)
  - Customers Service (being a “Customer” = having an account)
    - Purchase Order (PurchOrder)
    - Invoice (Inv)
    - Payment Advice (PayAdv)
    - Get Statement (GetStmt)
    - Notify overdue payment (Overdue)
- **Both over two kinds of binding**
  - Web - HTTP
  - Email - SMTP

- PortType – a logical interface
  - **Operations** - each is sequences of message types
  - **Message type is a data structure** - schema
- Binding, e.g. “SOAP over HTTP” – details of representation
- Service – defines one or more ports, each with
  - **Location** – URL – here sharing of locations
  - **Binding and thus portType**



PORTTYPE – an interface comprising a set of operations

- Organisation of functionality into portTypes and operations is similar to O-O design
- A portType is a coherent unit of exposed functionality – operations make sense together
  - **E.g. Currency conversion might be a service used in processing customer transaction**
  - **But would not expect a convertCurrency operation for this service**
- But ... larger granularity than O-O
- ... Deployment considerations
  - **Split between General and Customer may be that say General has a wider range of available bindings/locations**
- Each operation declares a number of messages which can be communicated as the interface to the operation
- These messages conform to one of four message exchange patterns – input/output sequencing ....

**PortType:Customer**

**Op:** PurchOrder

**In:** POin

**Out:** POout

**Fault:** Error1

....

**Op:** Overdue

....

**Op:** Inv

....

....

**PortType:General**

**Op:** GenInfo

....

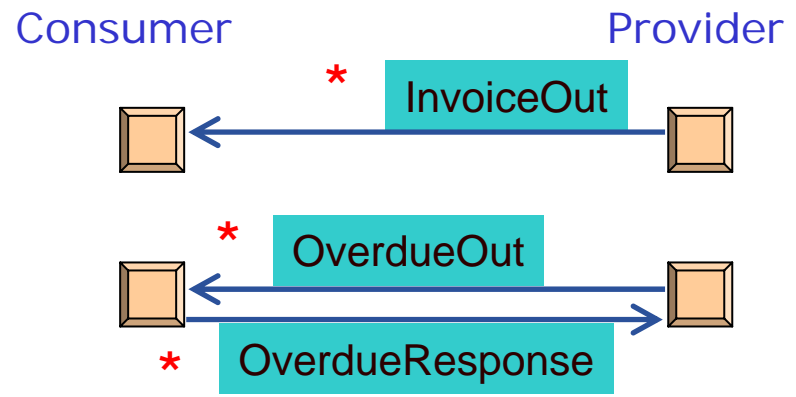
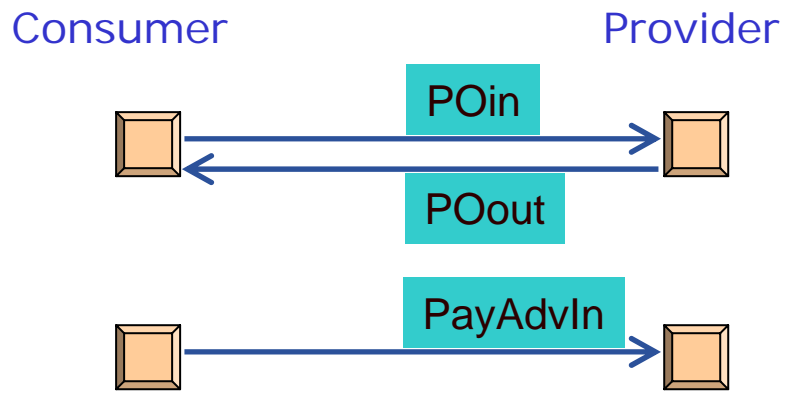
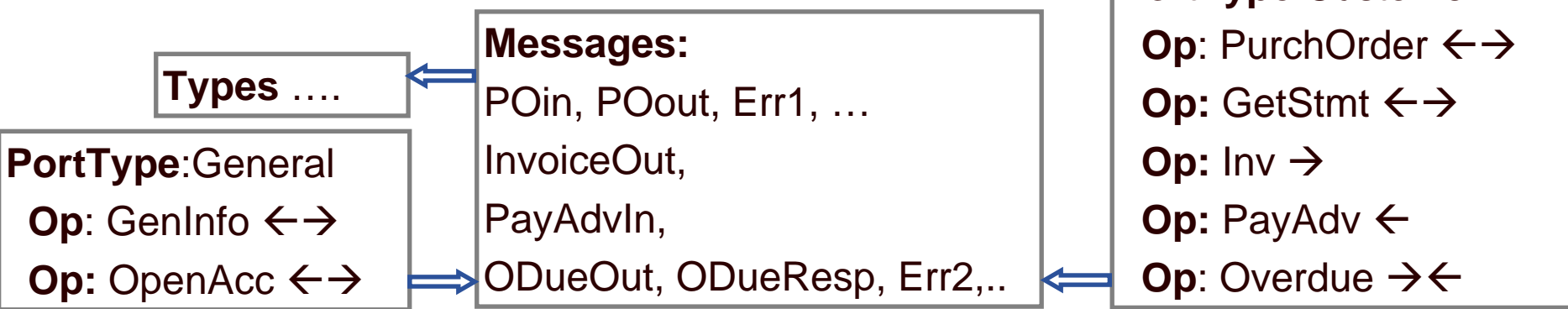
**Op:** OpenAcc

....

## Four patterns

- ↔ IN then OUT Request/Response – most usual
- OUT Notify \*
- ← IN One-way
- ← OUT then IN Solicit/Response \*

- \* Reversed roles
- Service Provider  
proactive = client
- Service Consumer  
reactive = server



## <wSDL:definitions

```

name="myWSDL"
targetNamespace=".../me1"
xmlns:tns=".../me"
  xmlns:soap=" ... /soap"
  xmlns:xsd=".../schema"
  xmlns:wSDL=" ... /wSDL"

```

```
xmlns:me2=".../me2" >
```

```
<wSDL:import namespace=www.me1
location=".../me2" >
```

```
<wSDL:import namespace=".../soap"
location=".../soap.wSDL" >
```

...

```
<wSDL:types> .... </>
```

```
<wSDL:message ...>...</><wSDL:message ...>...</>
```

```
<wSDL:portType ...>...</><wSDL:portType ...>...</>
```

```
<wSDL:binding ...>...</><wSDL:binding ...>...</>
```

```
<wSDL:service ...>...</><wSDL:service ...>...</>
```

```
</>
```

Brief documentation

For self-reference,  
Inter-reference

Standard definitions

Referenced definitions  
to allow a structure  
of related WSDLs

Import –  
requires associated  
namespace declaration

0..\* import

0..1 types  
Contains  
1..\* schema

0..\* message

0..\* portType

0..\* binding

0..\* service









- \*\*\* W. Vogels, **Web Services are not distributed objects**, Dec 2003.  
<http://weblogs.cs.cornell.edu/AllThingsDistributed/archives/000343.html>
- \*\* S. Wilson, K. Blinoo, D. Rehak, **Service Oriented Frameworks**,  
[http://www.jisc.ac.uk/uploaded\\_documents/AltilabServiceOrientedFrameworks.pdf](http://www.jisc.ac.uk/uploaded_documents/AltilabServiceOrientedFrameworks.pdf)
- \*\* Hao He, **What is S-O Architecture?**, Sept. 2003.  
<http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html>
- \*\* David Orchard, **Extensibility, XML vocabularies and XML Schemas**, October 2004  
<http://www.xml.com/pub/a/2004/10/27/extend.html>
- \* Dare Obasanjo, **W3C XML Schema Design Patterns: Dealing with Change**, July 2002  
[http://www.xml.com/pub/a/2002/07/03/schema\\_design.html](http://www.xml.com/pub/a/2002/07/03/schema_design.html)
- \* **XML Schemas : Best Practices** <http://www.xfront.com/BestPracticesHomepage.html>
- \* WC3, **Web Services Architecture**, Feb 2004. <http://www.w3.org/TR/ws-arch/>
- \* Sun Microsystems, **Web Services Made Easier: The Java APIs and Architectures for XML**, Technical White Papers, June 2002, Revision 3. <http://java.sun.com/xml/webservices.pdf>

- **Review**
- **SOAP**
- **WSDL – Details**
- **Tools ←**
- **Practicals**



## Axis is a “SOAP engine”

- converting between –
  - **JAVA objects/method invocation/returns**
  - **SOAP / XML “on-the-wire”**
  - **= serialising / de-serialising**

## Provides -

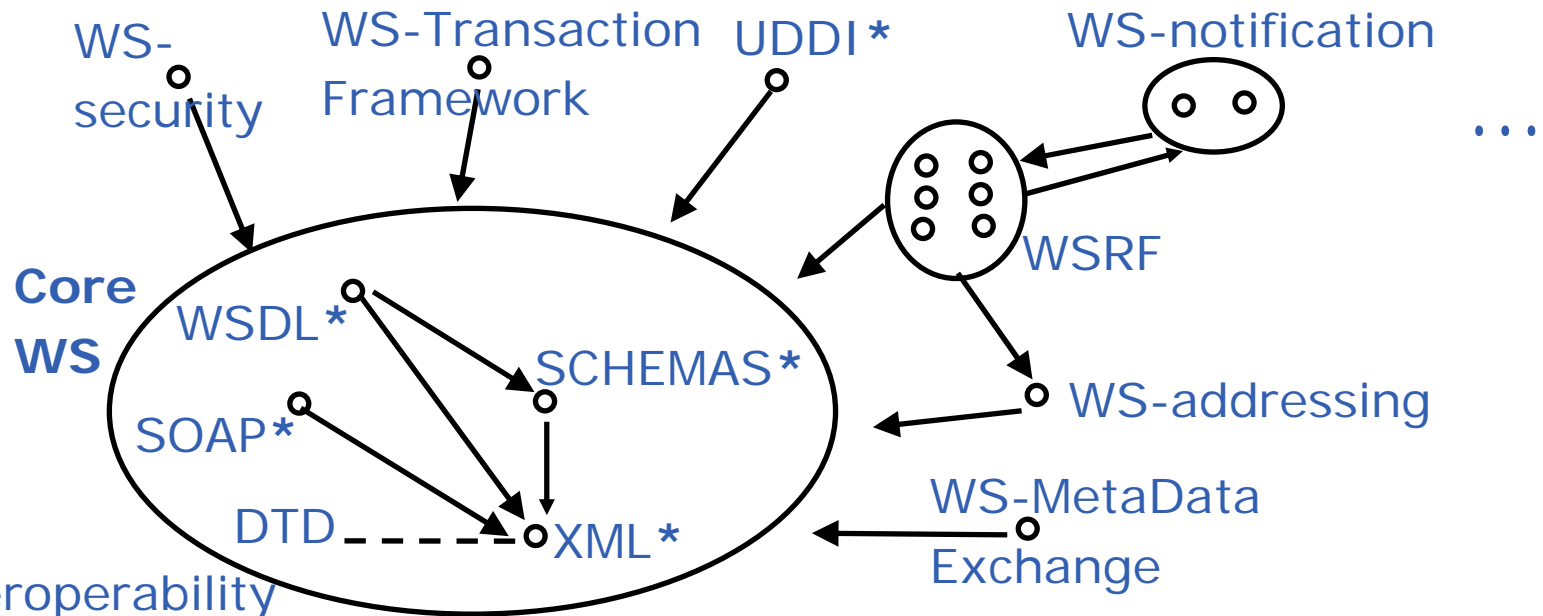
- **Simple stand-alone server**
  - No deployment descriptor needed
    - **very easy to do**
    - **no control**
- **a server which plugs into servlet engines such as Tomcat**
  - which is what we are doing
- **support for WSDL - WSDL2Java; Java2WSDL**
- **tool for monitoring TCP/IP pack**

## architecture allows extensibility –

- **custom processing (e.g. for particular SOAP headers)**
- **general framework for protocols handlers**

- **Message Exchange Patterns**
  - Not just request-response
  - Asynchrony of the others
- **Improved XML parsing**
  - streaming
  - faster
  - less memory
- **Modules for extensibility**
- **Archive Based Deployment**
  - hot deployment

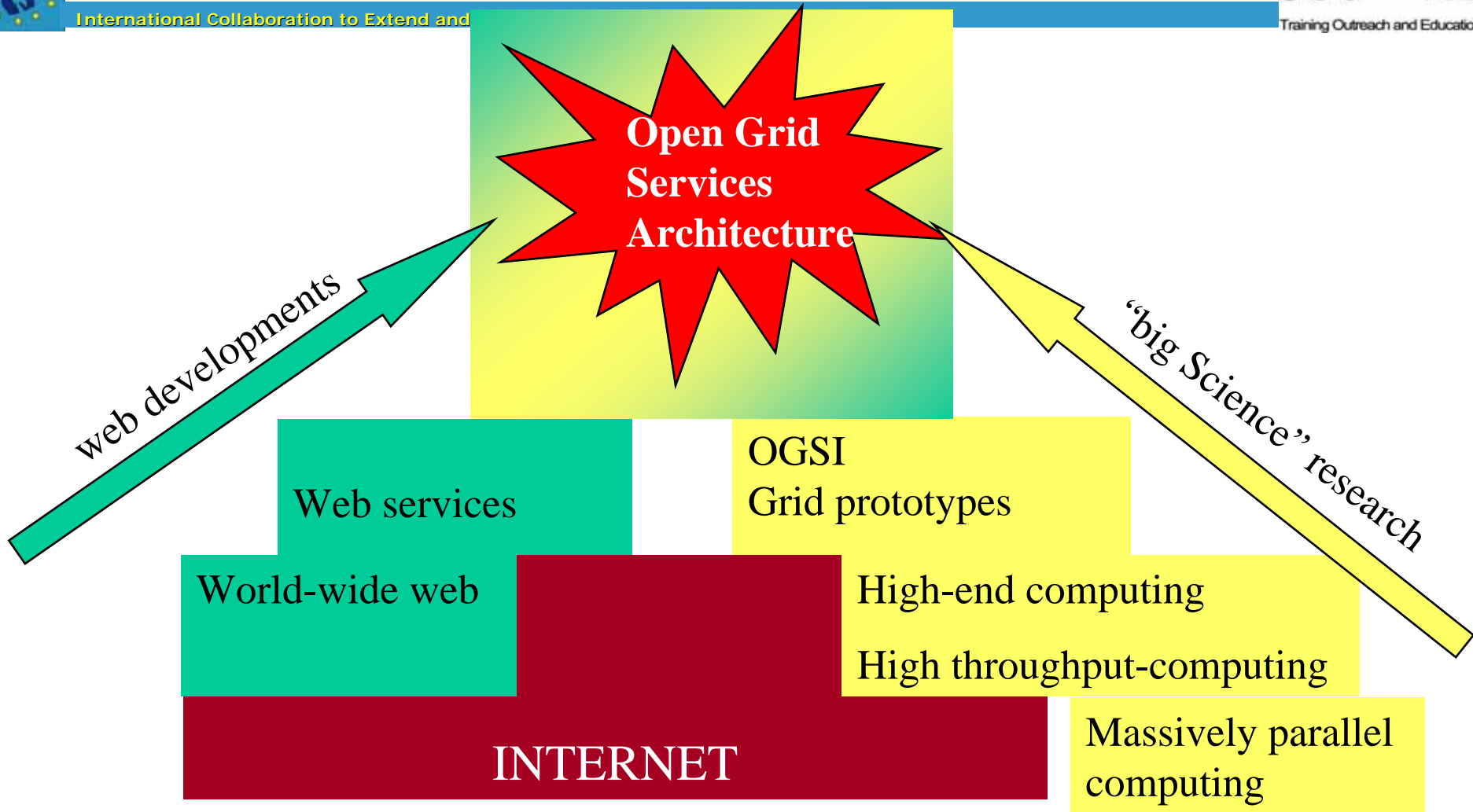
- Two main standards bodies –
  - W3C – actually produces “recommendations” – web community
  - OASIS – industry – IBM, Microsoft, Sun, ....
- These standards are factored to allow partial adoption and combination
  - The core standards
  - **WS-I** – clarifications to aid interoperability – <http://www.ws-i.org>
  - Higher level standards built on them



\*WS-Interoperability

- **Standards are emerging... some near acceptance and some being discarded**
  - For a summary see <http://www.innoq.com/soa/ws-standards/poster/>
- **Production grids are based on de-facto standards at present**
  - Inevitably!
  - GT2 especially
  - But locks a grid into one middleware stack unable to benefit from the diverse developments of new services
- **Some confusion remains after the OGSI era**
  - Many projects sidestepped this by using “pure” WS
- **Current way people try to create grid middleware is using Service Oriented Architectures based on WS, with WSRF**
- **Initial implementation based on WS-RF and OGSA is in Globus Toolkit 4**
- **Globus Toolkit 4 has been released - We'll soon have experience to test the perception that this is the way to go!**

- **“Open grid services architecture” OGSA– proposed in 2001**
- **Open Grid Services Infrastructure (OGSI)**
  - Globus Toolkit 3 resulted
  - Specified in 2003
- **Then in January 2004, OGSI to be replaced by emerging WS-RF**
  - NOTE: OGSA still under development (GGF)
- **Imbalances in OGSI that are addressed by WS-RF(OASIS)**
  - “Too O-O”
  - monolithic standards
  - WS community not engaged
- **Some confusion remains after the OGSI era**
  - Many projects sidestepped this by using “pure” WS (WS-I)
- **Globus Toolkit 4 has been released – see how it goes**
  - Doing very well – likely to be adopted by NGS



**Web Services and Grids have some similarities and some differences -**



## similarities

- Interaction across Organisational Boundaries
- Interaction between globally distributed components
- Interaction within a changing environment
- Interoperability required due to heterogeneity

## Web Services

- Mainstream
- Organisational Independence
- Coordination of
  - Application Logic
  - To give complex services
- Service Abstraction
- Short-lived Interactions

## Grids

- Specialised (as yet)
- Partial cooperation – VOs
- Co-ordination of resources
  - Applications (monolithic)
  - Processors / Storage / Instruments
- Virtual Computer Abstraction
- Persistence –
  - Infrastructure
  - Computation
  - Data
  - People



# GRIDS + WS = Grid Services

International Collaboration to Extend and Advance Grid Education

## Exploit commonality

### Efficiency of common solution to common problems

- Interaction across Organisational Boundaries
- Interaction between globally distributed components
- Interaction within a changing environment
- Interoperability required due to heterogeneity

## Integrate Differences

- Mainstream vs Specialised

### Make the Specialised a special-case of the Mainstream

- “inheritance” of
  - Existing architectural solution (present & future)
  - Industrial strength tools (present & future)
- Enables
  - Greater take-up of Grids
  - Wider integration of grids with other kinds of web Services
  - Integration of
    - *Grid as distributed virtual computer*
    - *Service approach to forming complex composite applications*

## Exploit commonality

### Efficiency of common solution to common problems

- Interaction across Organisational Boundaries
- Interaction between globally distributed components
- Interaction within a changing environment
- Interoperability required due to heterogeneity

## Integrate Differences

- Service Abstraction vs Virtual Computer Abstraction
  - Construct the Virtual Computer components as services
  - Retro-fit web-services wrappers
- Short-lived interactions vs persistence
  - Add persistence framework to WS and use that for grids
- Organisational independence vs co-operative VOs
  - Make VO services

## Web Services

- Short-lived Interactions

## Grids

- Persistence –
  - Infrastructure
  - Computation
  - Data
  - People

Need to add to basic web services the notion of persistency

**STATEFULL SERVICES**

WS-RF : Web Services Resource Framework ( OUTSIDE WS-I )

## Web Services

## Grid Technology

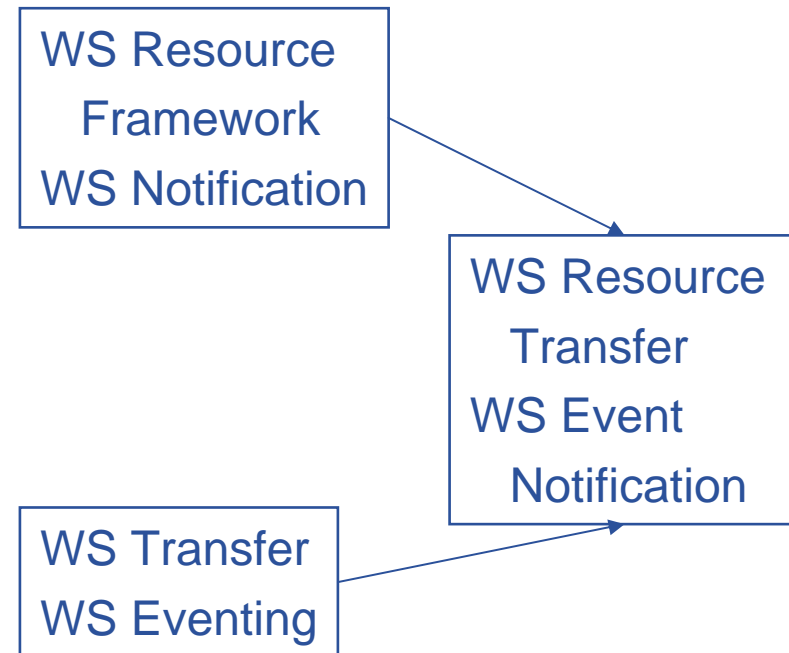
## Grid Services

- Commerce
- Standards
- Tools

- Research driven
  - Data-intensive
  - Compute intensive
  - Collaboration – sharing of resources
- Trust:  
opening resources

infrastructure for the information society

- **Resource**
  - Existence
    - creation
    - identifier
    - deletion
    - lifetime
  - Resource Properties – state elements
    - XML representation
    - get
    - put
    - partial get
    - partial put
- **Event Driven**
  - Subscribe to a topic
    - itself a resource
  - Notify a topic-relevant event



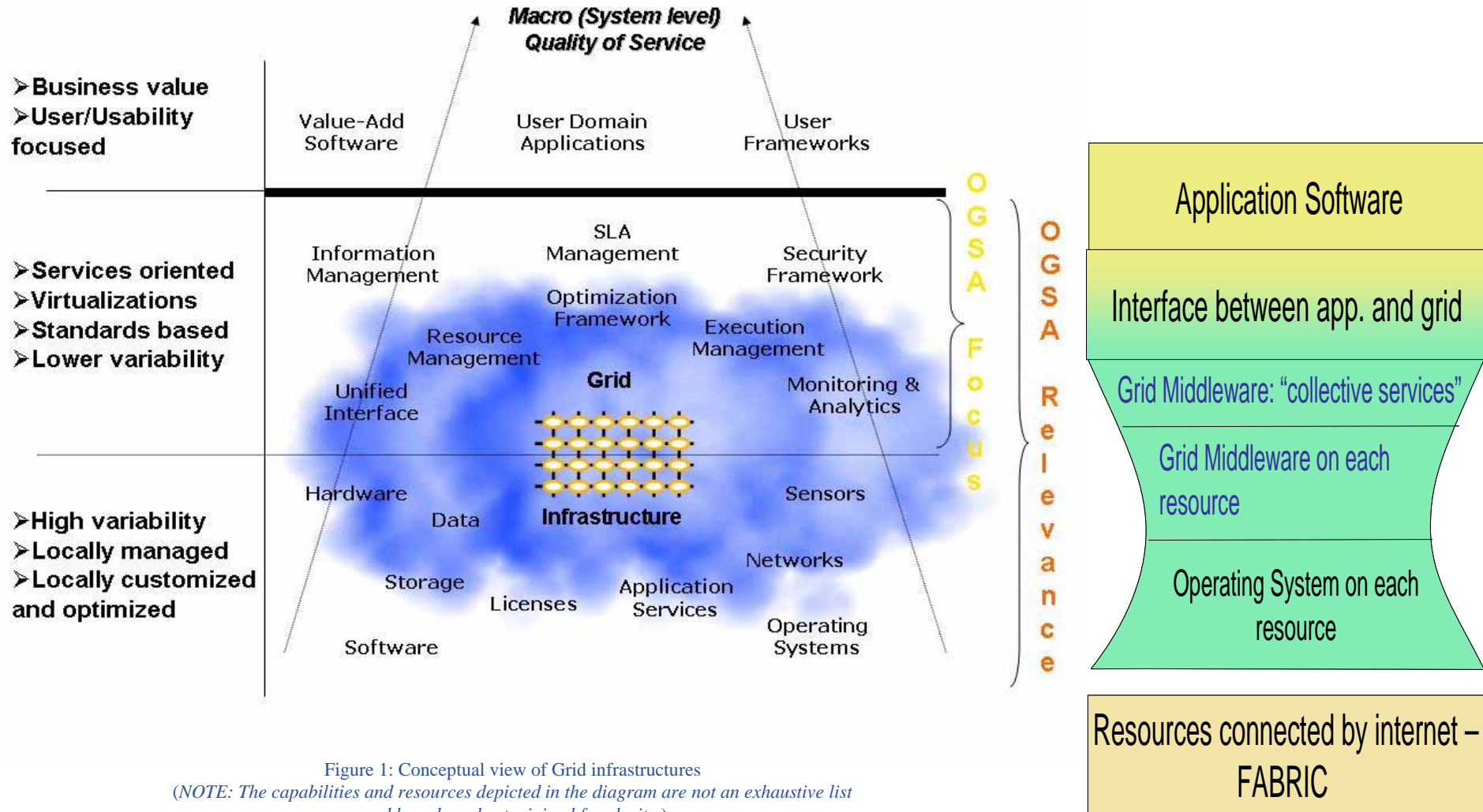
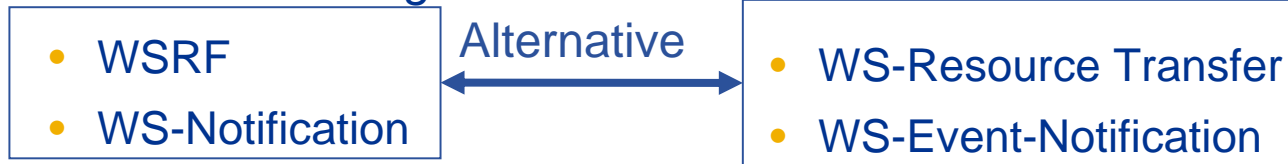


Figure 1: Conceptual view of Grid infrastructures  
 (NOTE: The capabilities and resources depicted in the diagram are not an exhaustive list, and have been kept minimal for clarity.)

- Aim is to establish standard functionality domains and associated standard specifications
- Infrastructure Services – Base Profiles

- Resources

- *WS-Addressing*



- ***WS-I Basic Profile***

- XML
- SOAP
- WSDL

- Secure Channel

- *WS-Security; XML-signature; WS-I Basic Security Security Assertion Mark-up Language (SAML)*

- Anonymous Channel

- *WS-I Basic Security*

# ABOUT HALF WAY

***TAKE A SHORT BREAK !***

***Later WSRF leading to Globus Toolkit 4 services***

## Goals

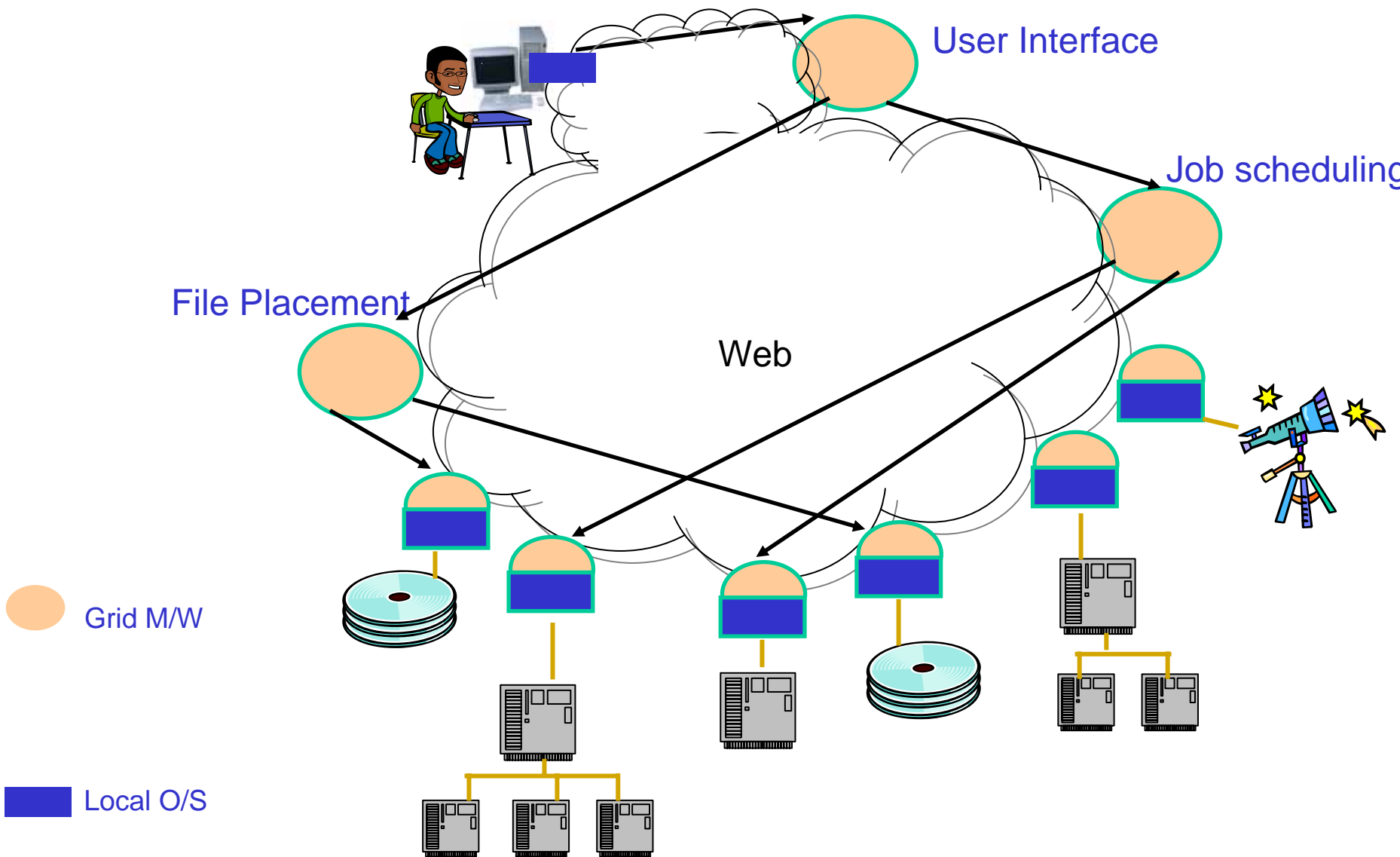
- An orientation to Web Services and to their role in Grid computing

## Content

- Web Services
- **Web Services and Grids** ●
- WSRF
- GT4

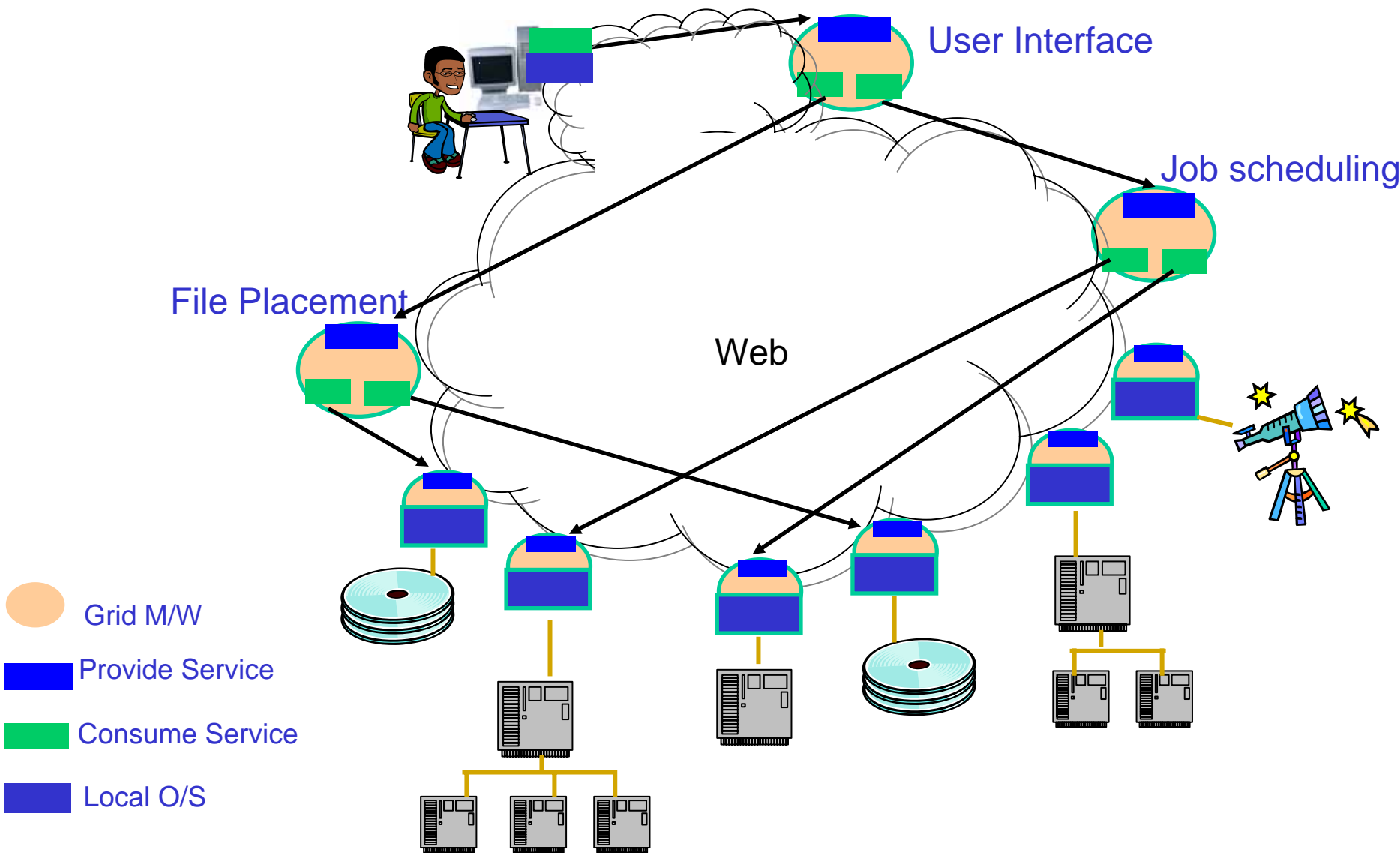
# Re-Package Grid Middleware

International Collaboration to Extend and Advance Grid Education



# Re-Package Grid Middleware

International Collaboration to Extend and Advance Grid Education



- **Grid Services = Web Services**
  - + Stateful resources (WSRF)
  - + Event-driven client activation (Notification)

## WSRF - A stateful (WS) resource ...

- **Is a repository for persistent state**
  - Like an object in an object-oriented architecture

## WS-N -The Notification-based Interaction pattern – Event Driven

- **Subscribe to a “Special Offers” Notification Service**
- **Notification Service sends a communication to me, I can reply with “buy it”.**

## Goals

- An orientation to Web Services and to their role in Grid computing
- No prior knowledge assumed

## Content

- Web Services
- Web Services and Grids
- **WSRF** ●
- GT4

## A stateful (WS) resource

- **Is a repository for persistent state**
  - Like an object in an object-oriented architecture
- **Has state that Comprises a set of state data**
  - Each item of state data is a resource property
  - E.g Account for running jobs on a remote machine
    - Resource Budget
    - Lifetime of Account
    - ...
  - E.g. Job
    - Time so far
    - Estimated Completion Time
    - ....
  - A resource property is expressible as an XML document,
    - which can in principle be retrieved and updated
  - Have a Resource Properties document which is a representation of all the exposed state



## A stateful (WS) resource

....

- **Has a well-defined life-cycle – creation and destruction**
  - Destruction can be
    - Immediate – explicit user request
    - Scheduled – “lifetime” : temporal garbage collection
- **Can be known and acted upon by one or more Web Services**
  - Has a globally unique identifier –
    - <http://www.company/JobService#Ac7>
    - Can be passed between services to identify the resource
    - Resource-qualified EPR (end-point reference)
- **Is associated with one or more web services, providing interface for manipulating it**
  - A WS-resource comprises:
    - its service
    - the resource itself

## A stateful (WS) resource

....

- is accessed using the “The Implied-Resource Pattern”
  - Implied Resource –  
An association between message exchange and the particular resource – an implied input to the operation
    - Implied – the resource identifier is NOT an explicit parameter in the request
    - Implicit association is either
      - *Static – association is made when the web service is deployed – 1:1*
      - *Dynamic – association at time of message exchange – which can be as a property of the address. Could be as a header.*
  - Pattern - **A set of usage conventions on existing technologies**

## Differences between

- Resource in Web services Architecture
- Object in Object-Oriented Architecture
- **Explicit State**
  - Resource State can be expressed as an XML document which in principle is retrievable and updatable
  - An object is defined by the operations on it and the affect of those operations on future operations
    - Could have an object state which is not representable as a document
      - An irrational number with operations:
        - *get/set n-th digit in decimal notation;*
        - *set to algorithmically-defined irrational, e.g. “pi”*
  - The type of a WS-resource is
    - the type of its resource properties document
    - not the signatures of its operations

- **Scheduled Destruction**

- Can say this resource may self-destruct after 3 days
- In O-O destruction is by either
  - Explicit or implicit in some user action
    - *(but in web, consumer service to do the action may be gone)*
  - Garbage collection
    - *(but un-realistic for web)*

- **Looser Encapsulation**

- In O-O an object has one interface
- In WS a resource can be operated on by several services
  - With several different interfaces –
  - the type of the resource is that of its properties doc, not the signatures of its operation

- **In WS, a resource needs to be at a coarse granularity**



- **Out-of-Band operations**

- In O-O everything happens in one coherent framework –

- Other than for the initial object

- Every object is created by the single initial object or an already created object
    - Every change to an object's state is a result of an operation performed on it by some other object deriving from the initial object

- In WS

- Creation and Modification can happen by non-WS mechanism
      - *Human intervention*
    - Services seem to spontaneously appear and disappear

- **Fault tolerant - In WS partial (permanent) failure is expected and accommodated; Partial failure is a permanent condition**



- **PortType** has to point to type definition of resource properties document
  - `<portType name="XPortType" wsrp:ResourceProperties="tns:XResourceProperties">`  
`<operation name="Xop"> ... </operation>`
  - This is what tells you the service supports a resource
- **GetResourceProperty** operation (mandatory)
  - To retrieve value of a single resource property
  - Mandatory if there is Resource Properties attribute
- **GetMultipleResourceProperties** operation (optional)
  - To retrieve values of several properties – important for granularity considerations
- **SetResourceProperties** operation (optional)
  - Provide a sequence of changes – Insert / Update / Delete
- **Note** – no standard Factory operation
  - Just a pattern
  - Need for flexibility in how resource instances created



- **If**
  - the Immediate Destruction Pattern is supported
- **Then**
  - that is done by a standard operation
    - wsrl:Destroy
- **If**
  - the Scheduled Destruction Pattern is supported
- **Then**
  - Resource Properties must include some specific properties concerning lifetime
    - `<xsd:element name="CurrentTime type="xsd:dateTime"/>`  
helps consumer determine clock difference
    - `<xsd:element name="TerminationTime nillable="true" type="xsd:dateTime"/>`
  - Must have specific operation
    - wsrl:SetTerminationTime
- **Flexibility of conditional standards, significance of “pattern”**



- **WS-Notification is draft standards dealing with the**
  - The Notification-based Interaction pattern – Event Driven
- **Subscribe to a “Special Offers” Notification Service**
- **Notification Service sends a communication to me, I can reply with “buy it”.**
- **Communication is back-to-front**
  - Initiated by Service Provider (client)
  - Received and Responded to by Service Consumer (server)
- **WSDL has message exchange patterns for this**
- **In grids – e.g. run a job, get notification of job termination**
- **Relation to WSRF**
  - A subscription is a WS-resource, must follow WSRF standards
  - A resource service can do notification –
    - to notify consumers of changes in state of the resource
      - *Value change*
      - *Destruction*
    - If it does this pattern then must follow WS-Notification standards

- **Model - Subscribing to a Notification service on some topics**
  - **E.g. My boss (Subscriber) informs a press-cutting service (Publisher) that it is to notify me (Consumer) of articles on WebServices (Topic) appearing in the popular press (Producer)**
  - **Topic Space - a forest of topic Trees**



- **Publisher – distributes notification messages according to subscriptions**
- **Producer – generates notification messages for Consumers**
- **Can combine Producer and publisher - same service generates the event and sends it to the subscribers; otherwise Publisher is a Broker**
- **Can separate them – producer generates the notification and sends it to a broker who distributes it according to subscriptions**
- **Subscriber creates a subscription for a consumer in a Publisher**
- **Consumer receives notification messages (may combine with subscriber)**



- **WSRF builds on**
  - **WS-Addressing – W3C Candidate Recommendation**
  - **WS-Notification**
- **WSRF comprises standards**
  - **WS-ResourceLifetime – OASIS working draft**
  - **WS-ResourceProperties - OASIS working draft**
  - **WS-RenewableReferences – who knows?**
  - **WS-ServiceGroup – OASIS working draft**
  - **WS-BaseFaults – OASIS working Draft**
- **WSRF supports –**
  - **WS-Notification**
    - **WS-BaseNotification – OASIS Public Review Draft**
    - **WS-BrockeredNotification – OASIS Public Review Draft**
    - **WS-Topics – OASIS working draft**

- Resource

- Existence

- creation
    - identifier
    - deletion
    - lifetime

- Resource Properties
  - state elements

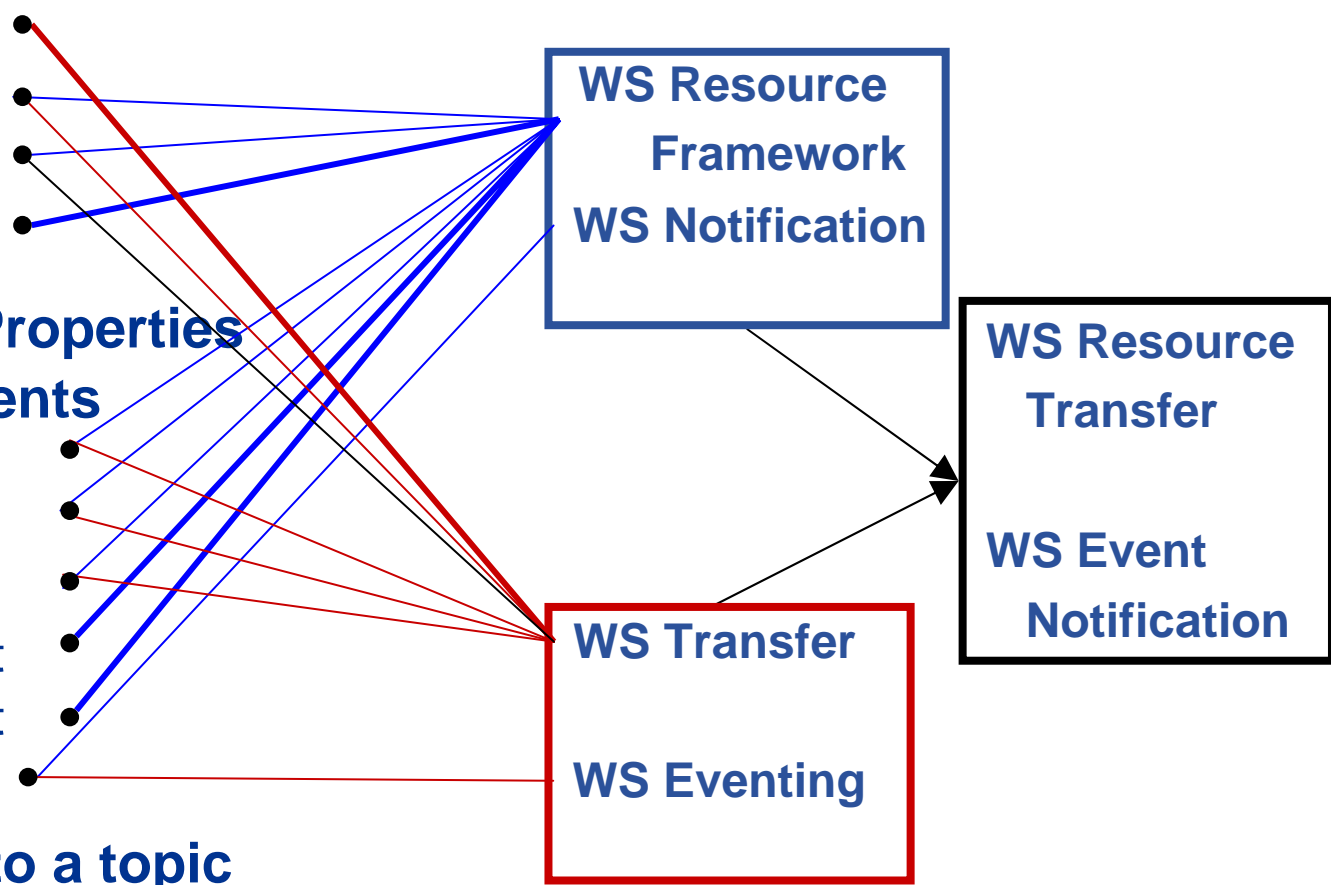
- XML rep.
    - get
    - put
    - partial get
    - partial put

- Event Driven

- Subscribe to a topic

- itself a resource

- Notify a topic-relevant event



Web service itself

(Front end)  
is stateless

Freely have multiple instances that come and go –  
Scalability  
Reliability

Maintains state in a back-end

Ac7 is a WS-resource

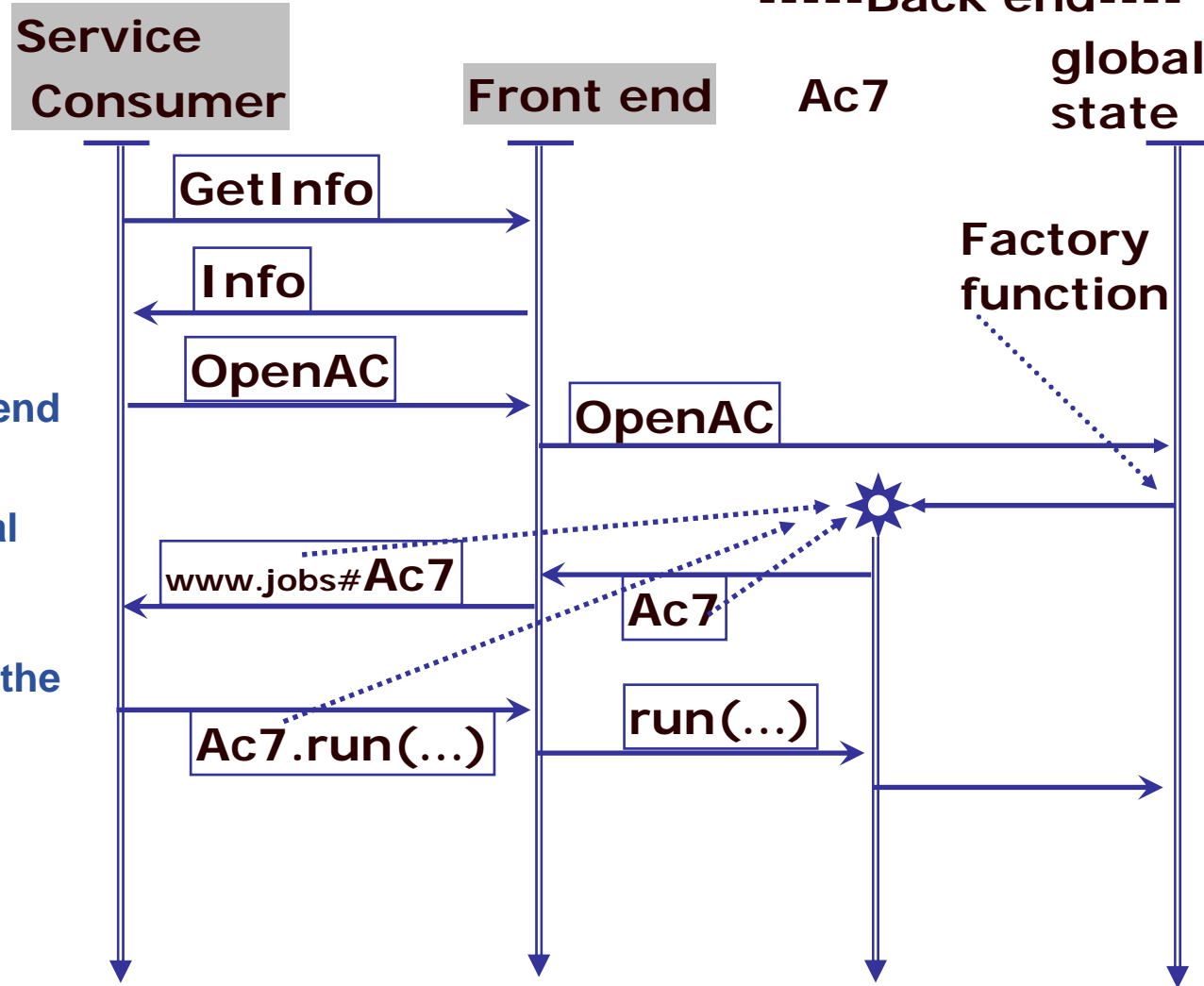
www.jobs#Ac7 is universal identifier (URI)

Can pass it to any service needing to operate on the resource

Job is also a resource

- www.jobs -Run Jobs Service-

-----Back end-----



- **Stateless** – implements message exchanges for which there is no access or use of information not contained in the input message. E.g. document compression / de-compression
- **Out-of-band persistent** state – response is affected by information that changes by some no-WS means. E.g. weather forecast service
- **Transient State (conversational)** – to co-ordinate a collection of related message exchanges E.g : shopping-basket;
  - Booking holiday - book hotel, flights and car-hire via different services with two-phase comit – confirm a reservation when all are held.
  - Proposed standards for this – WS-TransactionFramework
- **Persistent state (stateful resource)** – one message exchange produces a long-lived change in state which affects other message exchanges  
if shopping basket were carried forward from session, this would be persistent state
- **WSRF** is for Persistent State, not Conversational

- **A stateful (WS) resource**
  - **Is a repository for persistent state**
    - **Like an object in an object-oriented architecture**
  - **Has state that Comprises a set of state data (resource properties)**
    - **E.g Account for running jobs on a remote machine**
      - **Resource Budget**
      - **Lifetime of Account**
      - **...**
  - **Has a well-defined life-cycle – creation and destruction**
    - **Destruction can be**
      - **Explicit**
      - **Scheduled – “lifetime” : temporal garbage collection**
  - **Is associated with one or more web services, providing interface for manipulating it**
    - **A WS-resource comprises: its service; the resource itself**
  - **Can be known and acted upon by one or more Web Services**
    - **Via its URI**



- **Closely associated with WSRF is “Notification”**
- **Subscribe to a “Special Offers” Notification Service**
- **Notification Service sends a communication to me, I can reply with “buy it”.**
- **Relation to WSRF**
  - A subscription is a WS-resource
  - A resourced service can do notification –
    - to notify consumers of changes in state of a resource
      - Value change
      - Destruction
- **In grids – e.g. run a job, get notification of job termination**

- **WSRF builds on**
  - **WS-Addressing** – W3C Candidate Recommendation
  - **WS-Notification**
- **WSRF comprises**
  - **WS-ResourceLifetime** – OASIS working draft
  - **WS-ResourceProperties** – OASIS working draft
  - **WS-RenewableReferences** – who knows?
  - **WS-ServiceGroup** – OASIS working draft
  - **WS-BaseFaults** – OASIS working draft
- **WS-Notification comprises**
  - **WS-BaseNotification 1.0** – OASIS Public Review draft
  - **WS-BrockeredNotification 1.0** – OASIS Public Review draft
  - **WS-Topics 1.2** – OASIS working draft
- **Risky –**
  - **Not yet adopted, let alone WS-I**

## Goals

- An orientation to Web Services and to their role in Grid computing
- No prior knowledge assumed

## Content

- Web Services
- Web Services and Grids
- WSRF
- **GT4**





- **Context – Web / Grid Services**
- **GT4 Architecture** ←
- **Installation**

## A Grid services toolkit

- Architectural Concepts
- Development Framework (actual tools)
- Generic Components
- Specific Components

- **Architectural Concepts**

- WSRF/Notification –
  - WS-resources / resource properties
  - Lifetime / Scheduled Destruction
  - Subscription / notification
- GT4 specific
  - Resource factory service
  - Resource instance service
  - Singleton Resource
  - Resource Home

- **Development Framework (actual tools)**
  - Based on AXIS – web services development framework
  - Extended with
    - WSRF features
      - *Resource Properties attribute of PortType*
    - PortType Inheritance
      - *A new attribute of PortType*

```
<portType name="XPortType"
  wsdlpp:extends="wsrpw:GetResourceProperty"
  wsrp:ResourceProperties="tns:XResourceProperties">
  <operation name="Xop"> ... </operation>
```
    - Pre-processor – to flatten the WSDL
      - *Remove "Extends"*
      - *It is the flattened WSDL that is published*

- **Generic Components (libraries)**
  - WSDL – Common Definitions
    - Definitions for the standard operations of a WSRF service
    - ...
  
  - Java – code that can be inherited
    - “Class Xhome extends SingletonResourceHome”
    - “Class Yhome extends ResourceHomeImpl”
    - ...
  
  - Java – run-time support
    - Processing soap messages
    - ....
  
  - A GT4 container

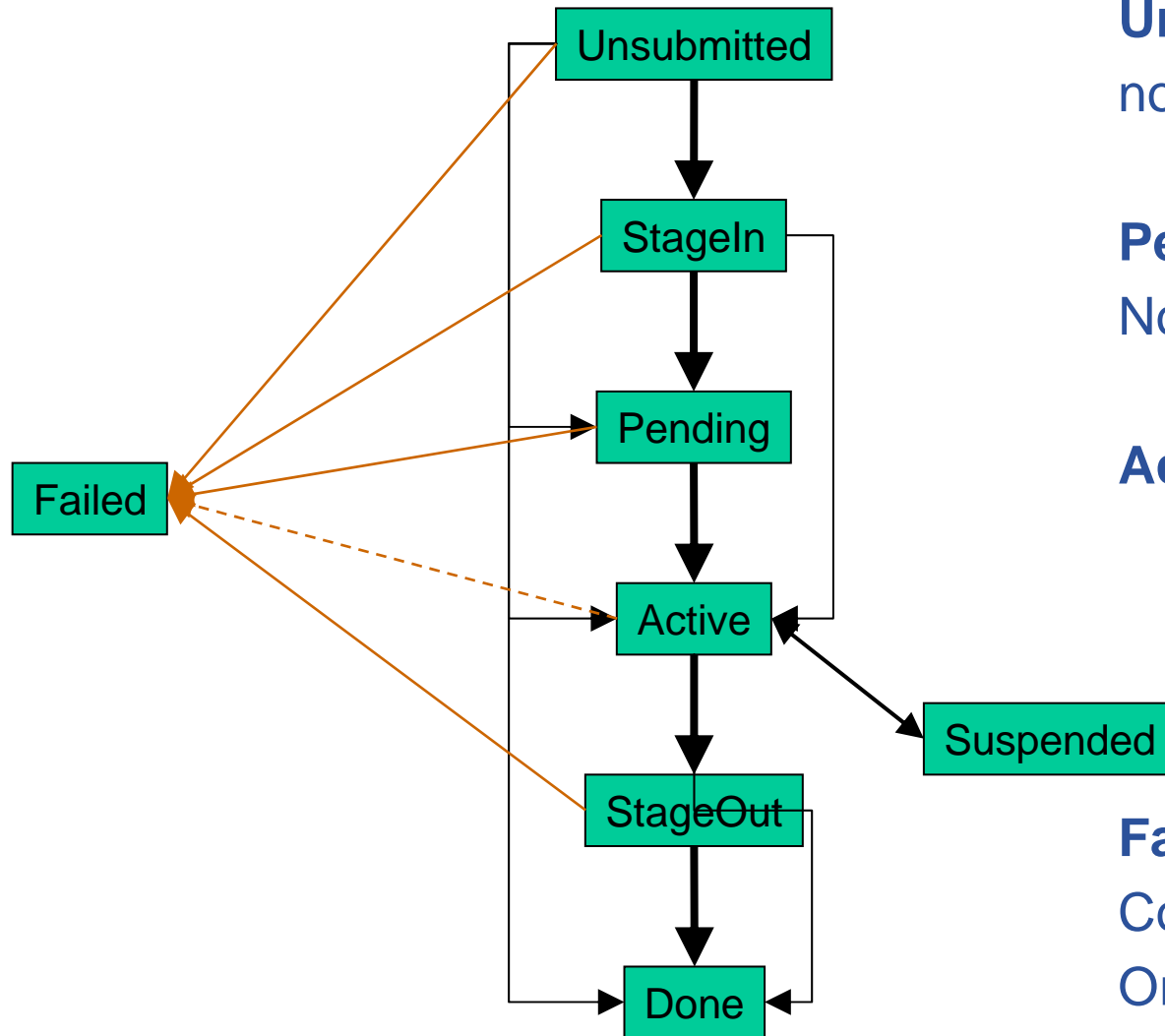




- **Issues Addressed**

- State
  - exactly-once execution semantics
- user-supplied executables
- input/output staging – need for user to manage staging of program/data files which may be large/shared/remote
- output streaming – access the output data as job is running
- control – e.g terminate job
- scheduling – job controlled by a scheduler
- monitoring – query / subscribe about job state

- **GRAM is meant to be used where it is important to be able to**
  - run arbitrary programs
  - achieve reliable operation
  - perform stateful monitoring
  - manage credentials
  - stage files
  - interact with schedulers
- **If you don't need the above, may be better to e.g.**
  - expose program as a web service



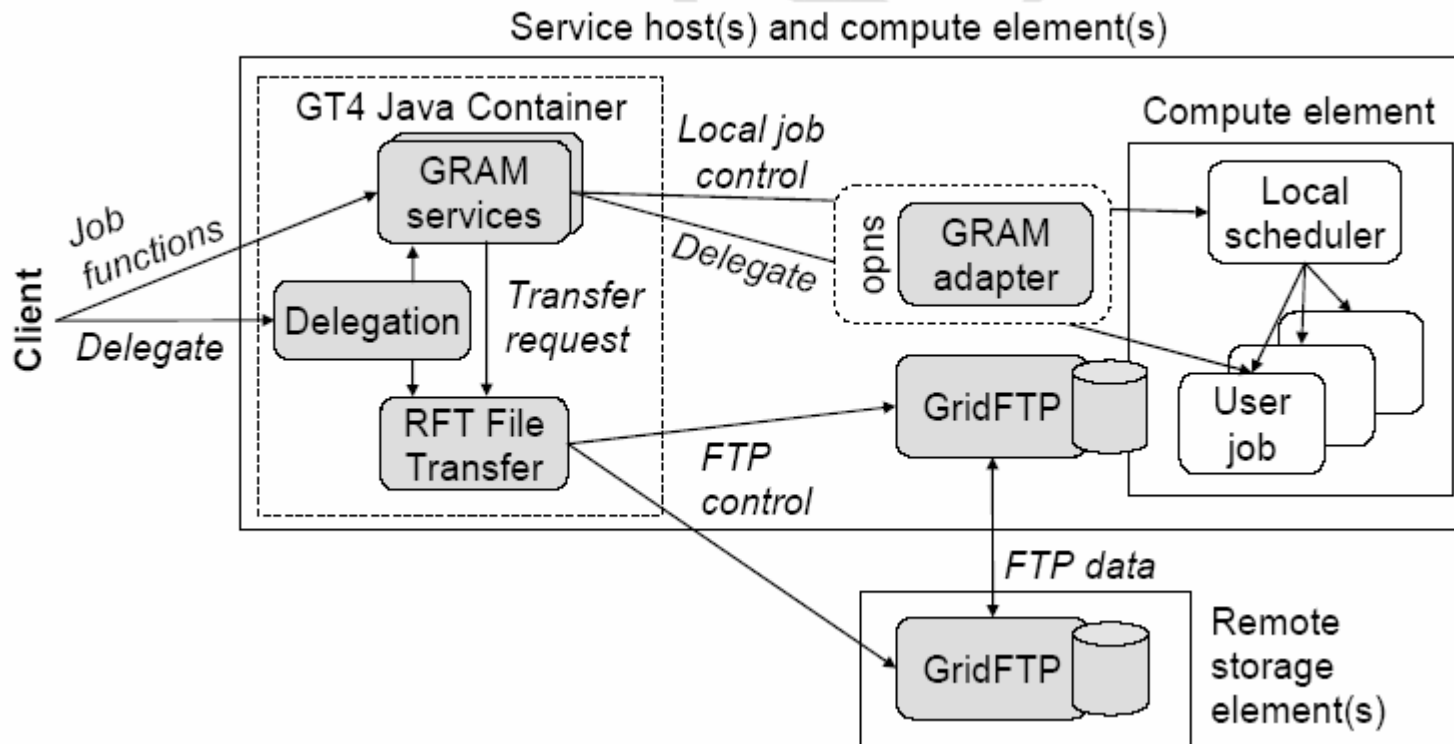
**Unsubmitted** –  
not yet submitted to scheduler

**Pending** – processor resources  
Not yet allocated

**Active** – Executing

**Failed** – terminated before  
Completion – error, user cancel  
Or system cancel

**!!! may be delay before  
change of state is seen !!!**



- **GT4 container – general purpose services**
- **Scheduler-specific GRAM adapter**
- **GridFTP server for data staging**

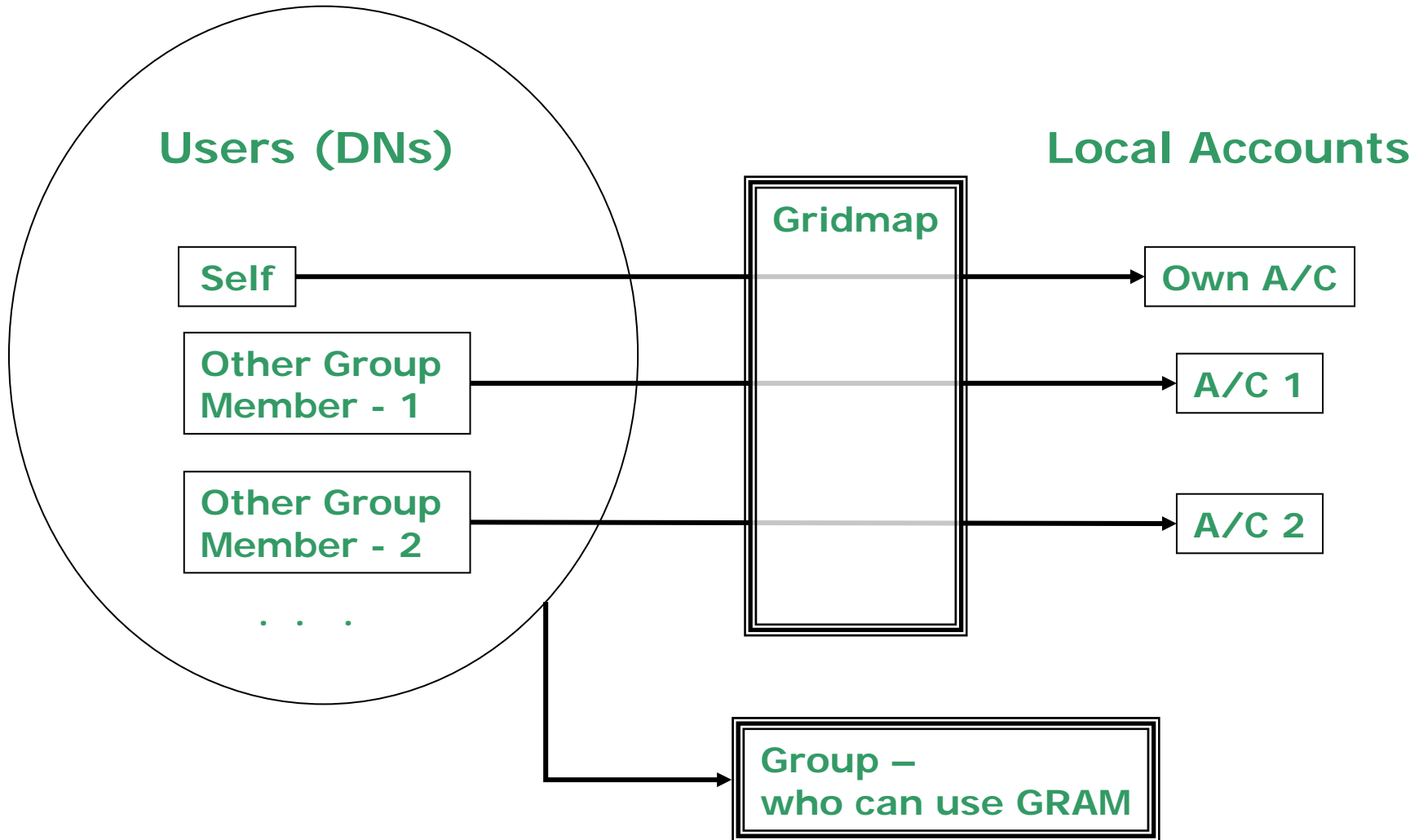
- **WS-RF resources (with WS-N) –**
  - managedJobs
    - **job details and its state**
  - delegated credentials
  - transfers in progress
- **Security**
  - built on certificates - GSI
  - Uses WS-Security (Web Services Security) mechanisms
    - **communications protocol providing a means for applying security to Web Services.**
  - done by an authorisation call-out; could be
    - **consult gridmap**
    - **contact an SAML server**
      - ***Security Assertion Markup Language (SAML) is an XML standard for exchanging authentication and authorization data between security domains***
  - results in local identity for executing the job

## GridFTP

- A secure, robust, fast, efficient, standards based, widely accepted data transfer protocol
- A Protocol
  - Multiple independent implementations can interoperate
- Globus also supply a reference implementation:
  - Server
  - Client tools (globus-url-copy)
  - Development Libraries

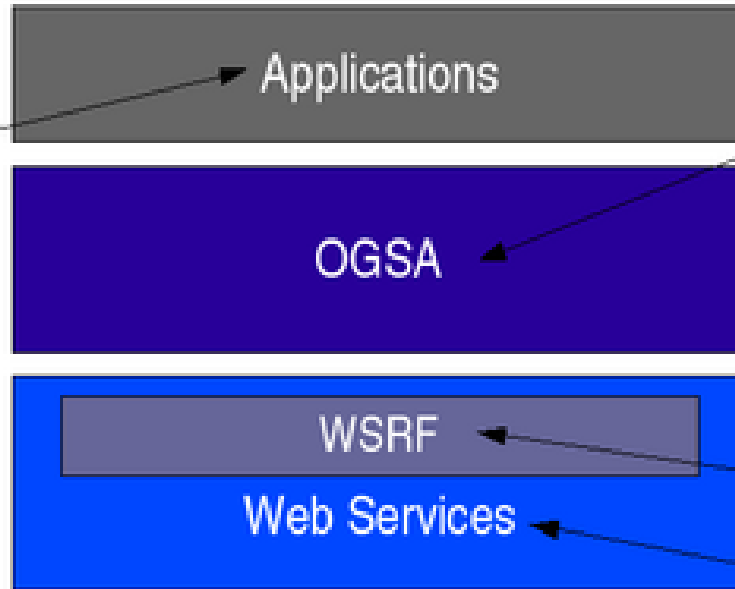
## RFT – Reliable Transfer Service

- Controls and monitors 3<sup>rd</sup> party multi-file transfer
  - exponential back-off
  - atomicity of multi-file transfer
  - parallel streams & TCP buffer size tuning
  - recursive directory transfer



- **Grid middleware in WSRF framework = Grid Service**
- **GT4 = A Grid services toolkit**
  - Architectural Concepts
    - WSRF/Notification
    - GT4 specific
      - Resource factory service
      - Resource instance service
  - Development Framework (actual tools)
    - Based on AXIS, Extended with
      - WSRF features
      - Interface Inheritance
  - Components

Grid applications are based on the high-level services defined by OGSA (i.e. not implemented from scratch using WSRF)



Standards in the works (GGF)

- VO management
- Security
- Resource management
- Job Management
- Data services
- etc.

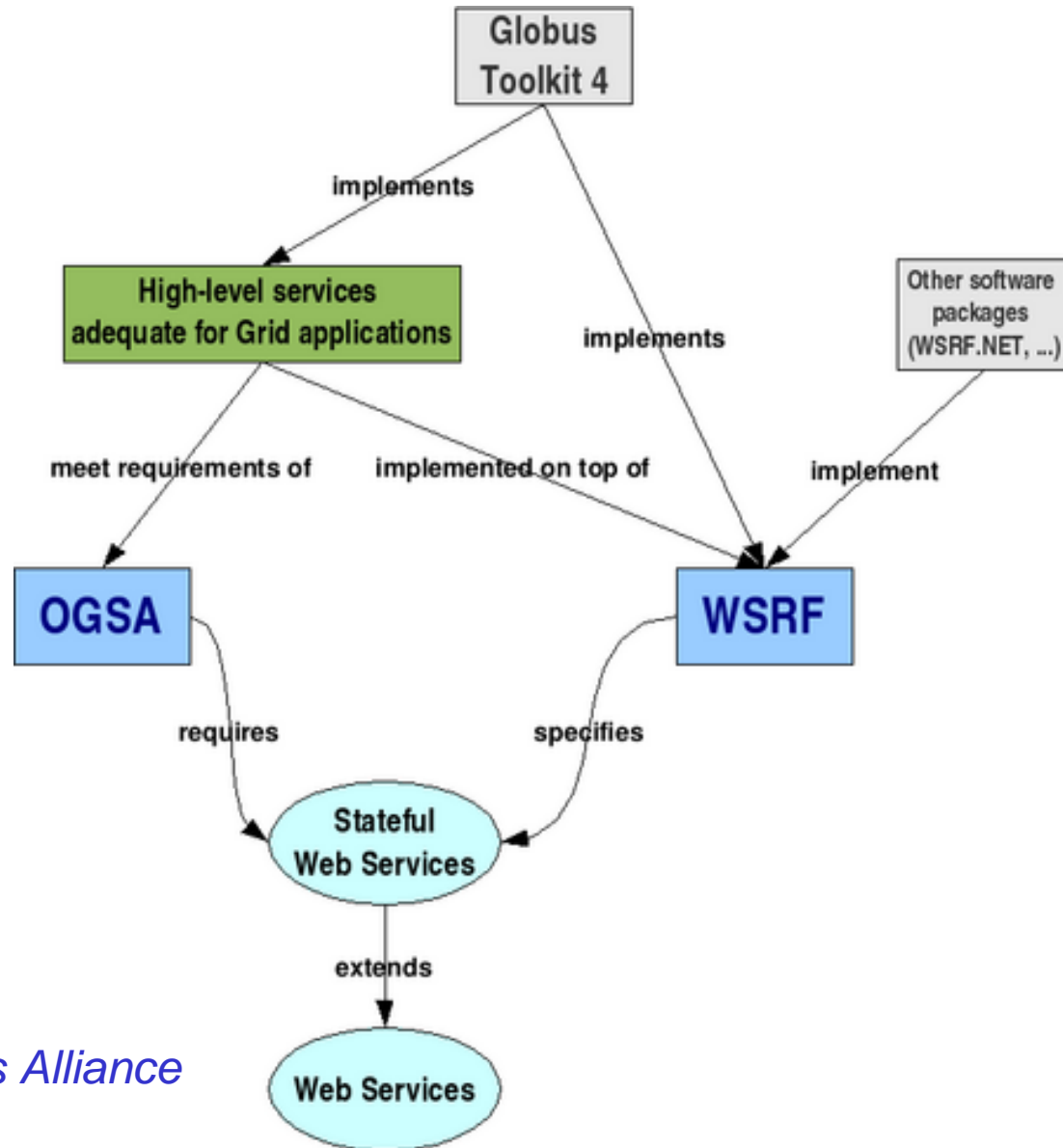
GT4 includes many of the services required by OGSA

Standardized (Oasis) and implemented (GT4)

Standardized (W3C) and implemented (e.g. Apache Axis)

*Diagram from Globus Alliance*

# GT4-view of OGSA and WSRF



*Diagram from Globus Alliance*

## E-Science

- Severe Storm Modelling
  - **Predicting storms based on real-time wide area weather instruments and large-scale simulation**
- National Fusion Collaboratory
  - **A VO for fusion research with S/W developed and executed on the application service provider model**
- Distributed Simulation
  - **Collaborative development and running of simulations across administrative boundaries**
- Reality Grid
  - **Parameter space exploration through computational steering and on-line high-end visualisation**
- VO Grid Portal
  - **Provides a coherent user view of the resources available to the VO**
- Learning Grid

## E-Business

- Commercial Data Centre
  - **Managing 1,000s of resources to reduce costs and increase utilisation – servers, storage, networks**
- On-line Media and entertainment
  - **Including interactive**
- Grid Resource re-seller
  - **Resource owners concentrate on core competencies; re-seller packages resources for sale to end users**

## General

- Service-Based Distributed Query processing
  - **Evaluation of queries over multiple services**
- Interactive Grids
  - **Fine execution granularity**
- Grid Lite
  - **Extending grid access to – PDAs, mobile phones, ....**
- Persistent Archive
  - **Mapping between old and new protocols/software/hardware**
- Grid Monitoring
  - **Across wide area networks with many dynamic and heterogeneous resources**

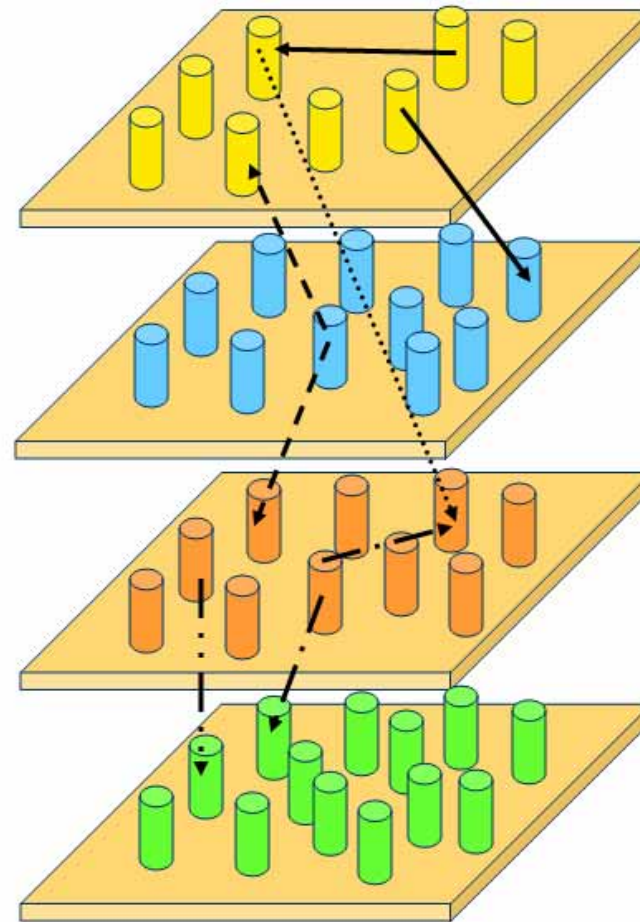
- From above use cases (and some more) identify needs for -
- Interoperability and support for Dynamic and heterogenous environments
  - **Resource virtualisation**
  - **Common management capabilities for resources**
  - **Resource Discovery and Query (on resource properties)**
  - **Standard protocols and Schemas**
- Cross-Organisation resource Sharing
  - **Global Name space**
  - **Metadata Services**
  - **Site Autonomy**
  - **Resource usage data – for accounting / billing**
- Optimisation –
  - **e.g advanced reservation; on-demand resourcing; utilisation tracking**

- Administrative Cost reduction
  - **Policy-based management – to automate grid resource control**
  - **Applications content management**
  - **Problem determination Mechanisms**
- Scalability
  - **Resource management needs to be heirarchical or peer-to-peer**
  - **Optimise for entire computational process as well as individual computation**
- Availability
  - **Automated Disastser Recovery**
  - **Automated fault management to avoid losing long-running multi-resource jobs**
- Ease of Use and Extensibility

- Execution
  - **Multiple job types – simple jobs .. Workflow**
  - **Job Management**
  - **Job Scheduling – across admin domains**
  - **Resource Provisioning – dynamically & automatically deploy and re-configure compute, data and network resources**
- Data
  - **Data access**
  - **Data consistency**
  - **Data persisteny**
  - **Data integration**
  - **Data location management**
- Security
  - **Authentication & Authorisation**
  - **Multiple security infrastructures**
  - **Perimeter security solutions**
  - **Isolation**
  - **Delegation**
  - **Policy Exchange**
  - **Intrusion detection, protection and secure logging**

- Built on Web Services Standards
  - **With**
    - Semantics
    - Additions
    - Extensions
    - Modifications (?)
  - **Which are relevant to grids**
- Composition / building block paradigm
- Deals with the service level
  - **Interfaces, semantics and interactions**
  - **Not the Services' internal architecture**
- Not
  - **Strongly layered**
    - Strongly layered means Level N can only interact with level N-1
  - **Strictly Object-Oriented**
    - The architecture concept is service, not object

- Source *uses* target capabilities to provide a service to a client  $\longrightarrow$
- Service *composes* the capabilities of the underlying services to provide a higher level capability  $-\ - \ - \ - \longrightarrow$
- Service *delegates* requests to a related service for fulfillment  $\cdots \cdots \cdots \longrightarrow$
- Service *refers* to the related service for substantiating a request (validation, concretization)  $- \cdot \cdot \longrightarrow$
- Service *extends* (subsumes and augments) the capabilities of the related service  $- \cdot \cdot \cdot \longrightarrow$





- **Aim is to establish standard functionality domains and associated standard specifications**
  
- **Infrastructure Services – Base Profile**
  - **Core Web Services**
    - XML
    - SOAP
    - WS-I
    - WSDL for definition of grid services
  
  - **Additionally**
    - Security
    - WSRF
    - WS-Notification

- **Execution Management Services (EMS)**
  - Finding execution locations
  - Selecting execution locations
  - Preparing for execution – deployment, configuration, data staging
  - Initiating execution
  - Managing execution – check-pointing, fault detection, re-start
  
- **Resource Services**
  - wrap the resources themselves
  
- **Job management and Monitoring Services**
  
- **Resource Selection Services**

- **Data Services**
  - **Move data where it is needed**
  - **Manage replicated copies**
  - **Run queries and updates**
  - **Transfer between data formats**
  - **Manage metadata**
  
- **Information (and Monitoring) Services**
  - **Support the ability to efficiently access and manipulate information about**
    - Applications
    - Resources
    - Services
  - **Functionalities**
    - Naming scheme
    - Discovery
    - Message Delivery
    - Logging
    - Monitoring

- **Security Services**
  - To facilitate the enforcement of the security-related policy within a (Virtual) Organisation – generally policy-driven

## Functionalities / example types of Security Services

- Authentication
- Identity mapping
- Authorisation
- Credential Conversion
- Audit and Secure Logging
- Privacy

- **Resource Management Services**
  - The resources themselves - e.g. re-boot a remote host
  - Grid resource management – e.g. reservation, monitoring
  - The OGSA infrastructure – e.g. monitoring a registry service
  
- **Self-Management Services**
  - at very early stage - a collection of requirements rather than specific service type being identified
  - **System components**
    - Computers
    - Storage devices
    - Networks
    - Operating Systems
    - Applications
  - **Need to be**
    - Self-configuring
    - Self-healing
    - Self-optimising



- OGSA – Framework for Grid Services
  - **Architecture –**
    - Service Oriented (change from “too Object Oriented”)
    - types of services
    - requirements
    - Functionalities
  - **Specific Interface Standards, e.g. JSDL**
  - **Base Profile**
    - Core Web services
    - Additions – WSRF
- Web Services
  - **Factored Standards – XML, XSD, SOAP, WSDL**
- Comparison between grid and WS
- Comparison between Objects and Services