



open middleware  
infrastructure  
institute uk

# Using the OMII-UK distribution

Steven Newhouse  
Director



# OMII-UK Training Session

- Background
  - What is OMII-UK?
  - Architecture & Distribution
- Installing the OMII Client
  - Job Submission & Job Monitoring
- Installing the OMII Server - Overview



open middleware  
infrastructure  
institute uk

# OMII-UK: What is it?

---



# OMII-UK

- A partnership between projects:
  - myGrid at Manchester (Carole Goble - Chair)
  - OGSA-DAI at Edinburgh (Malcolm Atkinson)
  - OMII at Southampton (Dave De Roure)
- Started January 2006 All funded for 3 years
  - Manchester – Expanded Engineering activity
  - Southampton – Expanded Community activity
  - Edinburgh – Continuation of OGSA-DAI team



# Objectives of OMII-UK

- To distribute a sustained, well-engineered, interoperable, documented and supported set of easily-used integrated middleware services, components and tools
- To engage proactively with user communities in defining and developing this distribution
- To maintain a leading international role in advanced e-Infrastructure middleware provision



# OMII-UK Activities

- User engagement
  - Forming partnerships with targeted user communities
- Sourcing
  - Working with UK and international service developers and middleware providers
- Software Engineering
  - Quality-assured software engineering, coordinated across OMII-UK partners and the managed programme
- Grid engagement
  - Tracking and engagement with the standards processes
- Sustainable business
  - Attracting partnerships and new investors

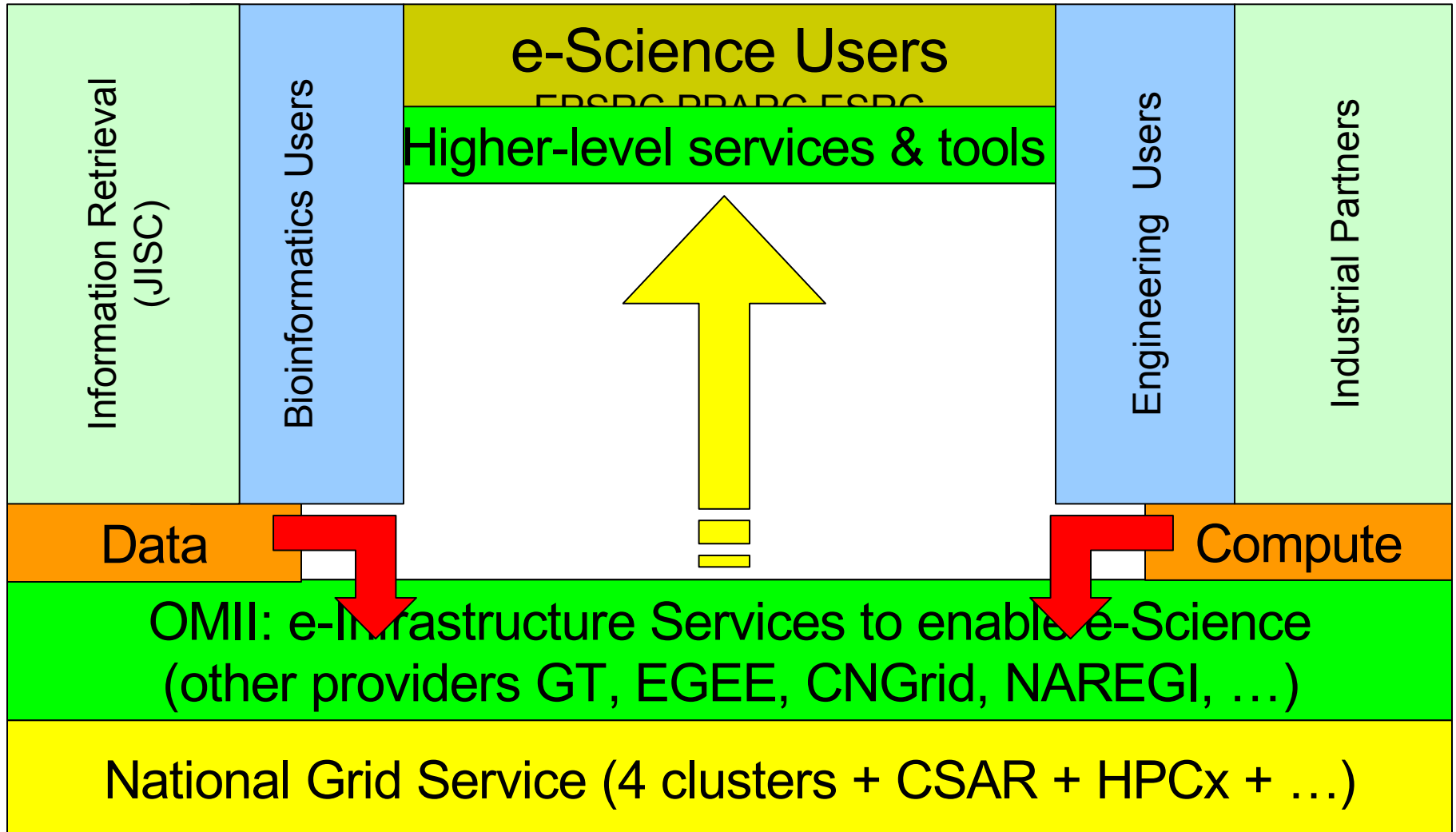


# OMII-UK in context

- OMII-UK is ***not*** a multisite research project
- OMII-UK is delivering services and products
  - Support & Training
  - Engineered Software: Client & Server distribution
    - OGSA-DAI, Taverna
    - Easy to install & use
- OMII-UK is a large activity:
  - Core teams – 40 employees across 3 sites
  - Including Contractors – 50 staff across 10 sites



# OMII-UK and the community





# For Service Providers

- **Goal:** I want others to access my resources & applications through composable services
- I want to provide secure controlled access to:
  - My applications:
    - Specify who can access which applications
  - My computational resources:
    - I can limit external usage of my resources
- Provides an interface that allows remote users to access my resources
- Enable collaboration with other partners



# End Users to access services

- **Goal:** I want to use other resources & applications
- Through a network of service providers I can...:
  - Gain access to applications that I do not have installed locally
  - Use remote machines with more CPU, memory or storage
    - Process larger problems sizes
  - Transparently switch between different service providers
    - No exposure to underlying OS, queuing policy, disk layout etc.

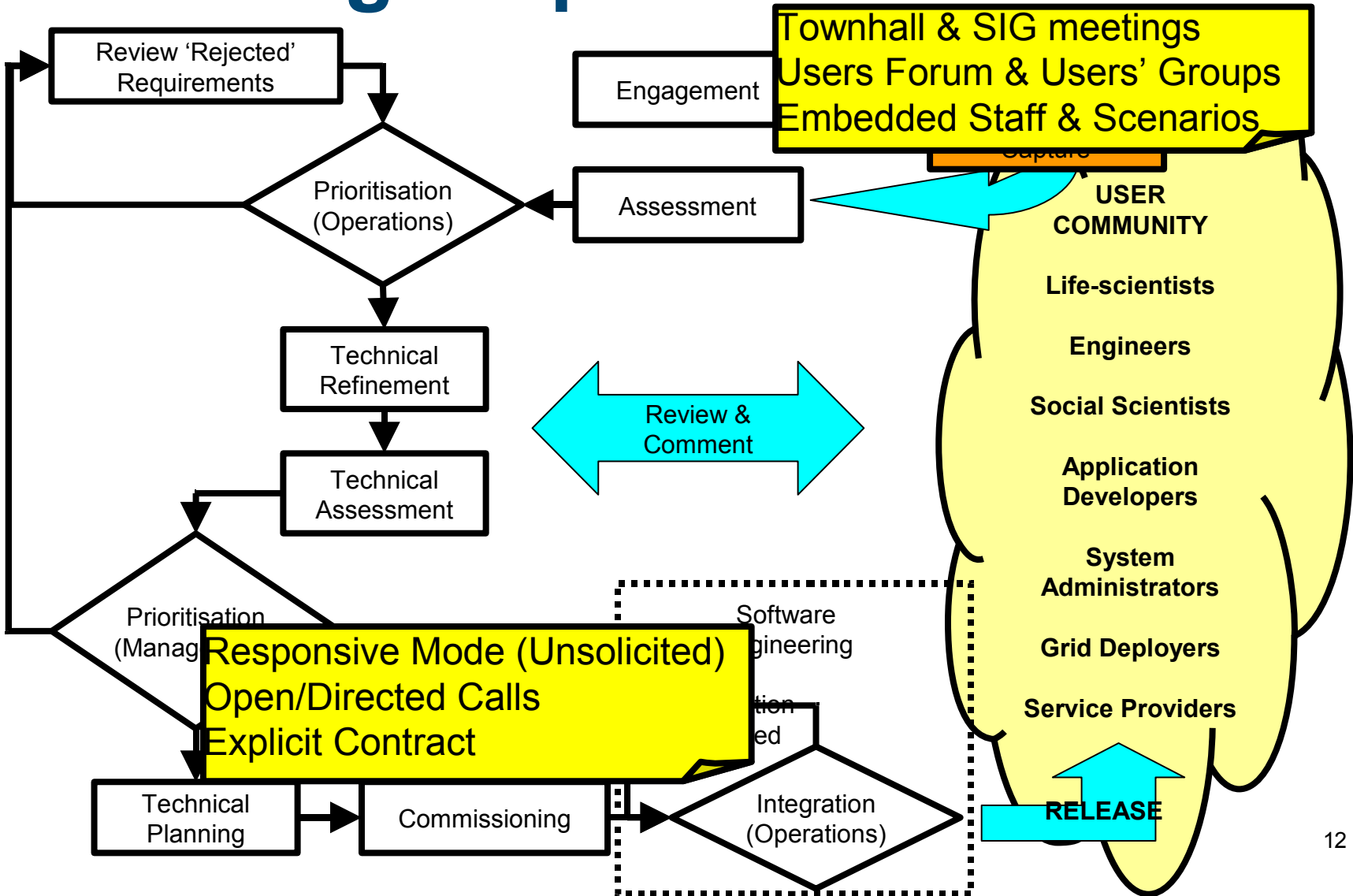


# Developers to build services

- **Goal:** I want to build new services by reusing existing services and standards
- Working within standards bodies to define application interface standards
- Providing open source implementations of infrastructure specifications
- Building on top of established broadly accepted specifications



# Gathering Requirements





open middleware  
infrastructure  
institute uk

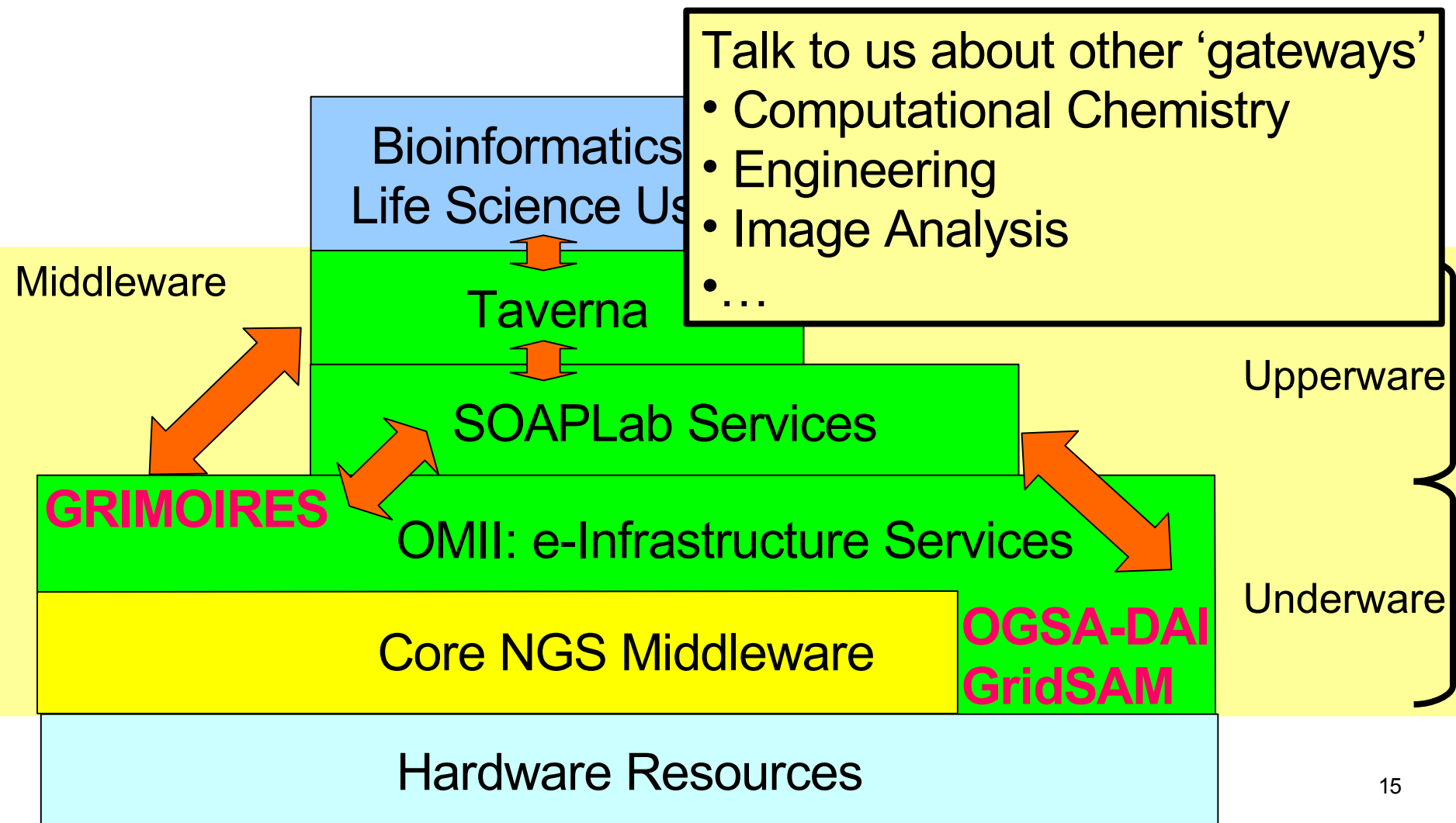
# Distribution & Architecture



# Grid Architecture Today

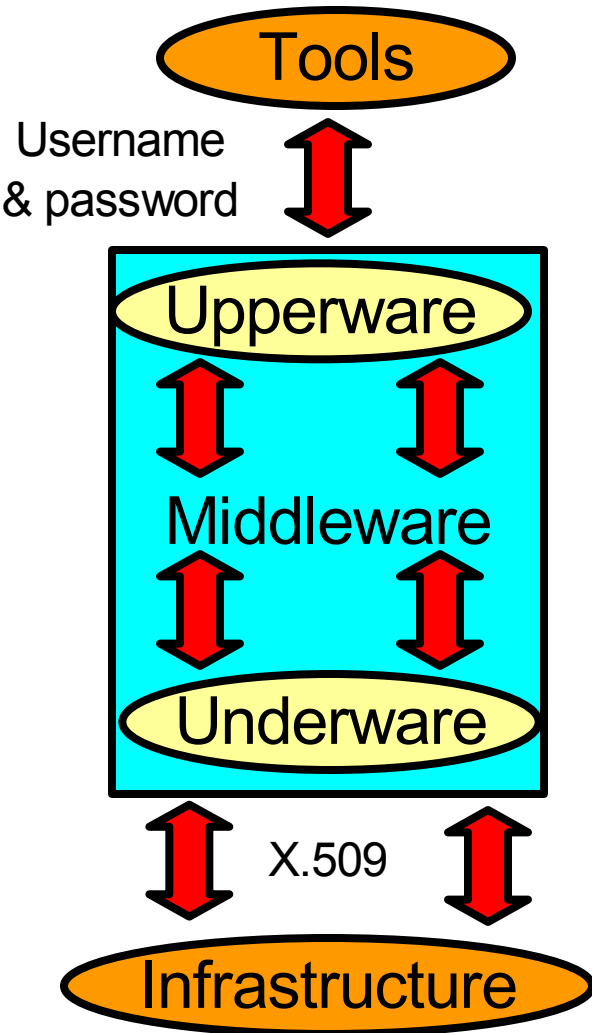
- The best way of designing Grids...
  - Loosely coupled services
  - Message based exchange
- The best way of running Grids...
  - Interoperability between versions & grids
  - Standards for infrastructure & services
- The best way of building Grids...
  - Leverage existing infrastructure & standards
  - Use Web Services...

# OMII-UK & National Grid Service: Life Sciences Gateway





# Managing Service Complexity



- User Perspective:
  - One service invocation does all
  - Customised to **their** problem
- Generic Middleware Provider:
  - Re-factor to exploit:
    - Lower-level services
    - Generic operations on the services
  - Need to work with deployed infrastructure
  - Build on standards
  - Promote reusability
- In meeting requirements
  - A healthy tension, e.g. a registry
  - User Capability → Software Engineering

# Where does our software come from?



- Open Source Community
  - Tomcat, Axis, etc.,
- Software Repository
  - Accept software contributions
  - Software deployed, tested & graded to provide feedback
- Managed Programme
  - Fill gaps to build a solid enabling infrastructure
  - Projects to bring research software to production quality



# Composable Services

- Job Submission & Monitoring service (GridSAM)
- Workflow services (Taverna, BPEL)
- Data Access & Integration service (OGSA-DAI)
- UDDI Registry service (Grimoires)
- Reliable messaging: WS-R & WS-RM (FIRMS)
- Notification using WS-Eventing (FINS)
- Scripting Environments:
  - Matlab & Jyton (GeodiseLab)
  - PERL (WSRF::Lite)



# Integrated Services

- **Provides:** Dynamic flexible authorisation model across services to support application execution.
- Authentication: CA issued X.509 certificates
- Authorisation: Interaction dependent authorisation process
  - Access control lists tied to process context and state
    - i.e. impose server side workflow requirements
  - Supports “delegation” and “subordination” actions
- Accounting: Activity matched against allocated quota
  - Clients control who can access “their” allocated quota
  - Collaboration with minimal overhead for service providers



# Standards

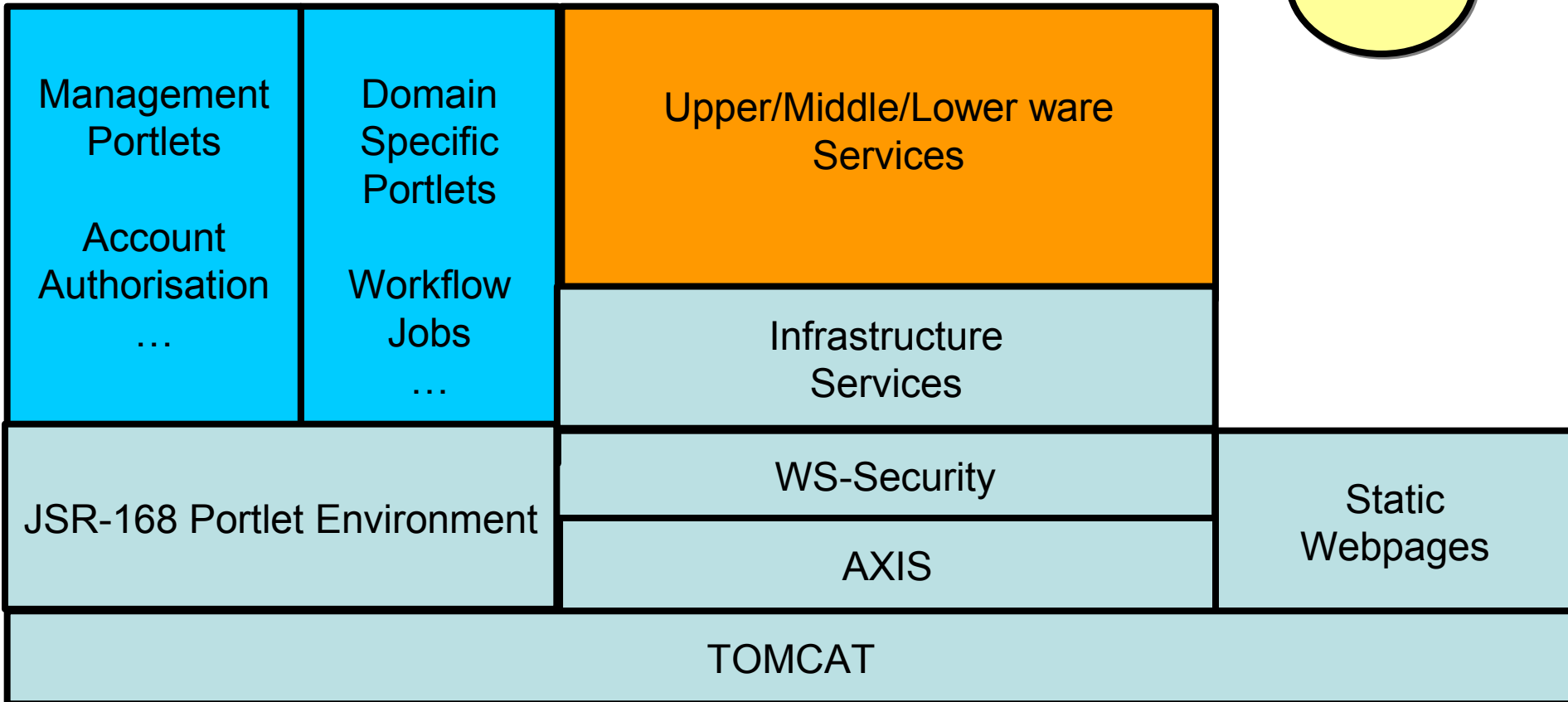
- Upperware Specifications
  - Community/Domain Driven: IVOA, Bioinformatics

Build on standards so that you get your core infrastructure for free!



# Architecture

Clients





# Delivering The Software

- Commissioning the required software
- (Re-) packaging into a distribution
- Verifying portability through deployment
  - NMI Build and Test Framework
  - ETICS - e-Infrastructure for Testing, Integration and Configuration of Software
  - OMII-Europe
- Repeatable testing
  - Measure robustness – bugs discovered per cycle



# Extensible Distribution

- Client & Server Distribution
  - Installation Script
  - Core Software
  - Optional bundles
    - Dependencies
- Domain/project specific distributions
  - Expand tarball, add in your bundles, regenerate tarball
  - Redistribute, download & install with your services
- Ability to automate installation
  - Most keyboard inputs can be specified as environment variables
    - Use wrapper script to drive mass deployment

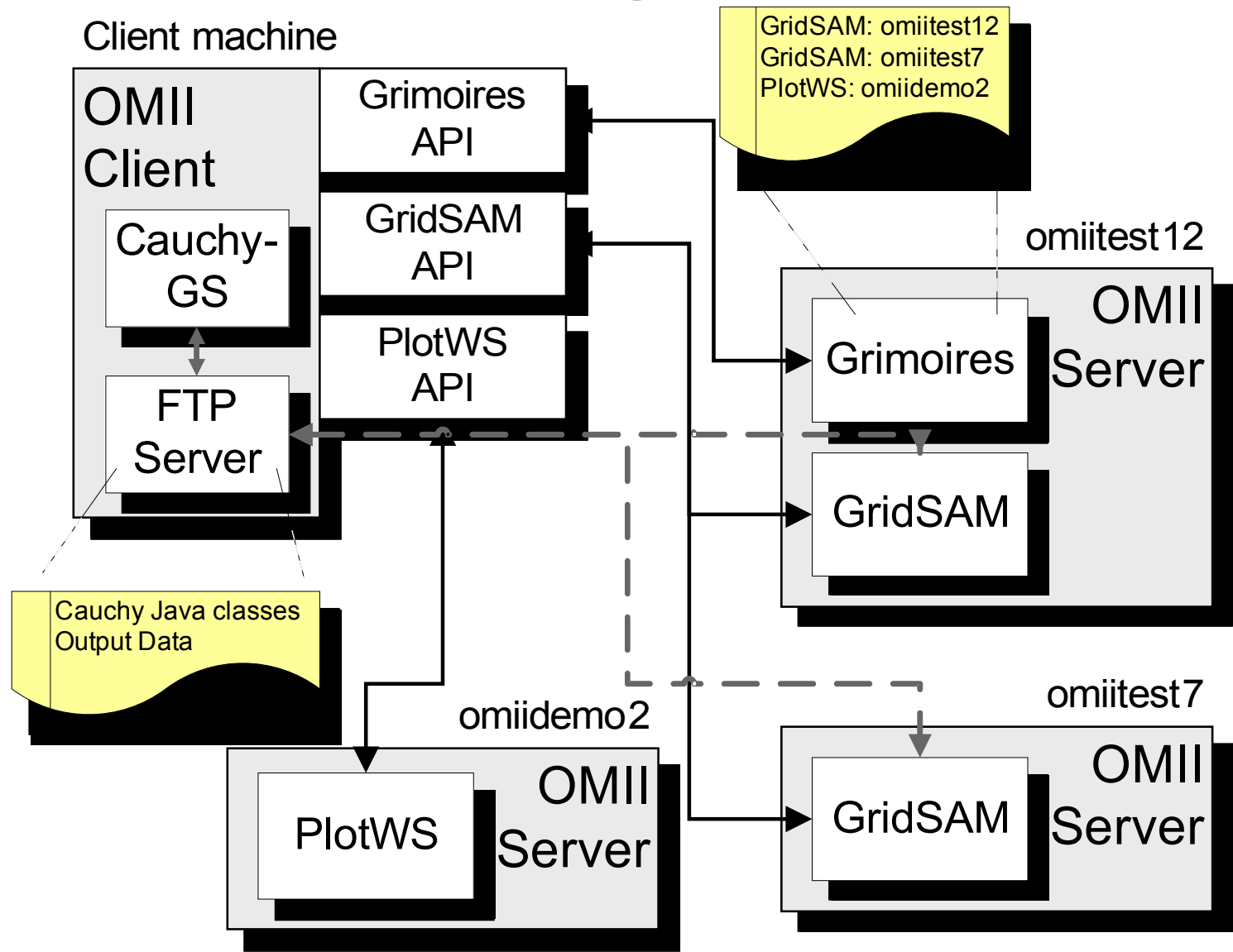


# Current OMII release - 2.3.3

- Open source infrastructure:
  - Tomcat using http and/or https
  - Axis 1.2RC3 + large attachment patch
  - WSS4J (WS-Security: sign message & verify signature)
- Updates to the Integrated Services
- Composable Services from the Managed Programme:
  - GridSAM for job submission and job monitoring
  - GRIMOIRES UDDI registry
  - Jython client side environment from GeodiseLab
  - WS-Eventing from FINS
  - OGSA-DAI WS-I for data access



# Classic SOA using OMII 2.3





# OMII 3.0.0 (June)

- Updates to the current services:
  - GRIMOIRES
  - GridSAM
  - FINS – Notification
  - GeodiseLab (Jython environment)
- Integration of new services (provisional):
  - OGSA-DAI
  - FIRMS – Reliable Messaging
  - BPEL Workflow Editor
    - ActiveBPEL execution engine
    - MANGO: Example application
  - Previews: Java bindings to SAGA, OGSA-BES



open middleware  
infrastructure  
institute uk

# Installing the OMII Client



# What we will cover...

- Where to get the client
- How to install the client
- Verifying the installation
- Resolving some common problems



# Where to get the client

- Register at [www.omii.ac.uk](http://www.omii.ac.uk) & login
- Go to the downloads page
- Download the **client** distribution ([version 2.3.3](#))
  - Tested under most 32 bit Linux distributions, Windows XP
- Distribution requires Java
  - Lightweight testing with most SUN & IBM JDKs/JREs
    - OK with most 1.4.2 & 1.5.0 versions
  - Do NOT use gcj (RH distributions)



# Installation Requirements

- OMII Client distribution
- Java already on your path
  - Check: which java
- JAVA\_HOME already set
  - Check: echo \$JAVA\_HOME



# The Client Distribution

- Install as 'normal' user
- In your home directory and expand:
  - `tar -zxf omii-client-2.3.3.tar.gz`
- Within the expanded directory, run the install script:
  - `cd omii-client-2.3.3/client`
  - `./OMIIClientInstall.sh`



# Installation Overview

- Define the client environment:
  - Location
  - Proxy (http & https)
- Get a certificate for the client:
  - Provide hostname & other details
- Complete Installation
- Verify the installation



# Inputs into the script

- Installation location:
  - Default: `$HOME`
  - Accept default & installation will take place in:
    - `$HOME/OMIICLIENT`
- Do you use a proxy to access the internet?
  - For http? NO
  - For https? NO
- Request a certificate...



# Certificate Generation – Input

- Machine name: **sjn-desktop.omii.ac.uk**
- Organisation: **University of Southampton**
- Organisational Unit: **OMII**
- Location: **Southampton**
- State: **England**
- Country: **UK**
- Email address: **s.newhouse@omii.ac.uk**



# Some notes on the certificate

- A certificate is needed on the client (WSS4J)
- Details passed to the OMII CA
- Provides a certificate of low value
  - No authentication checks
  - Expires after a month
- Values are not verified
  - Any non-null input will be OK

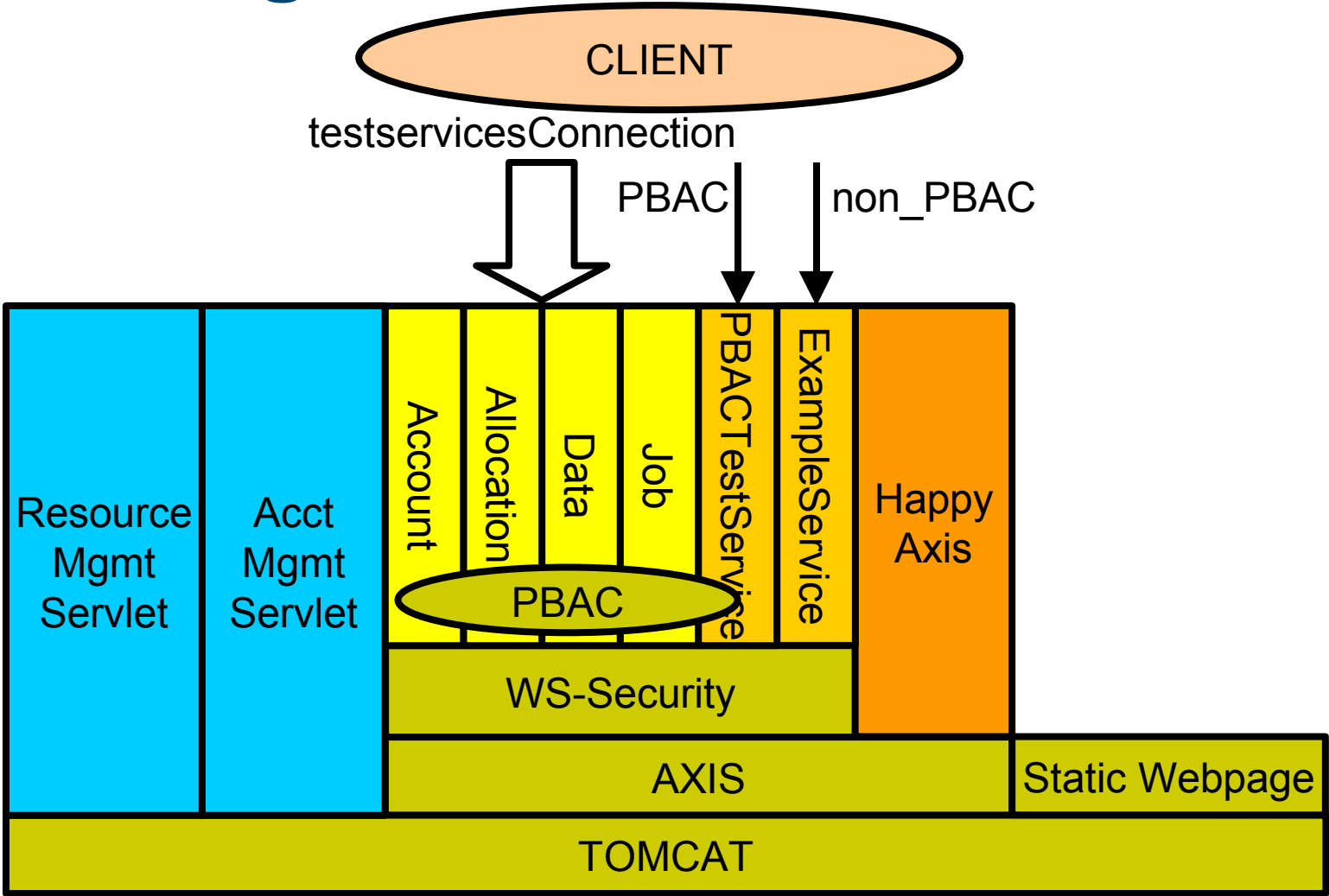


# Installation Complete!

- Files are copied & configured
- Next stage testing...



# Checking the client install





# Testing the installation

- Three tests at the end of the installation
- All will fail if the container is not up & running  
OR
- If you have no connection to the test servers



# Test server

- All installation tests can be run against:  
demo-2-3-3.omii.ac.uk



# Testing the basics

- Is the remote container up?
  - Browser to: `http://<remote host>:18080/`
  - Should see 'Welcome to Tomcat' page
- If not contact the service provider
  - For `*.omii.ac.uk` contact `support@omii.ac.uk`

# Testing the environment (non-PBAC tests)



- Enter service provider hostname and port
- Provide a test message that will be echoed back to you
- Access a secured service
  - Will fail if the client certificate installation is not correct

# Testing the Integrated Services (PBAC tests)



- Enter service provider hostname and port
- Provide a test message that will be echoed back to you
- Creates a 'conversation'
- Uses this conversation to access a PBAC service
  - Second phase will fail if not able to create a conversation



# Testing connection to the Integrated Services

- Connection Test (`testservicesConnection`):
  - Verifies that the 4 integrated services are accessible
  - Will fail if there is a server side problem
  - Will fail if the services are not accessible
    - Firewall, network failure etc.



# Examining the Installation

- Go to the installation directory:
  - `cd ~/OMIICLIENT` [default location]
- Can rerun the previous tests from here:
  - `invoke_nonPBACTestService`
  - `invoke_PBACTestService`
  - `test-services/testservicesConnection`



# Some common problems:

- The wrong JDK (but we support a lot):
  - Make sure both the PATH & JAVA\_HOME are set
  - Must point to the same JDK or JRE



# Summary

- Basic installation very simple
- Low-value certificate generated on demand
  - Will expire after 30 days (just to get you started!)
  - Advised to obtain a UK e-Science CA certificate
    - For more details: <http://ca.grid-support.ac.uk>
- Tests to verify client installation
  - To connect & interact with reference services



# Managed Programme Client

- `cd ~/omii-client-2.3.3/managed_programme`
- `./OMIManagedProgrammeClientInstall.sh`
- Specify the installation directory

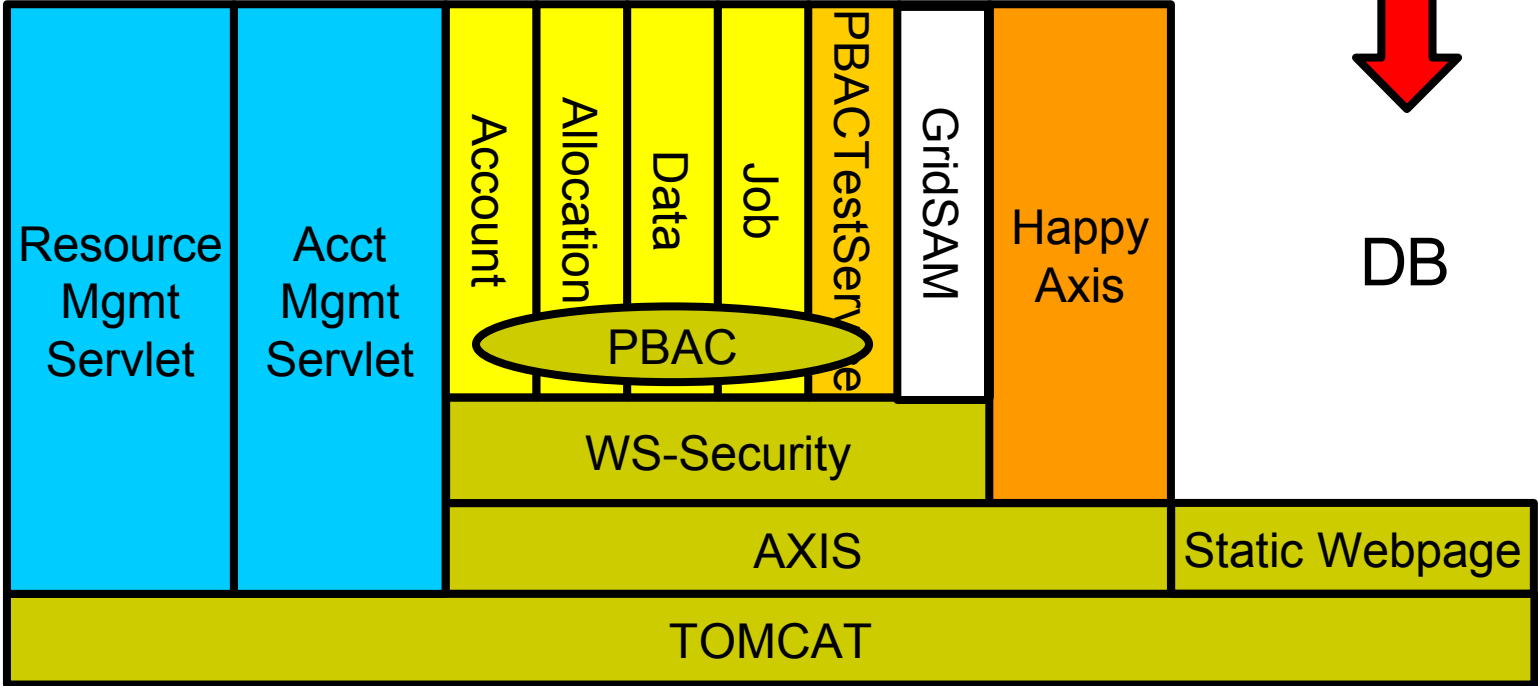


open middleware  
infrastructure  
institute uk

# Installing the OMII Server



# OMII Server Infrastructure





# Multi-layered Server Distribution

- Database
  - PostgreSQL 8 binary
- Base & Extensions:
  - WS Infrastructure: Tomcat & Axis,
  - WS Handlers: WSS4J
- Integrated Services:
  - Infrastructure: PBAC
  - Functional: Job & Data
  - Management: Account & Resource Allocation
  - Example Application: Cauchy
- Composable Services (Managed Programme)



# Current Solaris Support

- No support for x86 or SPARC
  - But generally Java based distribution
- This WILL change...
  - Working on a port using WUN server
- Two issues:
  - Ship PostgreSQL database binary
    - Deploy server on Solaris & database on Linux
  - Installation scripts built around Linux commands
    - GNU before Solaris commands may help a lot!



# Installing the OMII Stack

- Pre-requisites:
  - Most 32 bit Linux distributions, Max OSX (DB on Linux)
  - Either ROOT or NON-ROOT install
  - Java (JDK) Installation
- Expand the archive
  - `tar -zxf omii-server-2.0.0.tgz`
  - `cd omii-server-2.0.0`
- Start the installation script: `./OMIIstackInstall.pl`



# OMII Stack Installer

- WS Container
  - Database
  - Components
- Managed Programme (Composable) Services
  - Database
  - Components
- Integrated Services
  - Database
  - Components



# Database Install (Option 1)

- Process:
  - Enter the port
- Environment:
  - Do not try and install into an existing database
    - As we don't know where it might be, file permissions, etc.
  - Standalone installation means it has a single purpose.

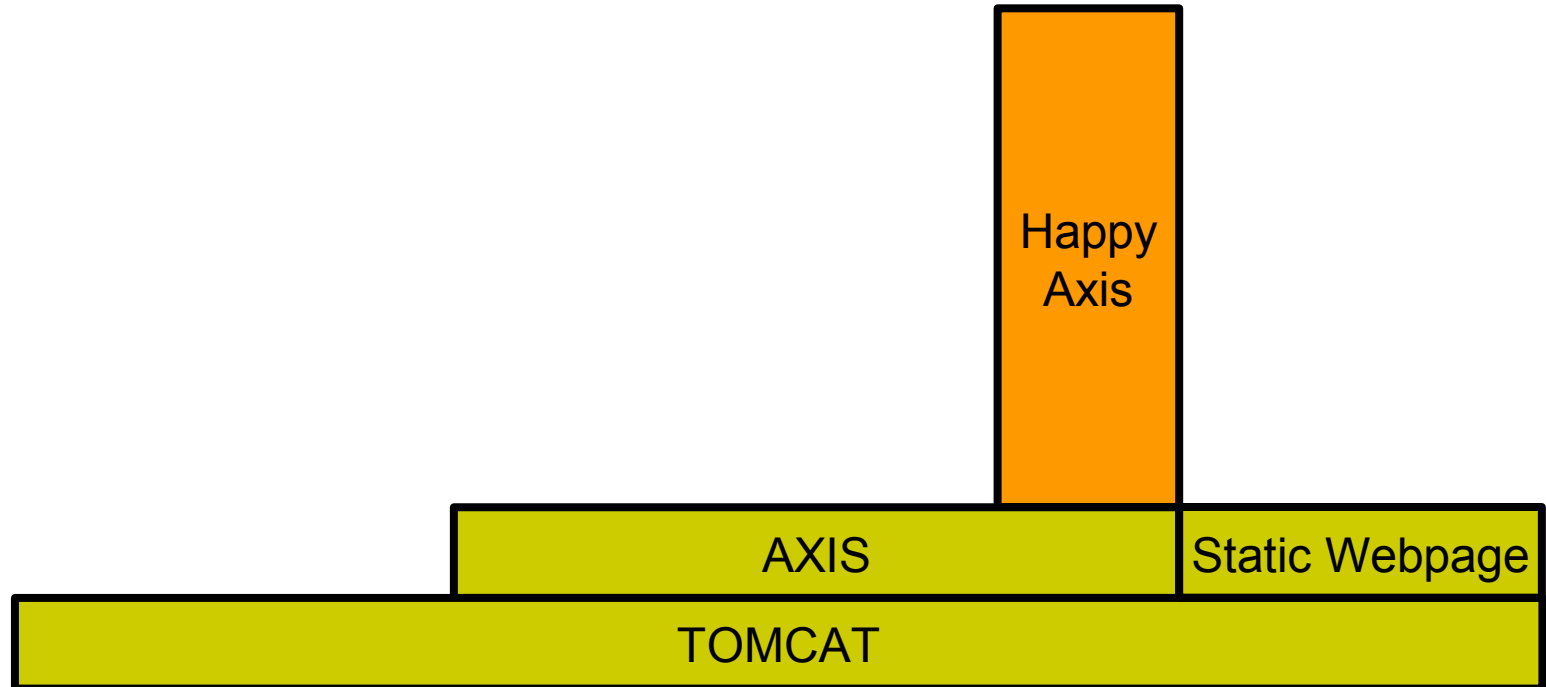


# Stack Install (Option 2)

- Process:
  - Specify your hostname
  - Specify the database host
  - Specify the database port
  - Specify the installation directory
- Activity:
  - Stage 1: Install Tomcat & Axis
    - Verifies JDK, disk space & free port (18080)
    - Expands & installs: Ant, Tomcat & Axis



# OMII Infrastructure – Base





# Tomcat & Axis Testing

- Starts the Tomcat container
- Verifies that:
  - The Tomcat home page exists
  - The Axis environment exists
  - The HappyAxis install is OK
- Stops the Tomcat container

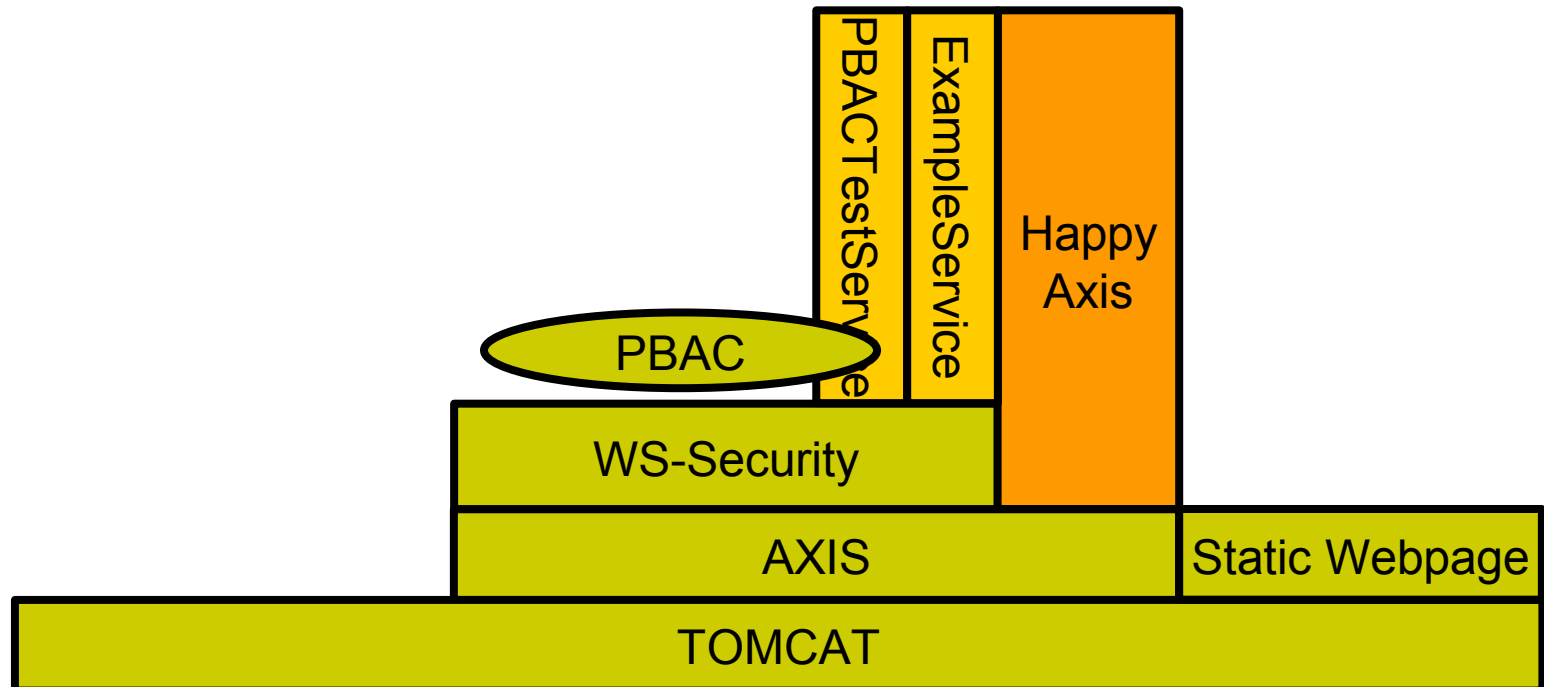


# If installing as root

- Creates:
  - Group: `omi_tomcat`
  - Users: `omi_tomcat_owner` & `omi_tomcat_user`
- Installation directory:
  - Owned by: `omi_tomcat_owner:omi_tomcat`
  - Runs as `omi_tomcat_user`
- Additional users can start/stop tomcat
  - Add into `omi_tomcat` group



# OMII Infrastructure





# Installing WSS4J

- Request a certificate for this machine
  - Similar information as the client install
  - Download archive and expand locally
- Rest of the install script requires <CR>
  - For default options!



# Integrating WSS4J with OMII

- WSS4J needed by:
  - Any service in the container (optional)
  - Any PBAC service
- Installation Script:
  - Capture the OMII security configuration
    - Location of the keystore & its password
      - If you change the default password you MUST alter the keystore password
    - Names of the certificates with the keystore
  - The defaults are acceptable for the:
    - Keystore obtained during the installation process
    - Default installation location
  - Deploy the test service

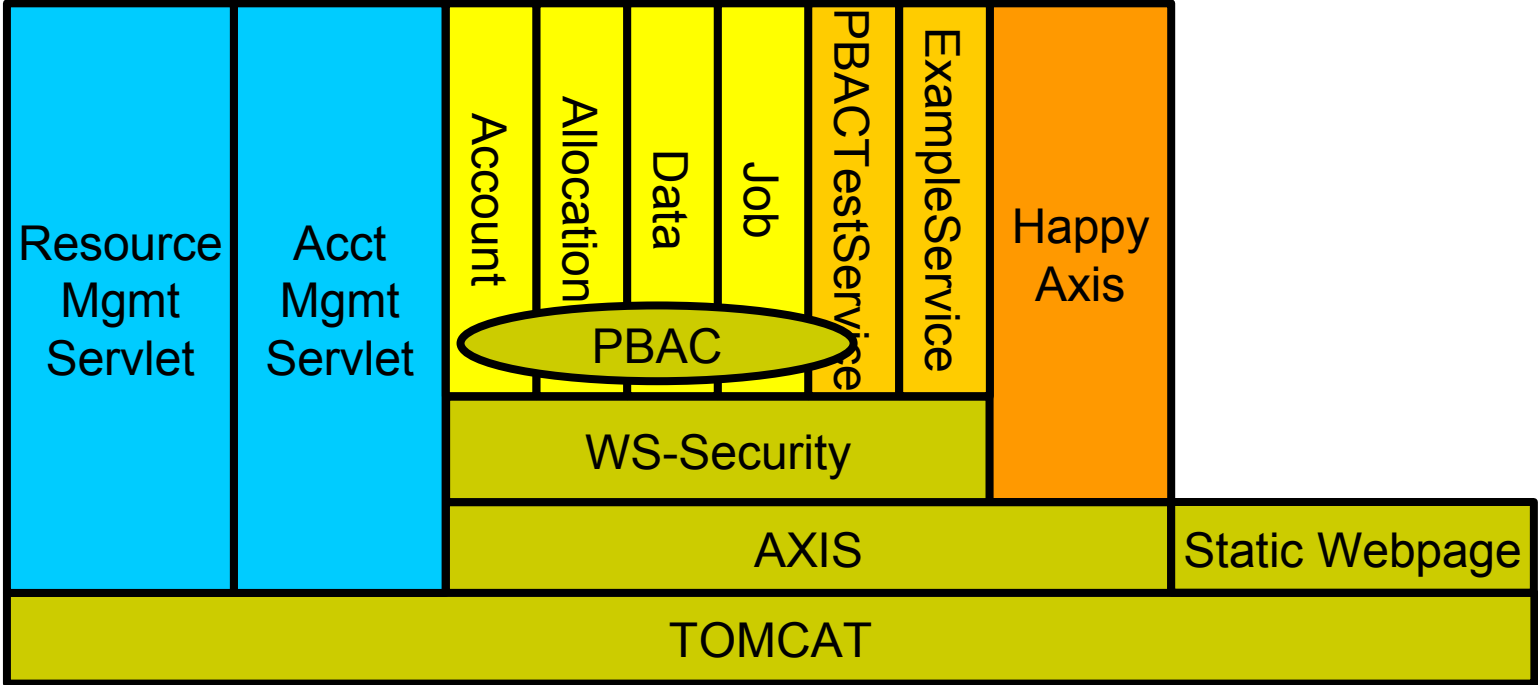


# Installing PBAC

- Tomcat started
- PBAC Example service deployed
- Tomcat restarted
- Test the PBAC installation as a client
  - Creates a conversation ID – returns a number
  - Send & receive a test message
- Tomcat stopped



# OMII Infrastructure – Services





# The Cauchy Install Script

- Pause in the installation script:
  - Do you wish to install Cauchy?
- User Input:
  - Service provide hostname:
  - JVM location:
  - OMII Server installation location:



**OMII Services install complete!**



# Verify Server Installation

- Use your client installation
  - `invoke_nonPBACTestService`
  - `invoke_PBACTestService`
  - `testservicesConnection`
- ! Make sure the container is started before running the test.



# Starting the OMII server

- Will need to start/stop container on occasions
- Commands in:
  - `OMII_BASE_HOME/jakarta-tomcat-5.0.25/bin`
  - Stop the container: `./shutdown_base.sh`
  - Start the container: `./start_base.sh`
- Verify the container has been started
  - Examine the page – `http://<HOST>:18080/`



# Examining the installation

- Make sure Tomcat is started
- View the list of deployed services:
  - Point a browser: `http://<HOST>:18080/axis`
  - Follow 'View the list of deployed Web Services' link
  - Look for the 'TestServices' service



# Managed Programme Services

- Set up the Database
  - StackInstaller Option 3
- Install the Services
  - Do you wish to install? (y/n)
  - Tomcat Username & Password
  - Select the components
- Will install whatever's there (GridSAM)



open middleware  
infrastructure  
institute uk

# Introductory OMII GridSAM Practical





# Tutorial Overview

- Overview of GridSAM
- Installation of Managed Programme client-side components (inc. GridSAM)
- Importing training certificates into the OMII client
- Overview of JSDL
- Job submission and monitoring:
  - With no file staging
  - With file staging
- Additional Java Exercise



# GridSAM Overview

- What is GridSAM?
- GridSAM Architecture
- Example Deployment



# GridSAM Overview

- What is GridSAM?
  - A Job Submission and Monitoring Web Service
  - Allows compute resources to be accessible securely over the internet
  - Single implementation of a standard for GGF Job Submission Description Language (JSDL)
  - Implementation will track an evolving interface for the OGSA Basic Execution Services (BES)
    - While maintaining the existing interface



# GridSAM Overview

- GridSAM OMII context
  - Developed by the LeSC (London e-Science Centre)
  - Funded by the OMII Managed Programme
  - Available as part of the OMII 2.x release



# GridSAM Overview

- What is GridSAM to the resource owners?
  - A Web Service to expose heterogeneous execution resources uniformly
    - Single machine through Forking or SSH
    - Condor Pool
    - Grid Engine 6
    - Globus 2.4.3
  - Acts as a client managing these resources



# GridSAM Overview

- What is GridSAM to end-users?
  - A set of end-user tools and client-side APIs to interact with GridSAM Web Services
    - Submit and start jobs
    - Monitor jobs
    - Terminate jobs
    - File transfer
    - Client-side submission scripting
    - Client-side Java API



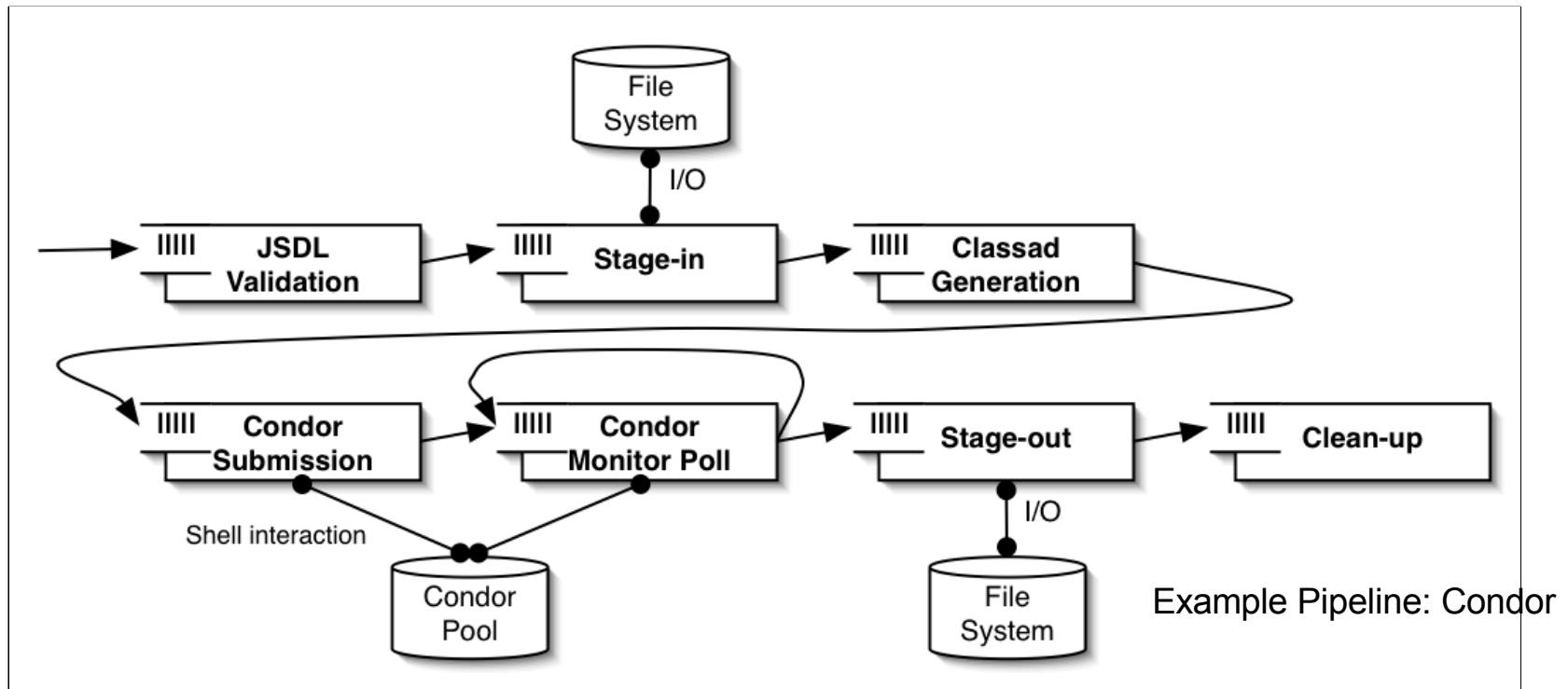
# GridSAM Overview

- GridSAM is not...
  - a scheduling service
    - That's the role of the underlying launching mechanism
    - That's the role of a super-scheduler that brokers jobs to a set of GridSAM services
  - a provisioning service
    - GridSAM runs what's been told to run
    - That's the role of a higher-level entity to provision software and the dependencies

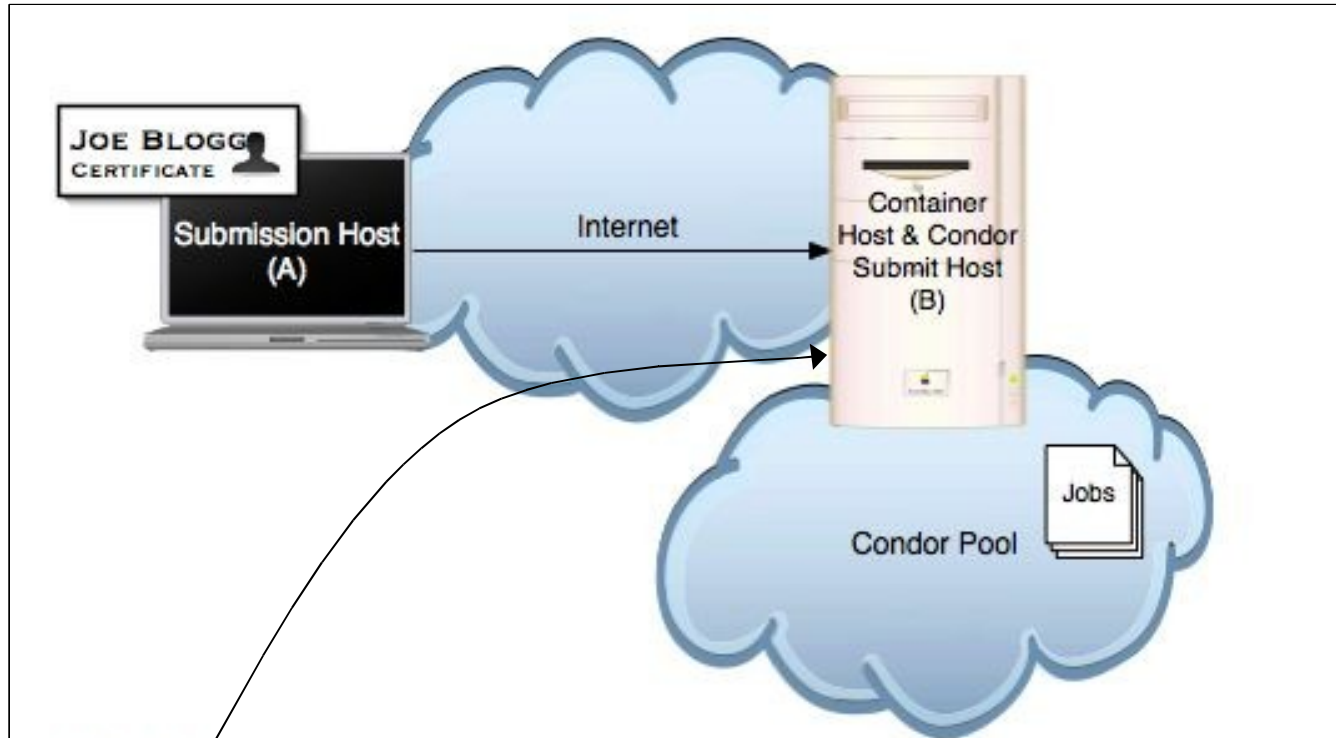


# GridSAM Architecture

- A staged event-driven architecture
  - Submission pipeline is constructed as a **network of stages** connected by **event queues**
  - Each stage performs a specific action upon incoming events



# Scenario: Condor Pool



# Introductory GridSAM Practical



- Installation of GridSAM client
- Importing training certificates into OMII client
- Overview of JSDL
- Job submission and monitoring:
  - With no file staging
  - With file staging



# Installation of GridSAM Client

- Installs into existing Windows/Linux OMII Client. On client type:

```
> cd ~/omii-client-2.3.3/managed_programme
```

## *Windows:*

```
> OMIManagedProgrammeInstall.bat
```

## *or Linux:*

```
> ./OMIManagedProgrammeInstall.sh
```

- Installs GridSAM into ~/OMIICLIENT/gridsam (and others)



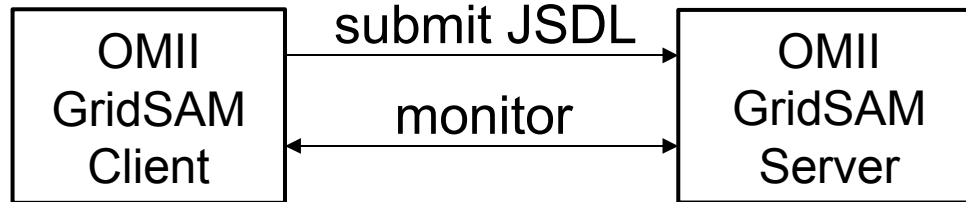
# Overview of JSDL

- JSDL 1.x: an official GGF (Global Grid Forum) recommendation for job description
- 1.0 spec downloadable:  
<http://www.gridforum.org/documents/GFD.56.pdf>
- Motivated by two main scenarios:
  - Interoperability between job management systems
  - Transformation/refinement of job spec within heterogeneous environments



# Job Submission/Monitoring: No File Staging

- **Objectives:** submit trivial job to GridSAM and monitor its progress



- Look at `<omii_client_home>/gridsam/data/examples/sleep.jsdl`:  
`<JobDefinition xmlns="http://schemas.ggf.org/jsdl/2005/06/jsdl">`  
  `<JobDescription>`  
    `<Application>`  
      `<POSIXApplication xmlns="http://schemas.ggf.org/jsdl/2005/06/jsdl-posix">`  
        `<Executable>/bin/sleep</Executable>`  
        `<Argument>5</Argument>`  
      `</POSIXApplication>`  
    `</Application>`  
  `</JobDescription>`  
`</JobDefinition>`



# Submit to GridSAM Server

- Ensure Java is on your path
- Type: `> cd ~/OMIICLIENT/gridsam/bin`
- Submit to GridSAM server (one line):  
`> ./gridsam-submit -s "http://demo-2-3-3.omii.ac.uk:18080/gridsam/services/gridsam?wsdl" -j ../data/examples/sleep.jsdl`
  - **Server: demo-2-3-3.ac.uk Port: 18080**
  - (On Windows: remove preceding ./)
- Unique job ID is returned
  - i.e. UID is *urn:gridsam:<characters>*



# Monitoring the Job

- Monitor job until completion:

```
> ./gridsam-status -s
```

```
"http://demo-2-3-
```

```
3.omii.ac.uk:18080/gridsam/services/gridsam?wsdl" -j
```

```
<unique_job_id>
```

- On Windows: remove preceding ./
- `<unique_job_id>` is entire *urn:gridsam:<characters>* string
- Job progress indicated by current state:
  - Pending, Staging-in, Staged-in, Active, Executed, Staging-out, Staged-out, Done



# Developer API

- Java API to access GridSAM services
  - Used within the command line clients
- Upcoming preview of SAGA specification
  - SAGA: Simple API for Grid Applications
  - Global Grid Forum Working Group
  - Java API to GridSAM
    - Could provide API implementation to other infrastructures, e.g. Globus



open middleware  
infrastructure  
institute uk

# Where to go from here?

---



# If you have questions...

- Look at the (extensive) documentation later
- Talk to me about your needs
  - Feature requests, enhancements, etc.
- Let me know what we can do to help you:
  - Professional services
  - Contact with the developer teams
- Web: <http://www.omii.ac.uk>
  - Follow links → Downloads → Feedback
- Email: [support@omii.ac.uk](mailto:support@omii.ac.uk)



# Finally...

- Any questions...