Astro Runtime as an Application Interface

Noel Winstanley
nw@jb.man.ac.uk
AstroGrid, Jodrell Bank, UK
AstroGrid

- A nearly complete Virtual Observatory (VO) system
- http://software.astrogrid.org
- Built upon Web Services – callable via SOAP
- Conforms to international (IVOA) standards where defined, elsewhere proposes new standards
- Useful for real research
- Security - final part that needs to be done.
  - Authentication & Authorisation
- The presentation shows how AstroGrid provides access to remote applications and data collections – the underpinnings of workflow.
Registry

- A hierarchical database (XML).
- Contains records that describe
  - data collections (coverage, catalogue structure, access methods)
  - remote applications (purpose, parameters, invocation methods)
  - supporting web services – storage, security, etc
  - other useful resources – e.g. client-side applications, organizations.
- Accessed by Web Service
- Query using XQuery, Keywords, or ADQL
- Used to locate (resolve) all other VO Web Services
- Exchanges records with registries in other VO projects (harvesting)
Myspace

- Distributed, location transparent, file storage
  - Each user has a single folder hierarchy – maintained by a filemanager service
  - Files in the hierarchy may be stored at different locations
    - each location is a filestore
  - clients typically interact only with the filemanager.
- AstroGrid services can read / write to Myspace
  - Place to stage results of long-running queries & computations
  - Used as a buffer for intermediate products of workflows
- Enables data to be kept near processing tools
- Being standardized in IVOA as VOSpace / VOStore
Remote Applications

- Common Execution Architecture - CEA
- Uniform method of describing and providing access to remote applications. Encompasses:
  - dataset access (e.g. Querying a catalogue database or image collection)
  - data processing – by VO-enabling popular legacy codes (e.g. X-matching, source extraction, simulations)
- CEA applications can be invoked from client scripts, UI, and server-side workflows
- Working System, with installed base.
- Described in IVOA Note
- Next generation IVOA proposal 'Universal Worker Service'
CEA Servers

- A CEA Server may provide one or more applications
- Asynchronous invocation
  - CEA server provides progress monitoring, notifications, and control
  - Results can be retrieved from the service, or staged to myspace, ftp server, etc.
  - All interfaces are standard web services.
- Servers, and the applications they provide, are described in the registry.
- Clients can query registry to find applications, and servers that provide them.
- Applications can be replicated across multiple servers, for fault-tolerance.
Execution schematic

Client → Registry

Registry → CEA Server

CEA Server → CEA Server

CEA Server → VOSpace

VOSpace → Registry

Registry → Client

Client → VOSpace

VOSpace → Client

Client → Read Inputs

CEA Server → Write results

Read Inputs → Registry

Write results → CEA Server

Notification → Monitor

Monitor → Invoke

Invoke → Resolve

Resolve → Client
Installed Base - 2006-05-10

- 50 CEA servers,
  - UK – portsmouth, ral, mssl, edinburgh, cambridge, leicesterc, jodrell
  - France, Russia,
- 70 applications available
  - Datasets – FIRST, INT-WFS, Ledas, SDSS,
  - Images – GOODS, HST-UDF, SDSS, XMM, Merlin
  - Existing Astro Apps – Sextractor, HyperZ, Galaxev, Pegase, BPZ, ACE, SWarp
  - New Apps – solar movie maker
  - General purpose – GNUPlot, R,
Accessing AstroGrid Services
User Perspective

- Two alternative user interfaces
- Portal – a web interface, accessed through browser
  - handy for occasional use
  - technical limitations of the web (pre-AJAX) make it awkward for advanced tasks
- Workbench – GUI client
  - Java WebStart Application
  - rich user applications
    - data discovery – astroscope & helioscope
    - application launcher – run a single CEA application
    - workflow builder – combine CEA applications
    - myspace browser
  - scripting access to VO
- PLASTIC Hub – data exchange with Topcat, Aladin, Vospec, Visivo
Workbench Screenshots

The text content of the document is not legible due to the nature of the images provided. It appears to be a collection of screenshots related to a software application, possibly for a scientific or educational purpose. The images show various interfaces and tools, likely related to data analysis or management, given the context provided.
Developer / Expert Perspective

- Three Alternatives
  - Call SOAP services directly, using WSDL
    - most basic – exposed to most complexity, necessary to understand services interact (e.g. resolution)
    - security – can require advanced SOAP handling.
  - Call AstroGrid delegate libraries
    - hides some complexity,
    - maybe not the cleanest or most reusable interface – developed for internal use.
    - Java-only – requires whole AstroGrid library stack
  - Use the Astro Runtime (AR)
    - Uniform facade interface to the VO.
    - Simpler to learn & provides extra functionality
    - Language Neutral
Astro Runtime (AR)

- A common facade library for virtual observatory services
  - aim to integrate all IVOA standards, popular ad-hoc services, and suitable helper functions.
- uniform abstraction level and types
  - cleaner API, less special cases, lower learning curve
- Shared component – single signon, configuration, cached registry entries, file trees
- Deployment alternatives
  - Webstartable desktop service / embeddable library / stand-alone application
  - Workbench is implemented upon AR – an AR instance is available to other clients when Workbench is running.
Design

- AR designed to be accessible from all programming languages
- Procedural design, rather than OO (astronomer friendly)
- Typical configuration - a service that runs on a user's desktop
  - accepts requests from other desktop applications
  - processes requests by calling webservices using the AstroGrid Java client libraries.
- Can also be embedded in a service implementation and called directly.
Access Methods

• At present, 3 ways to access AR functionality.
  • JavaRMI (Java, Groovy, Jython)
    – JVM-only inter-process communication
    – strongly typed
    – requires a minimal set of libraries
    – allows remote event listeners to be registered
  • XMLRPC (Python, Perl, C++, C#, Java)
    – Forerunner of SOAP: http://www.xmlrpc.com/
    – simpler types than SOAP
    – implementations for a wide range of languages
  • HTTP-Get (Shell, R, IDL, Matlab)
    – rough-n-ready procedure call
    – fallback for other languages
AR Schematic

Browser
Scripts
Users
Client Applications

AR
HTML
GUI
XMLRPC
JavaRMI

AstroGrid
IVOA
CDS
Dialogs
UI
System

Community
Registry
VOSpace
VOStore
IVOA..
JES
CEA
CDS
SIAP
SSAP

<- Desktop
Network ->

Noel Winstanley - nw@jb.man.ac.uk
AR Abilities

- AstroGrid
  - CEA: query, build, execute, monitor
  - Workflow – submit, monitor.
  - MySpace: read, write, list, create, delete
  - Registry: query, xquery, resolve
- IVOA – SIAP, SSAP; SkyNode to follow.
- CDS – Simbad, Vizier, coordinates, UCD.
- NVO – cone; NED, other ad-hoc to follow
- PLASTIC – send data to other client-side applications
- UI - control workbench user interface, display dialogues to prompt for input
Connecting to AR - Code

Don't Panic!
Java: RMI

```java
public class Connect {
    public static void main(String[] args) {
        try {
            Finder f = new Finder();
            ACR acr = f.find();
            // retrieve a service - by specifying the interface class
            Configuration conf =
                (Configuration)acr.getService(Configuration.class);
            // call a method on this service.
            Map l = conf.list();
            for (Iterator i = l.entrySet().iterator(); i.hasNext(); ) {
                System.out.println(i.next());
            }
            // retrieve another service from the acr - this time by name
            Registry registry = (Registry)acr.getService("astrogrid.registry");
            // use this service..
            URI u = new URI("ivo://org.astrogrid/Pegase");
            System.out.println(registry.getResourceInformation(u));
            // returns a struct of data.
            // registry.getRecord(u) returns a org.w3c.dom.Document.
            u = new URI("ivo://uk.ac.le.star/filemanager");
            System.out.println(registry.resolveIdentifier(u));
            // returns a java.net.URL

        } catch (Exception e) {
            e.printStackTrace();
        }
        // shut the app down - necessary, as won't close by itself.
        System.exit(0);
    }
}
```
Python: XML-RPC

1. Import xmlrpc library
2. Read AR configuration file
3. Construct xmlrpc endpoint
4. Create client
5. Get reference to service
6. Call service function

```python
#!/usr/bin/env python
# Noel Winstanley, Astrogrid, 2005
# minimal example of connecting to acr and calling a service.
import xmlrpclib
import sys
import os

# parse the configuration file.
prefix = file(os.path.expanduser("~/.astrogrid-desktop")).next().rstrip()
endpoint = prefix + "xmlrpc"
print "Endpoint to connect to is", endpoint

#ac connect to the acr
acr = xmlrpclib.Server(endpoint)

# get a reference to the registry service from the acr.
registry = acr.astrogrid.registry

# call a method
print registry.getResourceInformation('ivo://org.astrogrid/Pegase')
    # returns a struct of data

print registry.getRecord('ivo://org.astrogrid/Pegase')
    # return the xml of a registry entry (string)

print registry.resolveIdentifier('ivo://uk.ac.le.star/filemanager')
```
Uses for AR

- Environment for implementing VO Services
  - Workflow engines, VOEvent processors.
- VO-enable existing client-side applications
  - Topcat (.uk), Aladin (.fr), Visivo (.it), VoSpec (.es), ...
- New VO client-side Applications
  - AstroGrid Workbench, Sampo (.fi)
- Commandline utilities for VO
- Client-side Scripting (expert users) – (python, perl)
  - Script contains control flow
  - performs work by accessing CEA applications via AR
  - more interactive operation than batch Workflows
- Access VO in existing wrapper environments. - e.g. PyRaf, Parceltongue
Future Plans

- Will maintain backwards compatibility
- To add
  - missing service types - SkyNode
  - other ad-hoc astronomy webservice
  - VOTable, Xpath, XML helpers
  - make workbench UI components more controllable
- Track developing standards (VOSpace, SSO)
- Multi-session ACR for server-side deployment.
- Improve error reporting.
- Submit AR interfaces to IVOA in some way
References

- CEA
  - http://www.ivoa.net/Documents/latest/CEA.html
- Workbench Launch
  - http://software.astrogrid.org/userdocs/workbench.html
- AR Site – docs & download
  - http://software.astrogrid.org/beta/ar
- AR Wikipage
  - http://wiki.astrogrid.org/bin/view/Astrogrid/AstroClientRuntime
- Plastic
  - http://plastic.sourceforge.net