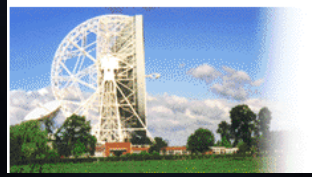
PPARC

A PPARC funded project

The Astro Runtime
for application developers

Noel Winstanley
Jodrell Bank, AstroGrid

with the part of Noel played by
John Taylor,
IfA Edinburgh/AstroGrid



Jodrell Bank Observatory

THE UNIVERSITY OF EDINBURGH
University of Leicester
University of BRISTOL
UNIVERSITY OF CAMBRIDGE
UNIVERSITY of LEEDS
UNIVERSITY of EXETER
CCLRC Rutherford Appleton Laboratory
THE UNIVERSITY of MANCHESTER
University of Portsmouth
Queen's University Belfast
UCL MSSL

The Astro Runtime

uniform access to all VO services, from all programming languages

- How do applications access VO services?
- Why do we need the Astro Runtime?
- One size fits no-one: Astro Runtime variants
- Examples of using the Runtime

How do we access VO Services?

- Use published WSDL to generate own SOAP client, call services directly
  - need to understand how AG services interact
  - security – needs advanced SOAP handling
  - SOAP difficult or impossible from some platforms
  - Many protocols to learn
- Call methods on the Astro Runtime
  - Clean Facade Interface to VO Services
  - Provides extra benefits
- Info http://software.astrogrid.org/developerdocs/

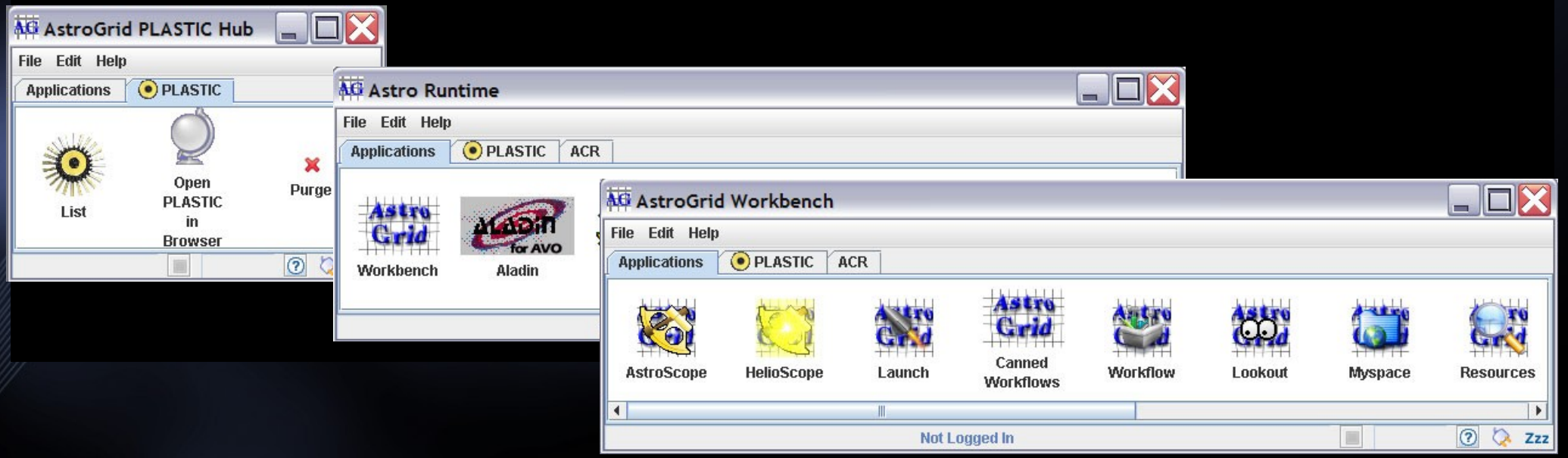
AstroRuntime Terminology

- ACR (Astro Client Runtime) is a desktop service that makes it simple for other programs to access VO services.
- ASR (Astro Service Runtime) is the server-side equivalent – same API, but no GUI components and multi-user support.
- Workbench is a suite of GUI applications built upon the ACR
- http://software.astrogrid.org/beta/ar/
  - Single-click launch using Java WebStart
  - choose 'Workbench Launch'
  - try it now :)

Astro Runtime variants

All variants are webstartable (except ASR) and available as executable jars and embeddable libraries

Variant	Size	Plastic Hub	Access to VO services AG,CDS,NVO,IVOA	Dialogs myspace browser...	Apps AstroScope...
Plastic Hub	4M				
ASR	13M				
ACR	20M				
Workbench	24M				

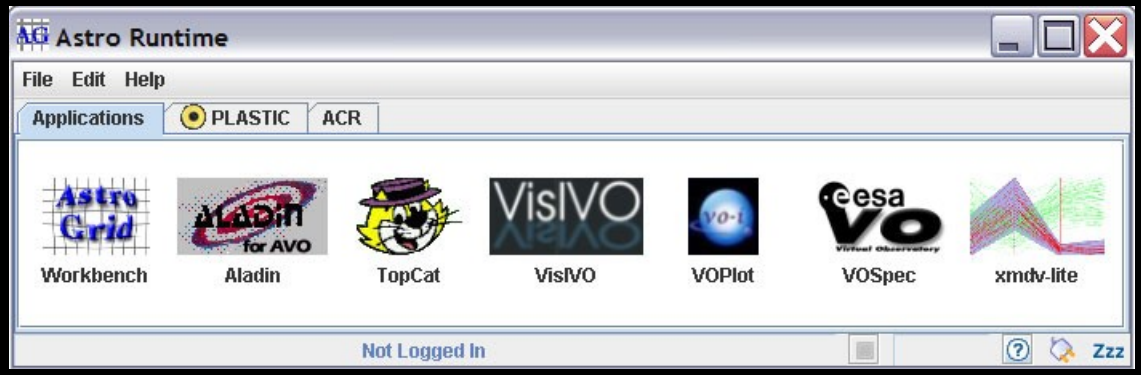


Astro Runtime variants

Only discuss these in this talk

All variants are webstartable (except ASR) and available as executable jars and embeddable libraries

Variant	Size	Plastic Hub	Access to VO services AG,CDS,NVO,IVOA	Dialogs myspace browser...	Apps AstroScope...
Plastic Hub	4M				
ASR	13M				
ACR	20M				
Workbench	24M				



ACR – Purpose

- A uniform way to access VO components..
  - remote: web services – SOAP, REST, etc
  - client side: GUI components; dialogues; helper libraries
- ... from any programming, scripting or shell language
- ... on any platform

ACR Design

- ACR designed to be accessible from all programming languages
- Procedural design, rather than OO (astronomer friendly)
- A service that runs on the user's desktop
  - accepts requests from other desktop applications
  - processes requests by calling webservices using the AstroGrid Java client libraries.
- Components
  - ACR provides a large set of components / services that can be called by any of the access methods
  - related components organized into modules.

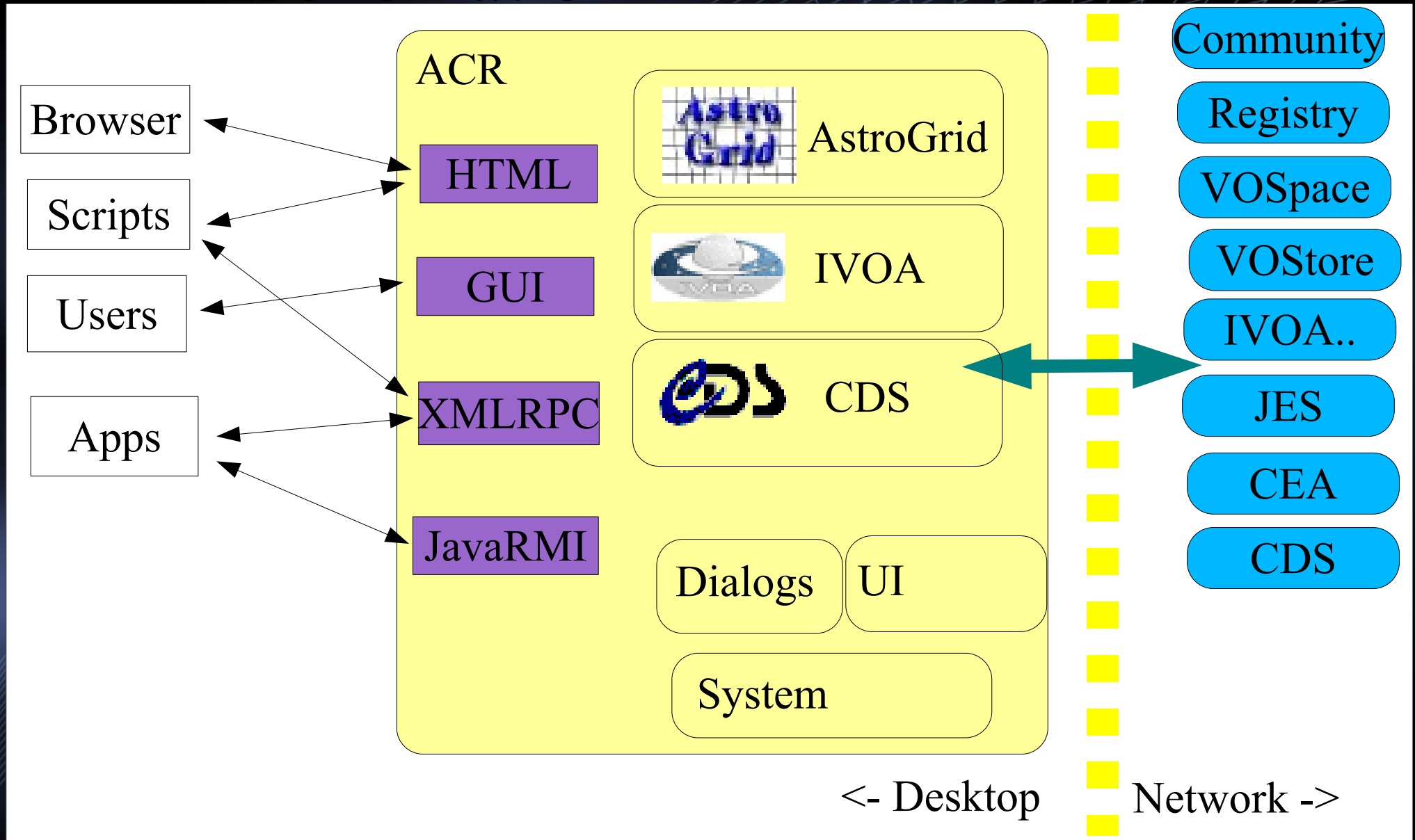
What's in it for developers?

- A library of virtual-observatory functions
- A common facade for the VO / AstroGrid
  - aim to integrate all VO standards, popular ad-hoc services, and suitable helper functions.
- uniform abstraction level and types
  - cleaner API, fewer special cases, shallow learning curve
- single configuration
  - taken care of – client programmer doesn't need to care.
- simple deployment
  - trivial to install using Java WebStart and easily embeddable
- Shared component – single signon, cached registry entries, myspace trees, insulated from change

Access Methods

- 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

ACR Schematic



What services can I get at?

- IVOA – SIAP, SSAP, skyNode, adql converter, registry
- “IVOA” - cone search, VOSpace
- AstroGrid – CEA applications and workflows
- CDS – GLU, sesame, UCD, VizieR

Code demo

In this demo I'll show you how to access VOSpace using Roy's favourite language: Python

Applications using the Astro Runtime

Searches the registry
Queries SIAP services
Saves to MySpace

VAST

Launches CEA apps
on HPC resources

Browses
MySpace

Topcat

ALADIN

Searches the registry
Queries SIAP services
Performs cone searches
Queries SSAP services

AstroScope

Contacts and references

Noel Winstanley Noel.Winstanley@manchester.ac.uk

John Taylor jdt@roe.ac.uk

The Astro Runtime

http://software.astrogrid.org/beta/ar/

http://wiki.astrogrid.org/bin/view/Astrogrid/AstroClientRuntime

API Docs

http://software.astrogrid.org/beta/ar/xmlrpc.html

http://software.astrogrid.org/beta/ar/apidocs/index.html

More references:

AstroRuntime code recipes (Java, Python, Perl, C, bash, R, Matlab...)

http://wiki.astrogrid.org/bin/view/Astrogrid/AcrRecipes

AstroRuntime tutorial

http://wiki.astrogrid.org/bin/view/Astrogrid/MakingAppsVOAwareWorksheet

Report on use of ACR in Aladin (Boch)

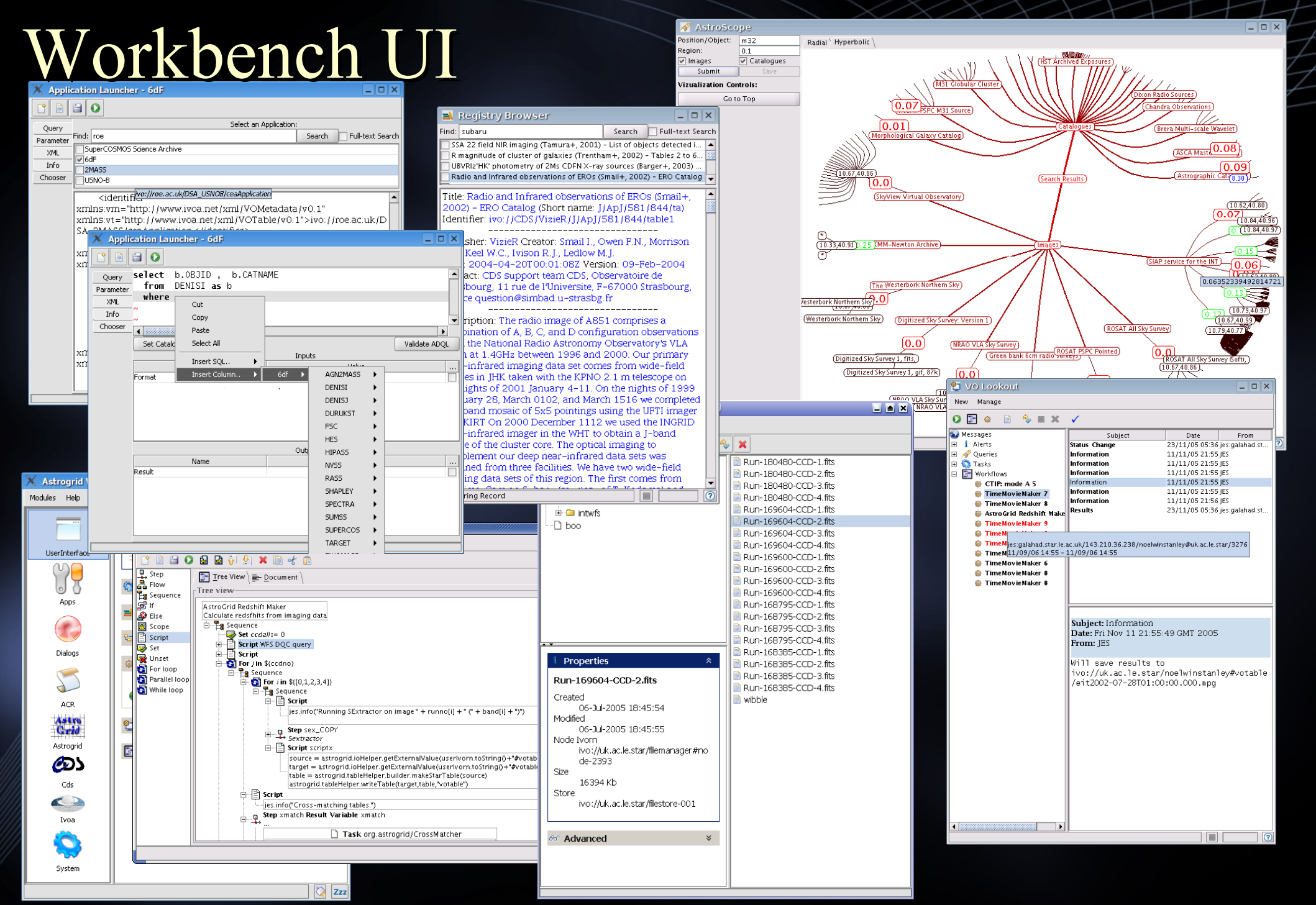
http://eurovotech.org/twiki/bin/view/VOTech/UsageOfAcrApiInAladin

Other presentations on the AR

http://www.ivoa.net/internal/IVOA/InterOpOct2005Applications/acr-voclient-ivoa-oct-2005.sxi

http://wiki.astrogrid.org/bin/view/Astrogrid/AgTechWorkshopJan06

Workbench UI



Stop

Further information including code examples follows this slide.

Java RMI

Import ACR classes

Instantiate finder

Find running ACR, or execute new

Get reference to service

Alternative way to get service

Call service function

Tell program to exit

```
import org.astrogrid.acr.Finder;
import org.astrogrid.acr.astrogrid.Registry;
import org.astrogrid.acr.builtin.ACR;
import org.astrogrid.acr.system.Configuration;

import java.net.URI;
import java.util.Iterator;
import java.util.Map;

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

Import xmlrpc library

Read ACR configuration file

Construct xmlrpc endpoint

Create client

Get reference to service

Call service function

```
#!/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

#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')
```

Perl XML-RPC – same pattern

Import xmlrpc library - alternatives?

Read ACR configuration file

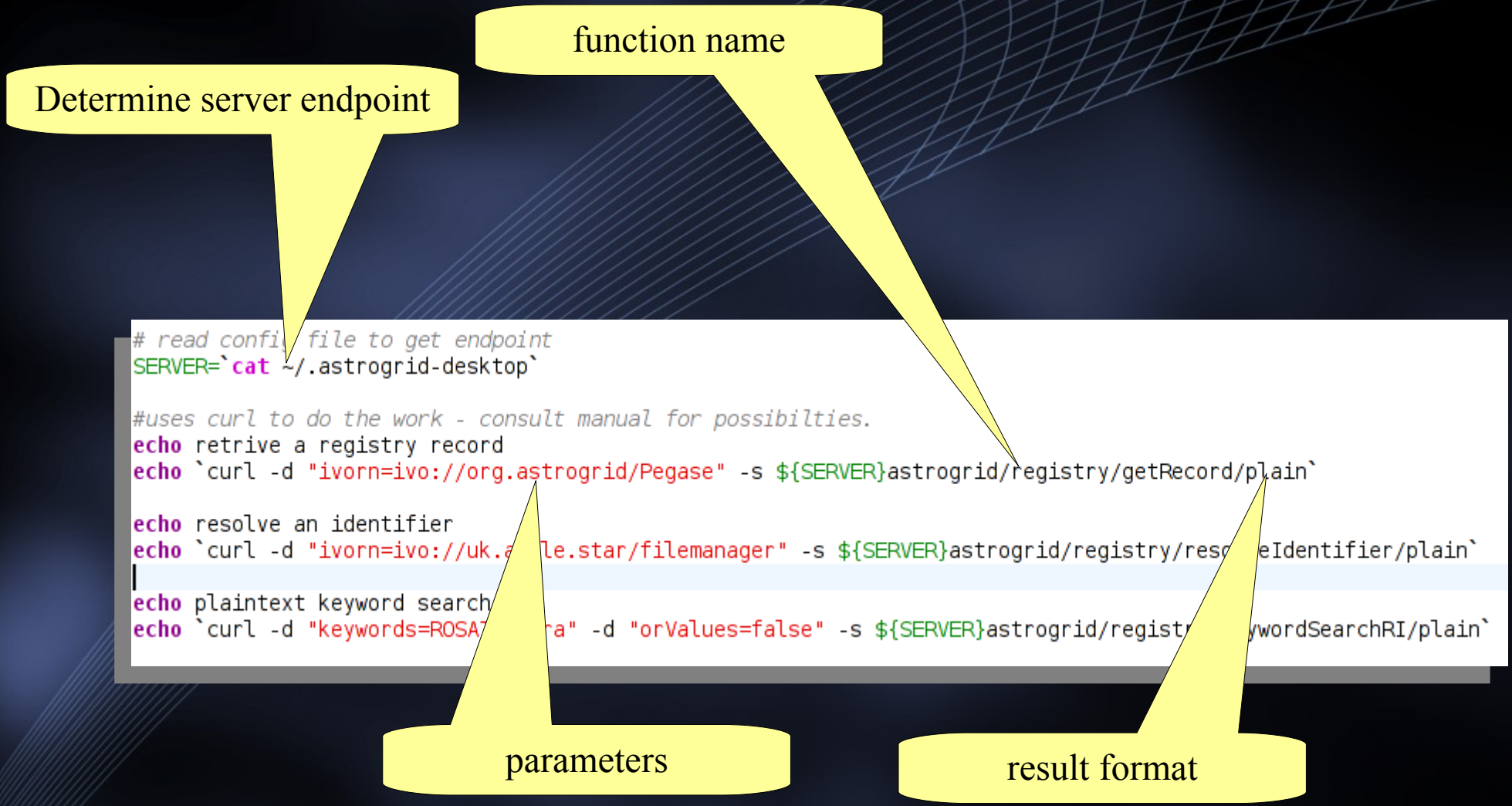
Construct xmlrpc endpoint

Create client

Call service function

```
#!/usr/bin/perl
#Noel Winstanley, Astrogrid, 2005
#basic perl example - incomplete.
#connects to acr using xmlrpc interface.

#xmlrpc client for perl, downloadable from cpan
use Frontier::Client;

# create the server
# don't know how to find current user's home dir,
#or how to read in files nicely - hope someone can show me this
open(CONFIG_FILE,"/home/noel/.astrogrid-desktop")
   || die("Could not open acr config - check ACR is running");
$prefix=<CONFIG_FILE>;
close(CONFIG_FILE);
chomp $prefix;
$url = $prefix . "xmlrpc";
#create xmlrpc client
$acr = Frontier::Client->new(url => $url);

# call some methods on the acr
$record = $acr->call('astrogrid.registry.getRecord'
                     ,'ivo://org.astrogrid/Pegase');
print $record, "\n";

$endpoint = $acr->call('astrogrid.registry.resolveIdentifier'
                     ,'ivo://uk.ac.le.star/filemanager');
print $endpoint, "\n";
```

Shell – raw HTTP



- develop this using HTML interface