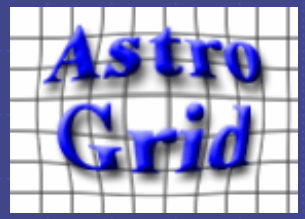


PPARC

A PPARC funded project

AstroGrid's Common Execution Architecture

Guy Rixon, reporting on behalf of Paul Harrison and the other AstroGrid developers

SC4DEVO-1, Pasadena, July 2004

AstroGrid

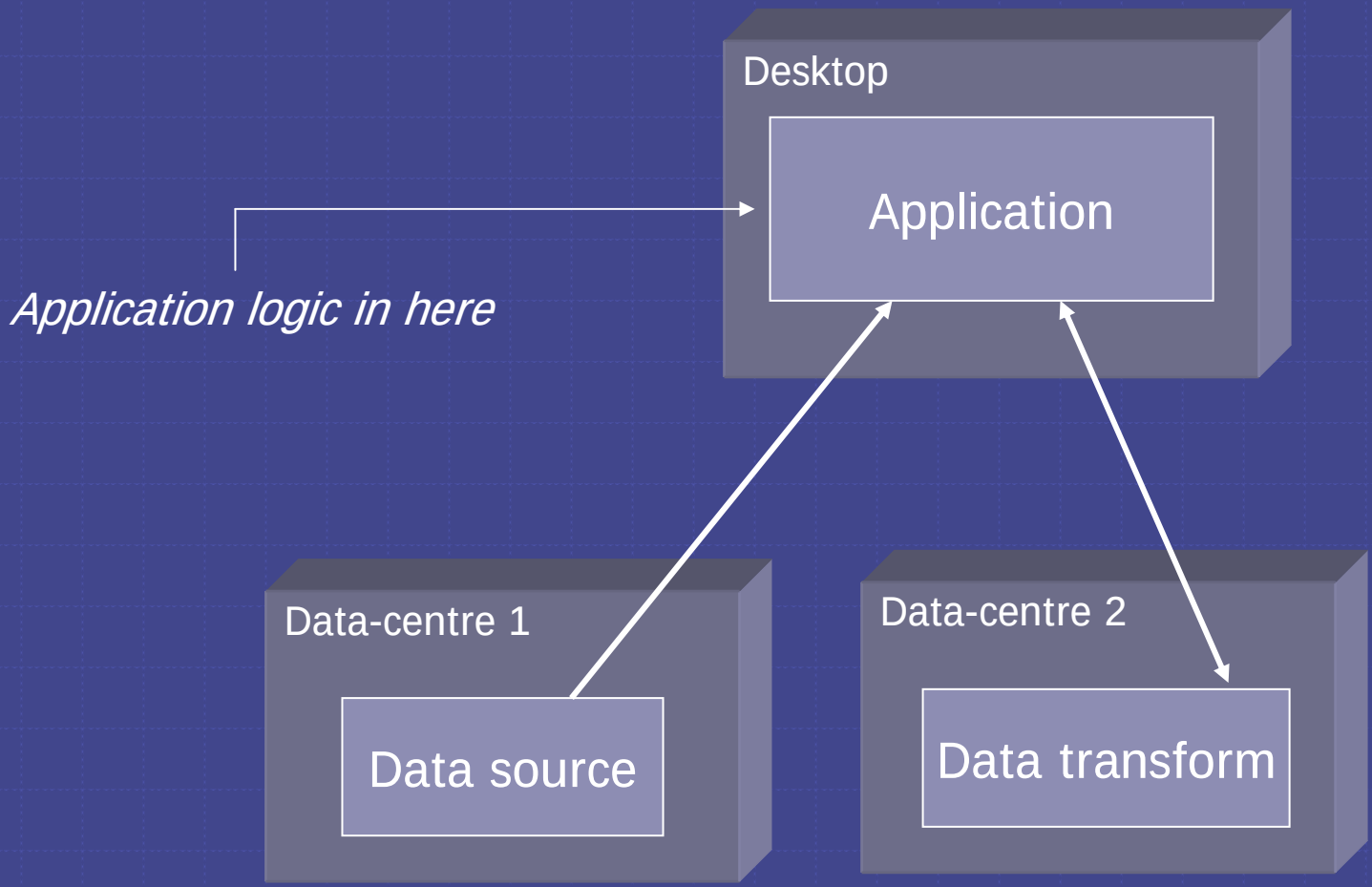
- UK national virtual-observatory project
- See cover slide for participants and funding
- Tasked to produce a VO toolkit to be reused by service providers
- Collective author of a SOA using web services
- ***Common Execution Architecture***:
  - A major part of AstroGrid architecture
  - Part of the service-oriented architecture
  - A way of making and using web services
  - Some important applications to data mining

SOA for data mining

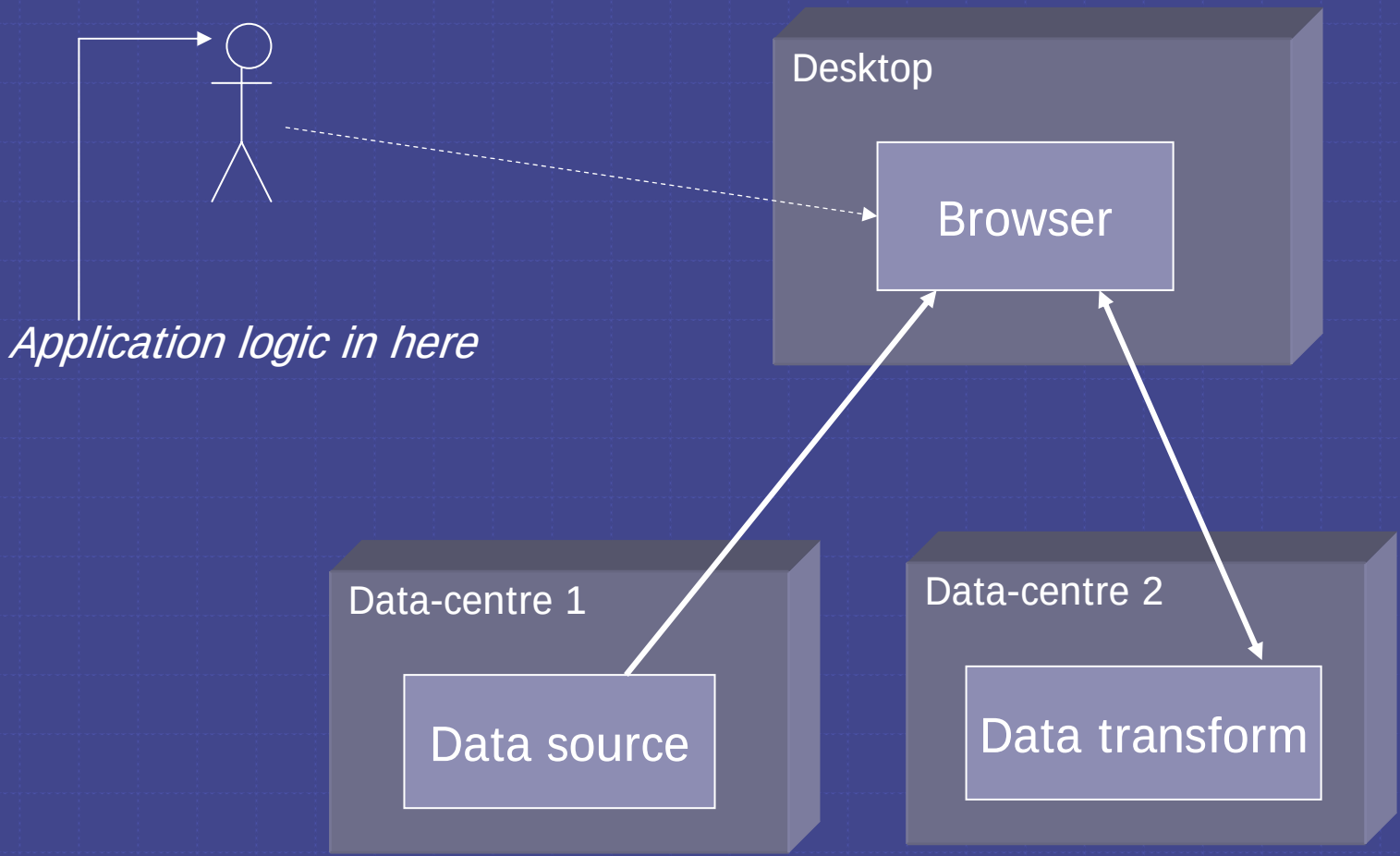
How can we carve up the distributed application into services?

And where do we put the *application* logic?

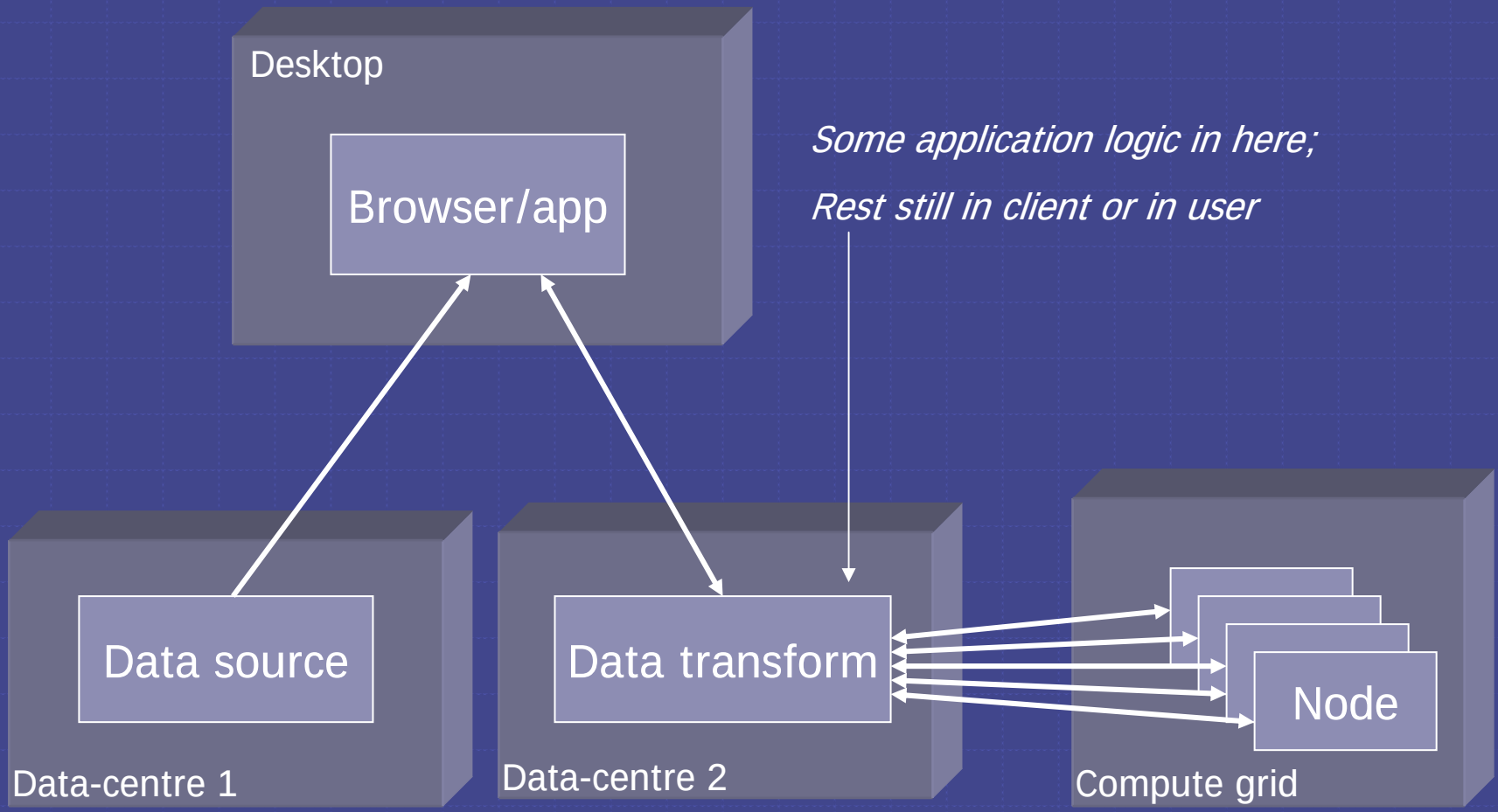
Distributed app: conservative



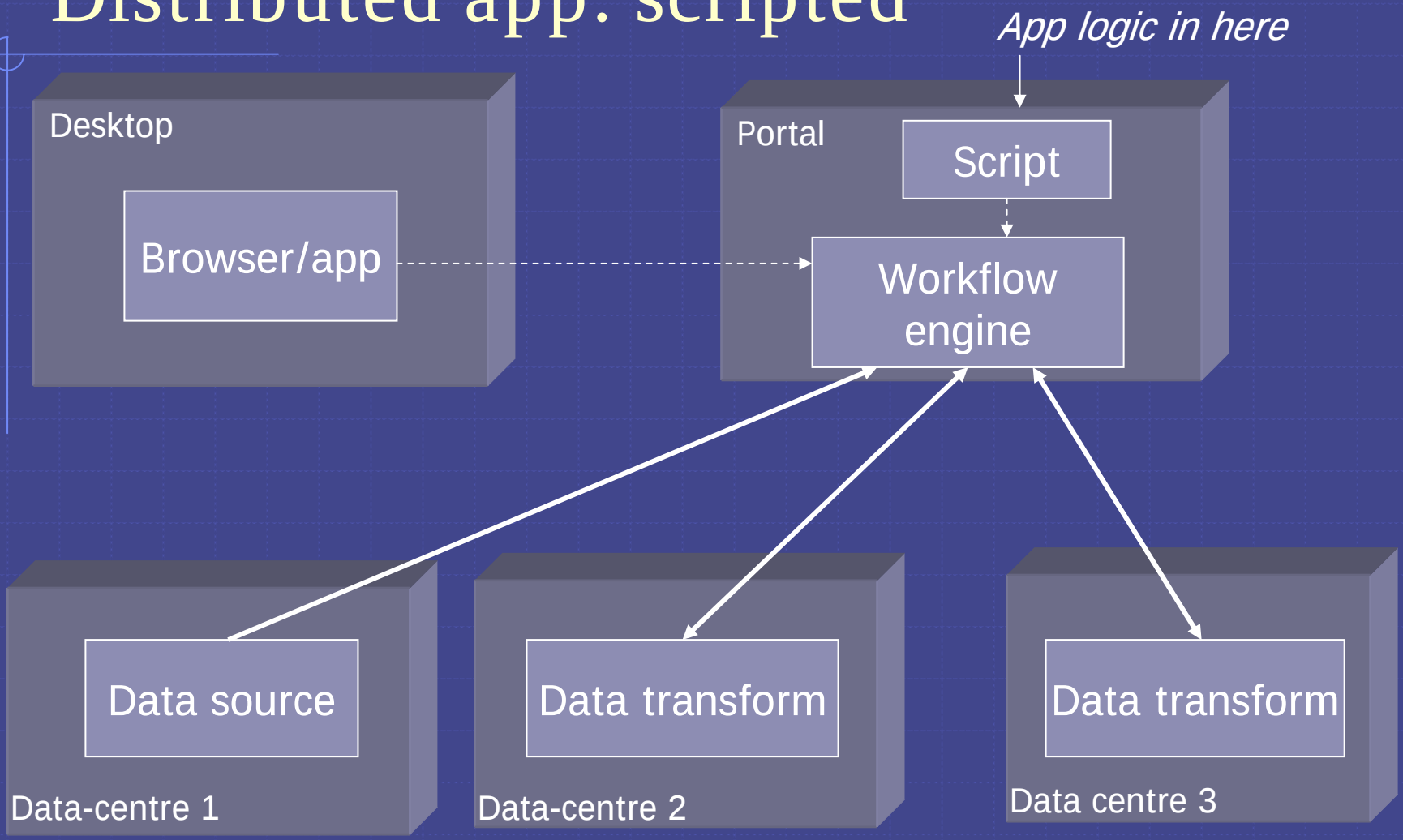
Distributed app: browser-based



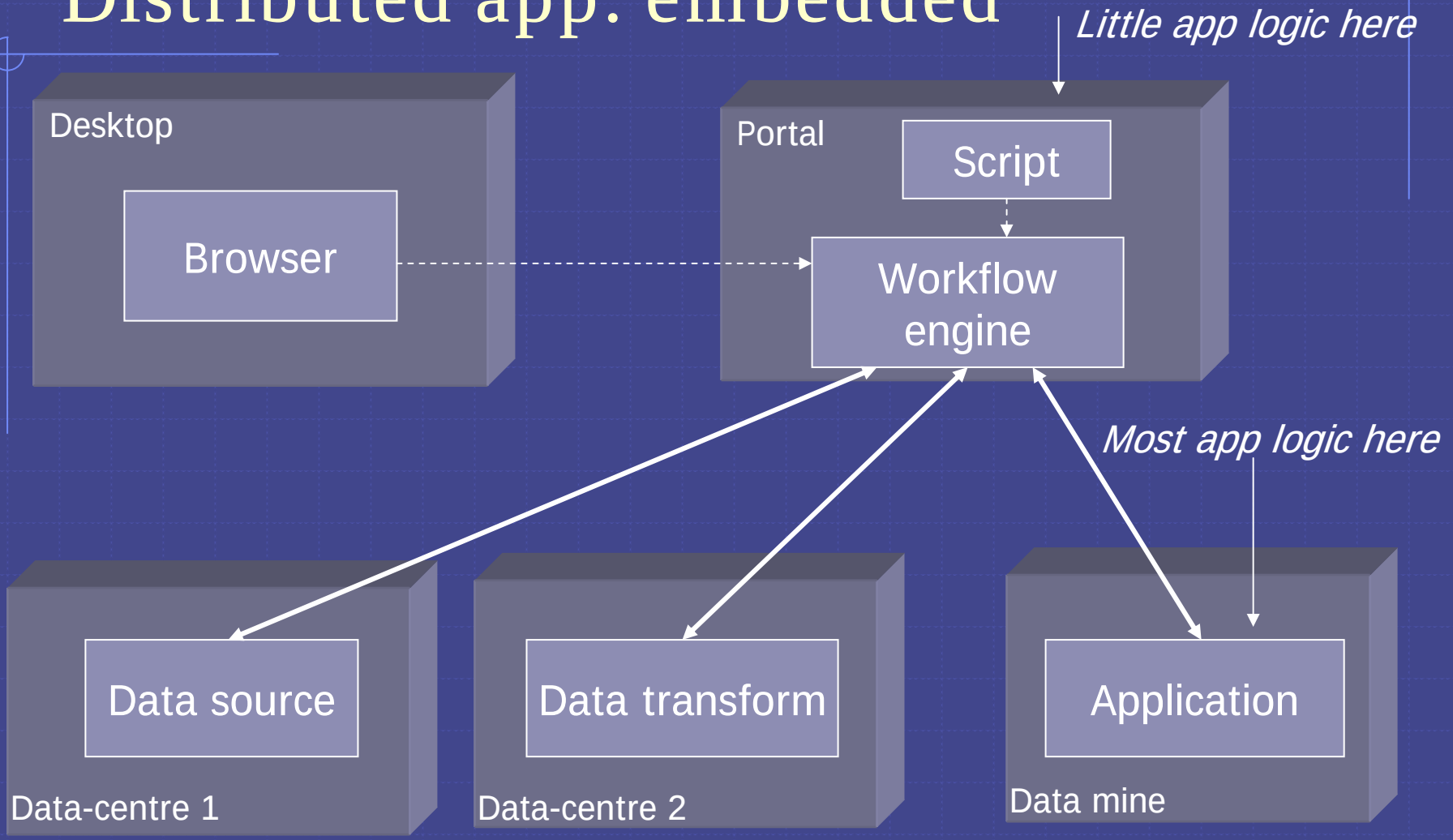
Distributed app: intragrid



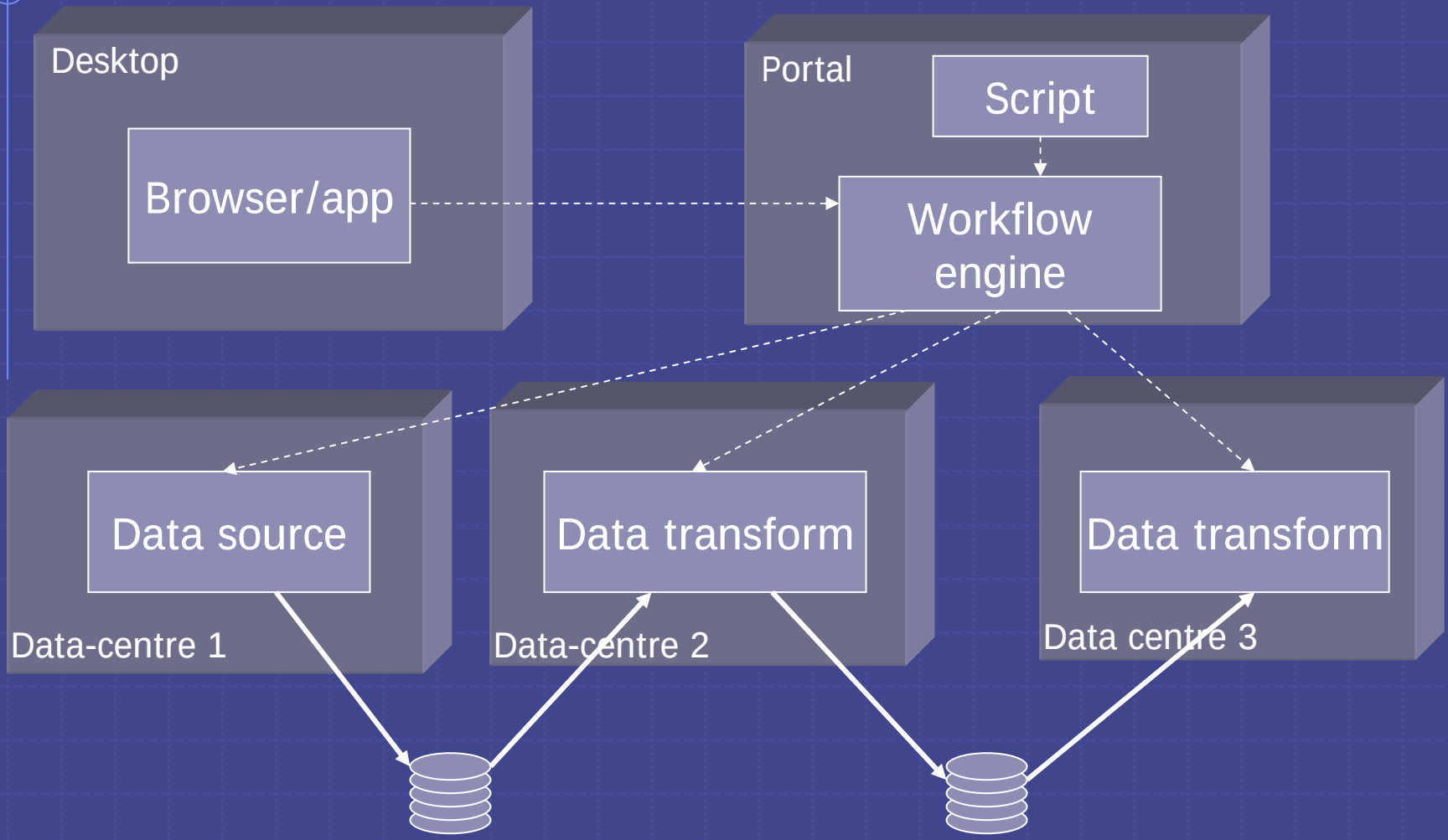
Distributed app: scripted



Distributed app: embedded



Distributed app: data grid



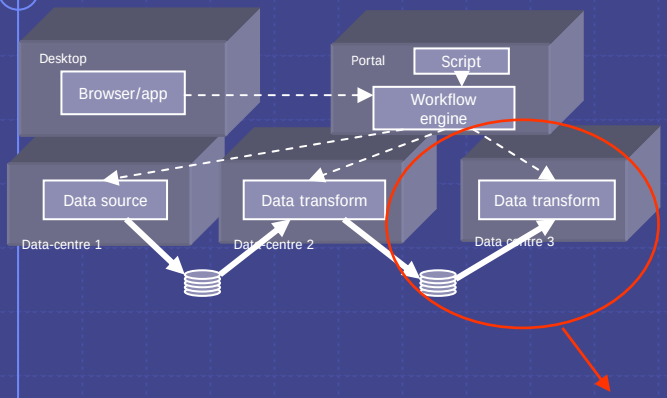
Issues to be addressed

- Boring, technical:
  - How to write apps as w/s?
  - How to write w/s clients?
  - How to connect apps to storage?
  - How to register functions?
- Interesting, scientific:
  - Who has useful algorithms?
  - How to match algorithms to data?
  - Best place to run computations?
  - How to share code?
  - Provenance?
  - How to share kudos?
  - Is more science possible?

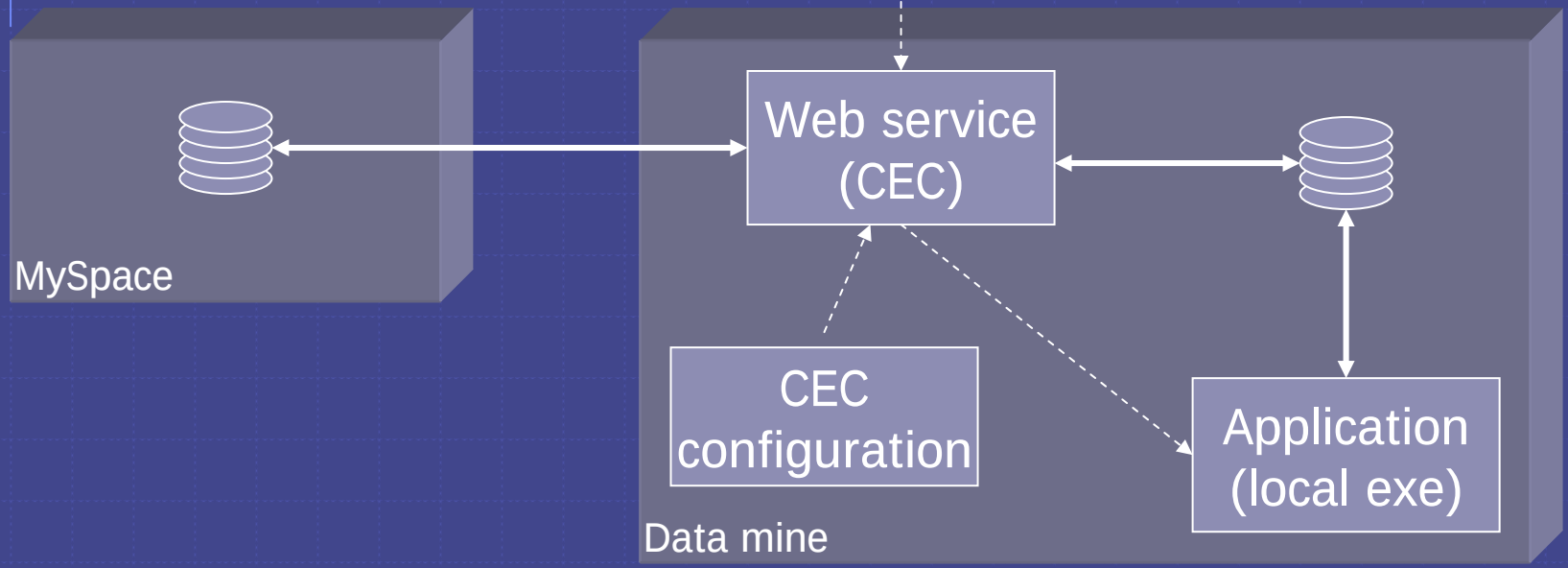
AstroGrid architecture and products

- AstroGrid bits and pieces let techies solve the tech problems so that scientists can concentrate on the science issues.
- AstroGrid supports:
  - Browser-based app
  - Scripted app
  - Data grid
  - Embedded app (pre-installed exes)
- *Common Execution Architecture* lets all parts but exe for embedded app be reusable code.

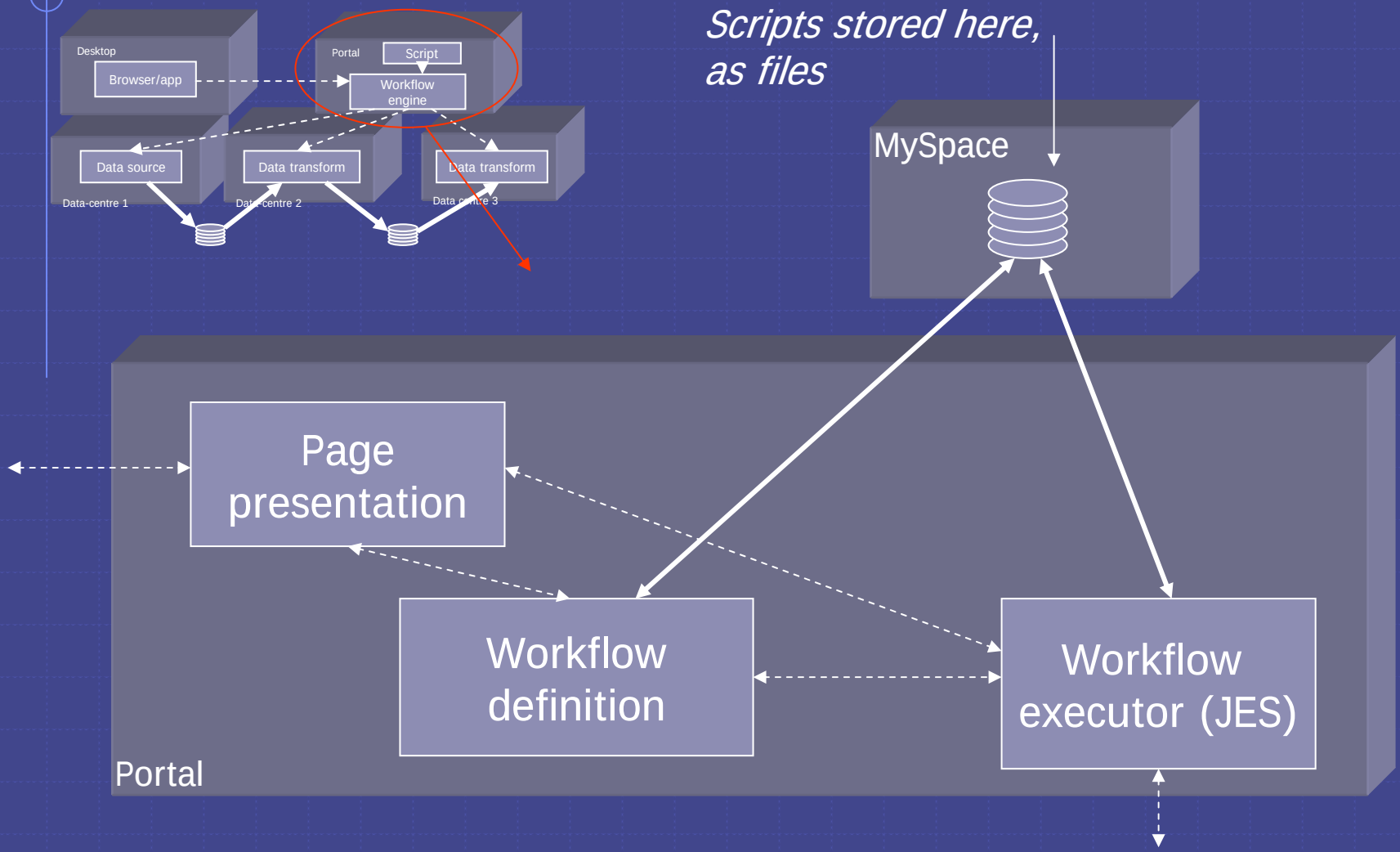
Common Execution Architecture (1)



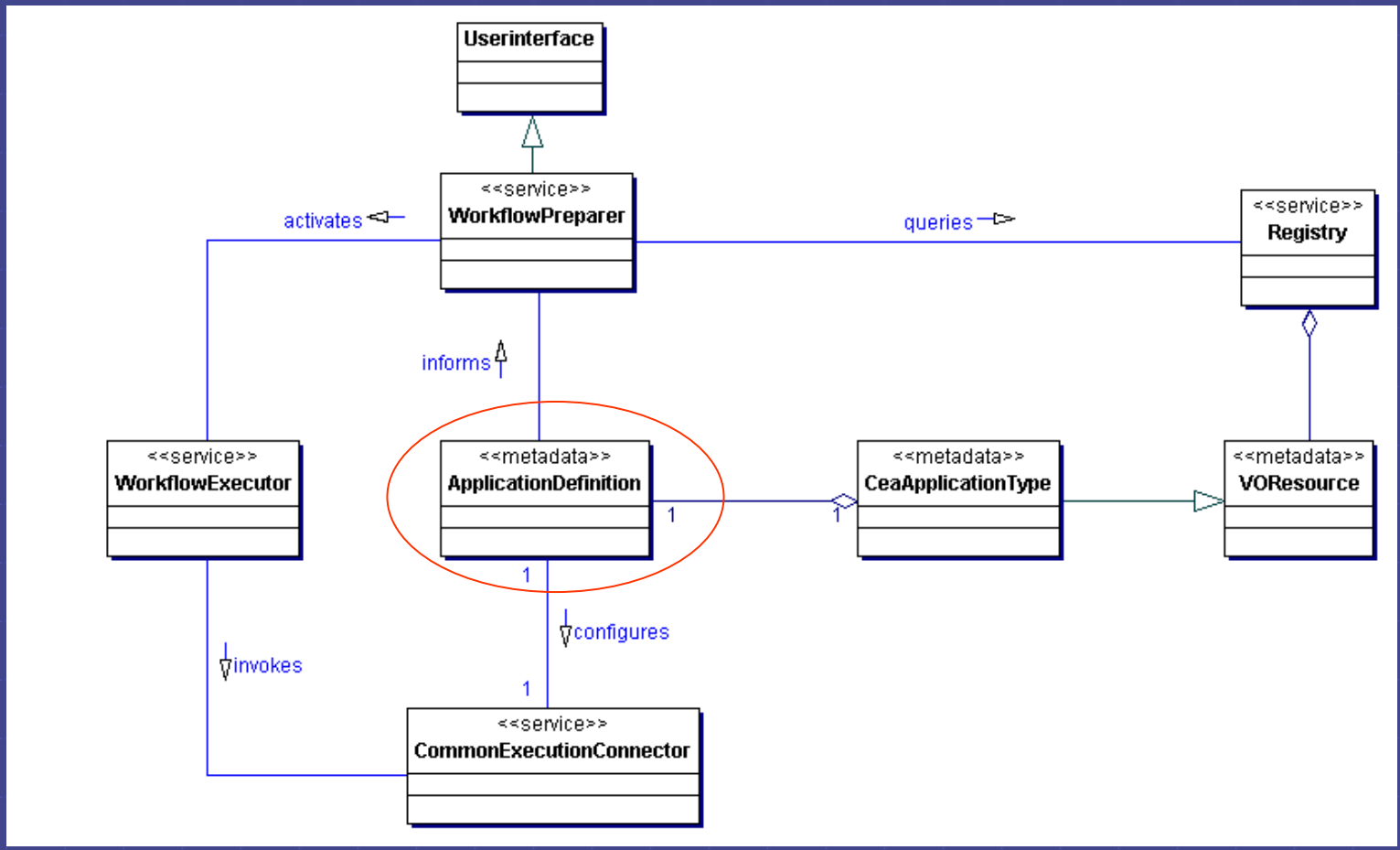
"CEC" = Common Execution Connector



Common Execution Architecture (2)



Common execution architecture (3)



Common language throughout system allows s/w reuse.

How to write an app using CEA?

- Don’t write a web service!
- Write a command-line exe
- Configure a CEC to run your app:
  - I.e. describe parameters in the CEA language
- Or: get your app configured on someone else’s CEC

How to publish an app using CEA?

- Take the *<ApplicationDefinition>* that informs the CEC.
- Wrap it in a *<VOResource>* of sub-type *CeaApplicationType.*
- Publish the *<VOResource>* in IVO registry.

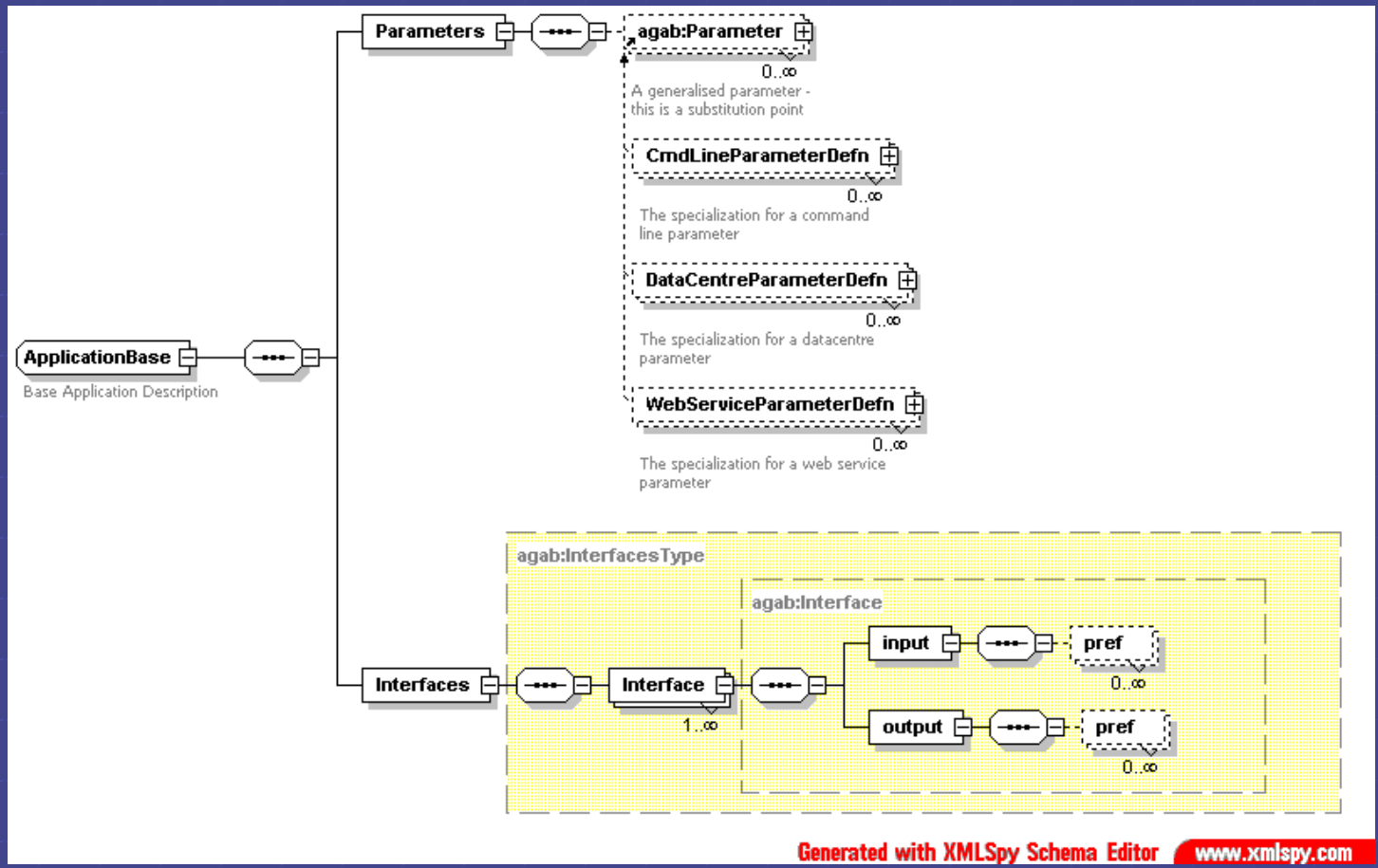
How to call an app-service using CEA?

- Look it up in the resource registry.
- Get the *<ApplicationDefinition>* from its *<VOResource>.*
- Give the *<ApplicationDefinition>* to a CEA delegate => define SOAP call.
- Add parameters from UI according to <ApplicationDefinition>.
- Bake in pre-heated SOAP engine @ HTTP 1.1.

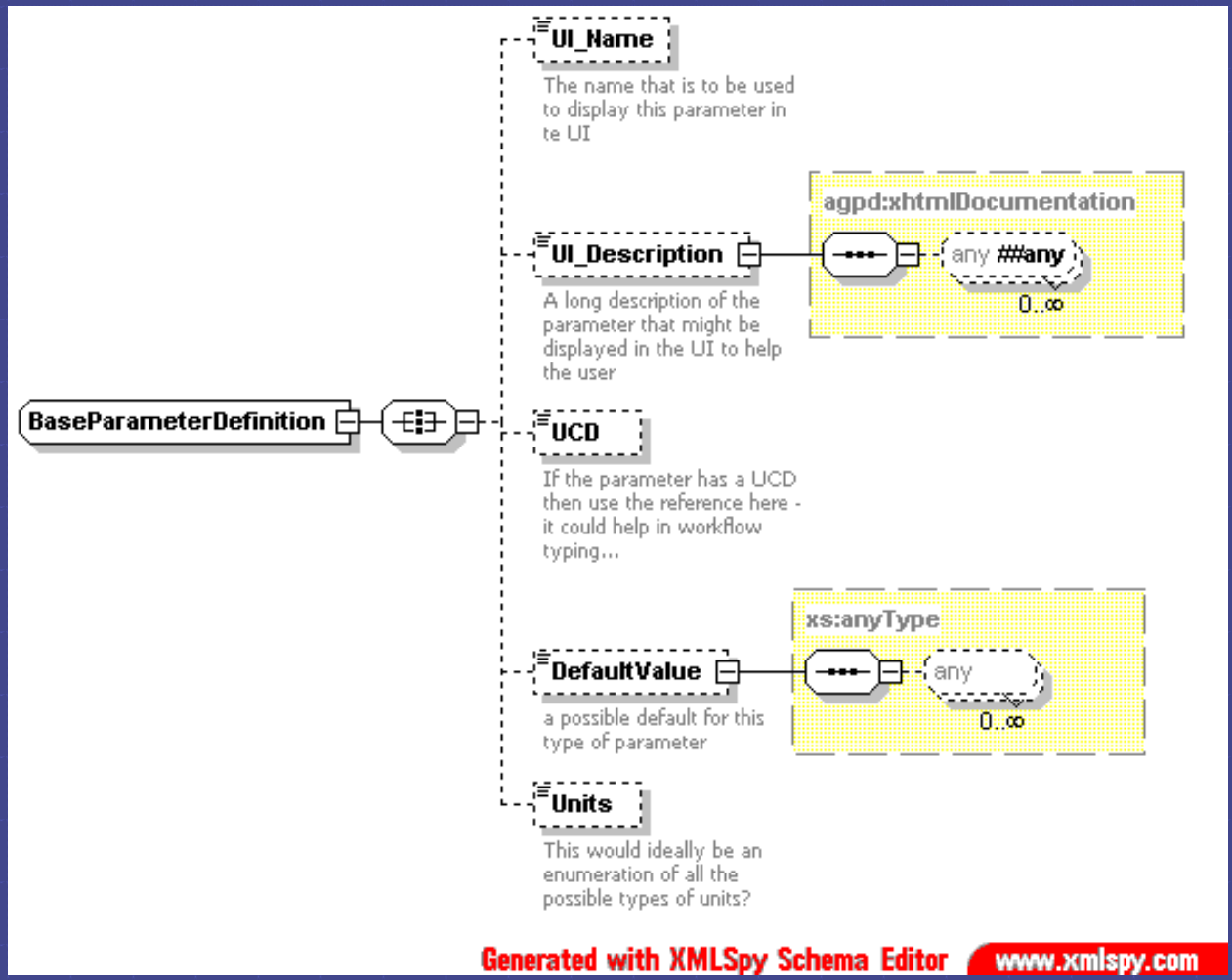
CEA language example

```
<ApplicationDefinition>
    <Parameters>
        <ParameterDefinition name="DetectionImage" type="binary">
            <UI_Name>Detection Image</UI_Name>
            <UI_Description>The image that is used to detect sources.
                Basic position, shape and size information is derived from this
                image</UI_Description>
        </ParameterDefinition>
        <ParameterDefinition name="PhotoImage" type="binary">
            <UI_Name>Measurement Image<UI_Name>
            <UI_Description>The Image that is used to measure photometric
                                parameters</UI_Description>
        </ParameterDefinition>
        <ParameterDefinition name="config_file" type="text">
            ...
```

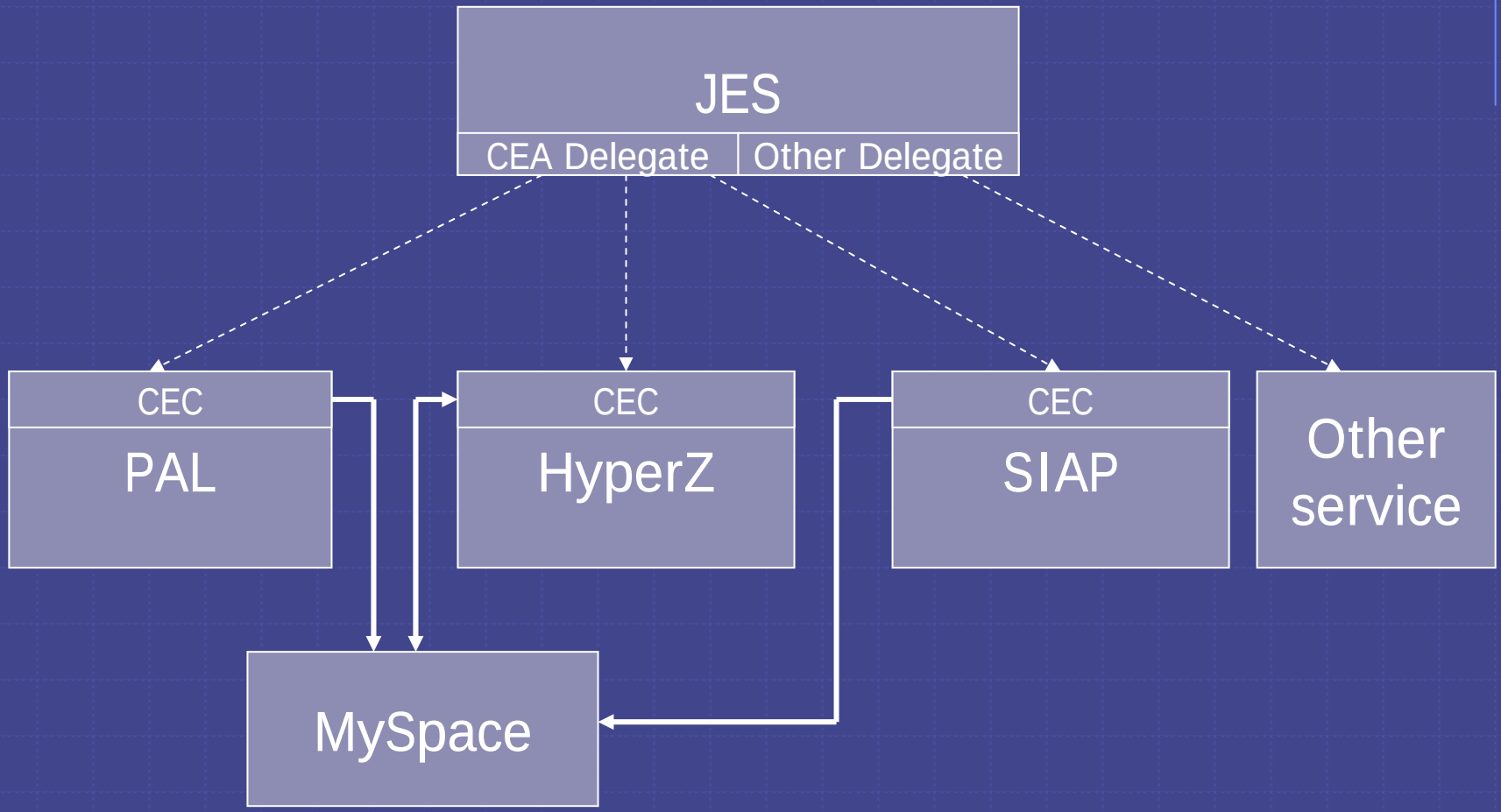
CEA language schema (1)



CEA language schema (2)



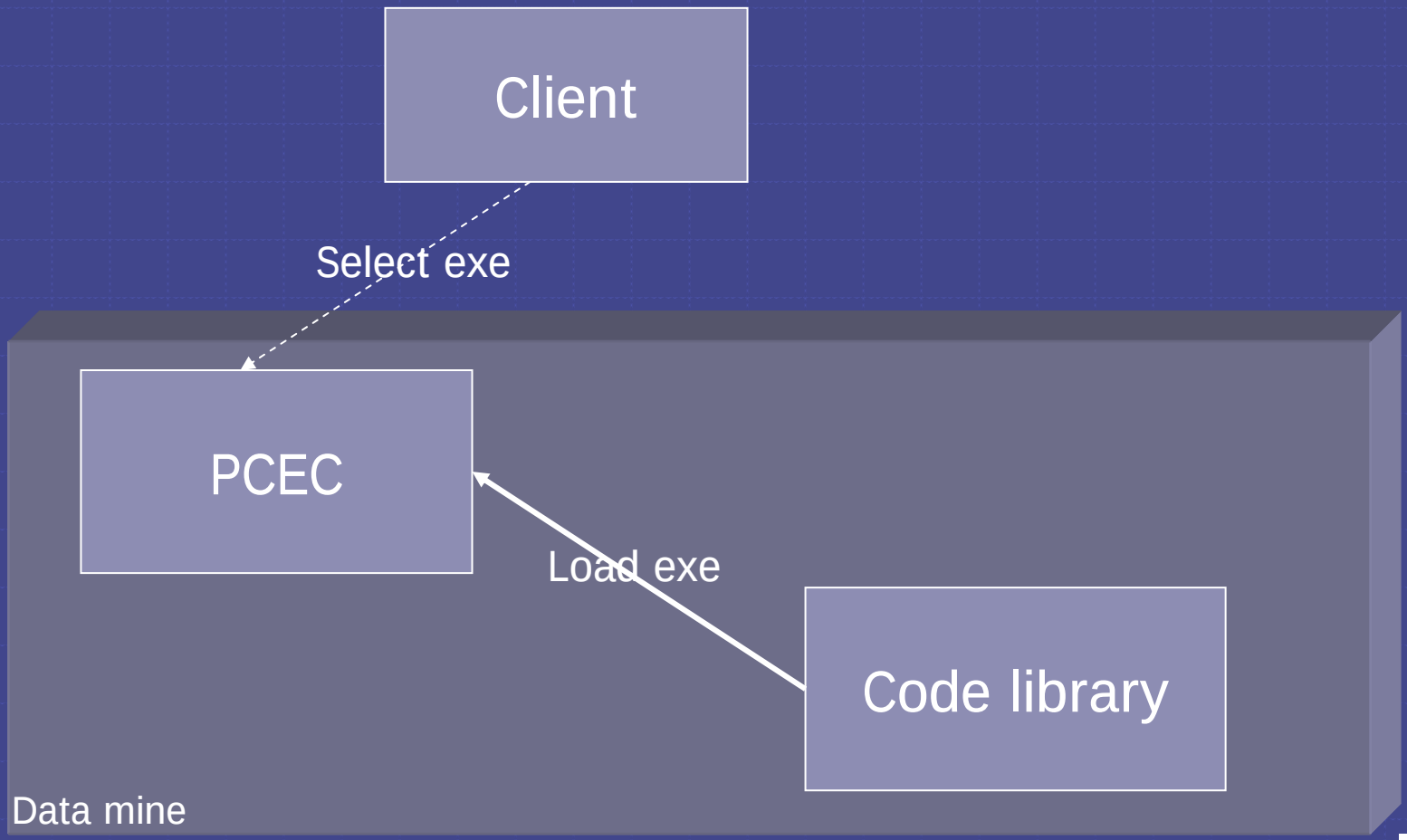
CEA facilities so far



Future evolution

- Add facilities to CEC:
  - Asynchronicity
  - Security
  - More parameter types
- New facility: field-programmable CEC (PCEC)

PCEC (1): local code libraries



PCEC (2): distributed code-library