OGSA-DAI Architecture

EPCC, University of Edinburgh

Amy Krause

a.krause@epcc.ed.ac.uk

International Summer School on Grid Computing - July 2003

Using OGSA-DAI

Release 3

Overview

- GridServices recap
- OGSA-DAI overview
- Scenarios
- Components:
  - Design
  - Configuration
- Component Interaction

OGSI Recap

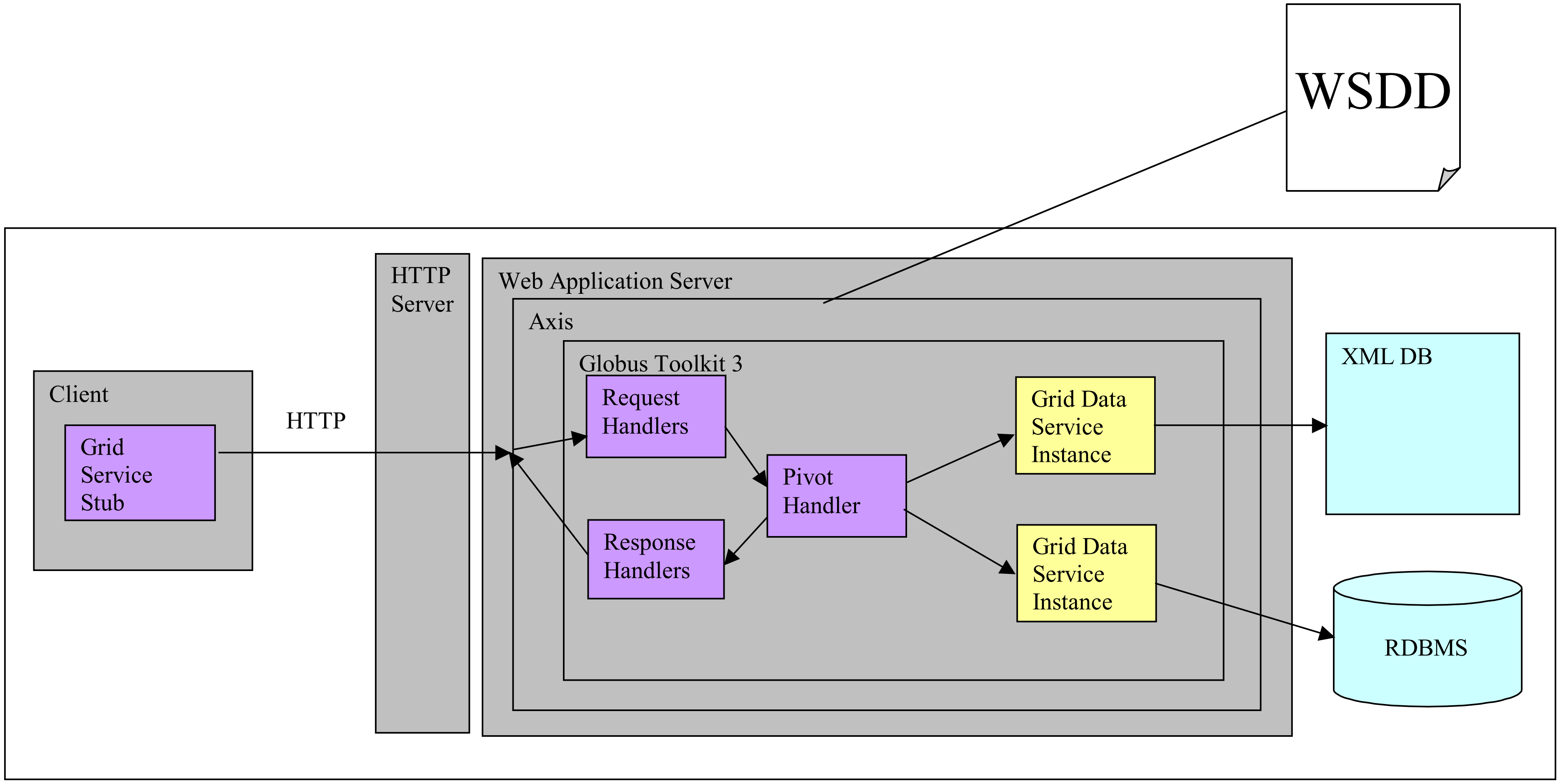
- Exploits existing web services properties
  - Interface abstraction (GWSDL resp. WSDL v1.2)
  - Protocol, language, hosting platform independence
- Enhancement to web services
  - State Management
  - Event Notification
  - Referenceable Handles
  - Lifecycle Management
  - Service Data Extension

See: The OGSI Specification (version 1.0 at GGF8)

Globus OGSI Implementation

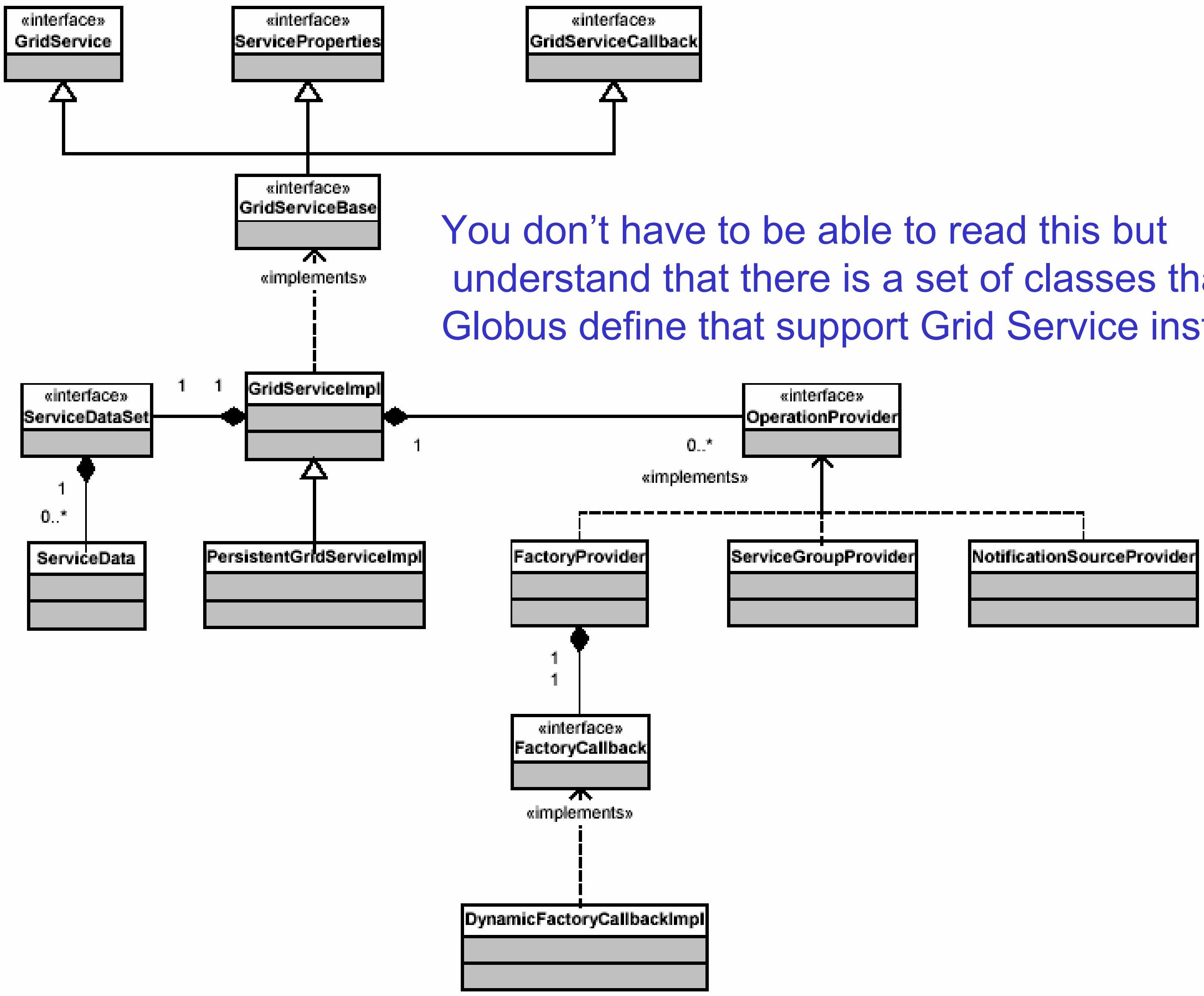
- Globus Toolkit 3 Release – June 03

The GT 3 Java Container



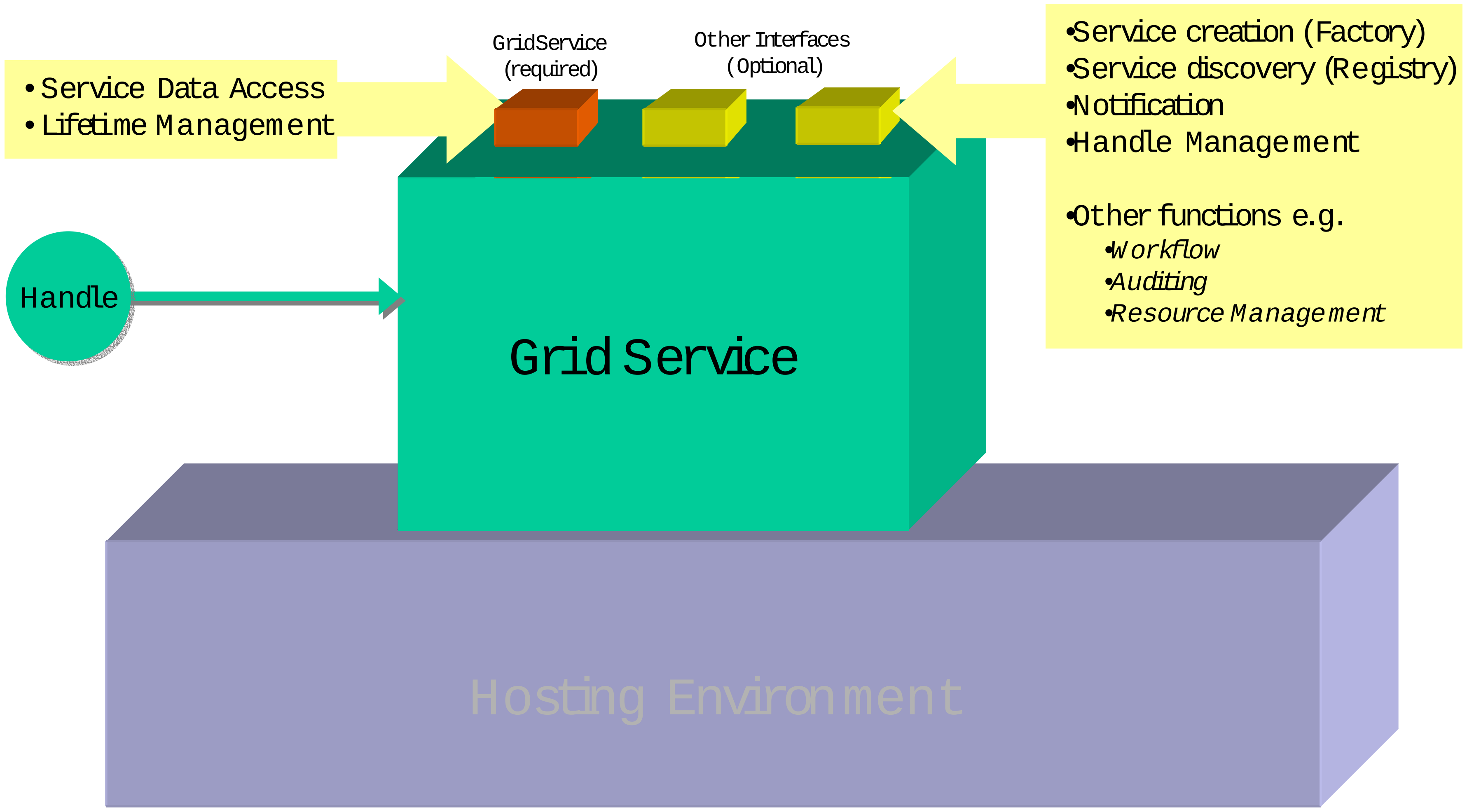
J2EE wrappers also included with JBoss as EJB container

Globus Server Side Model!?

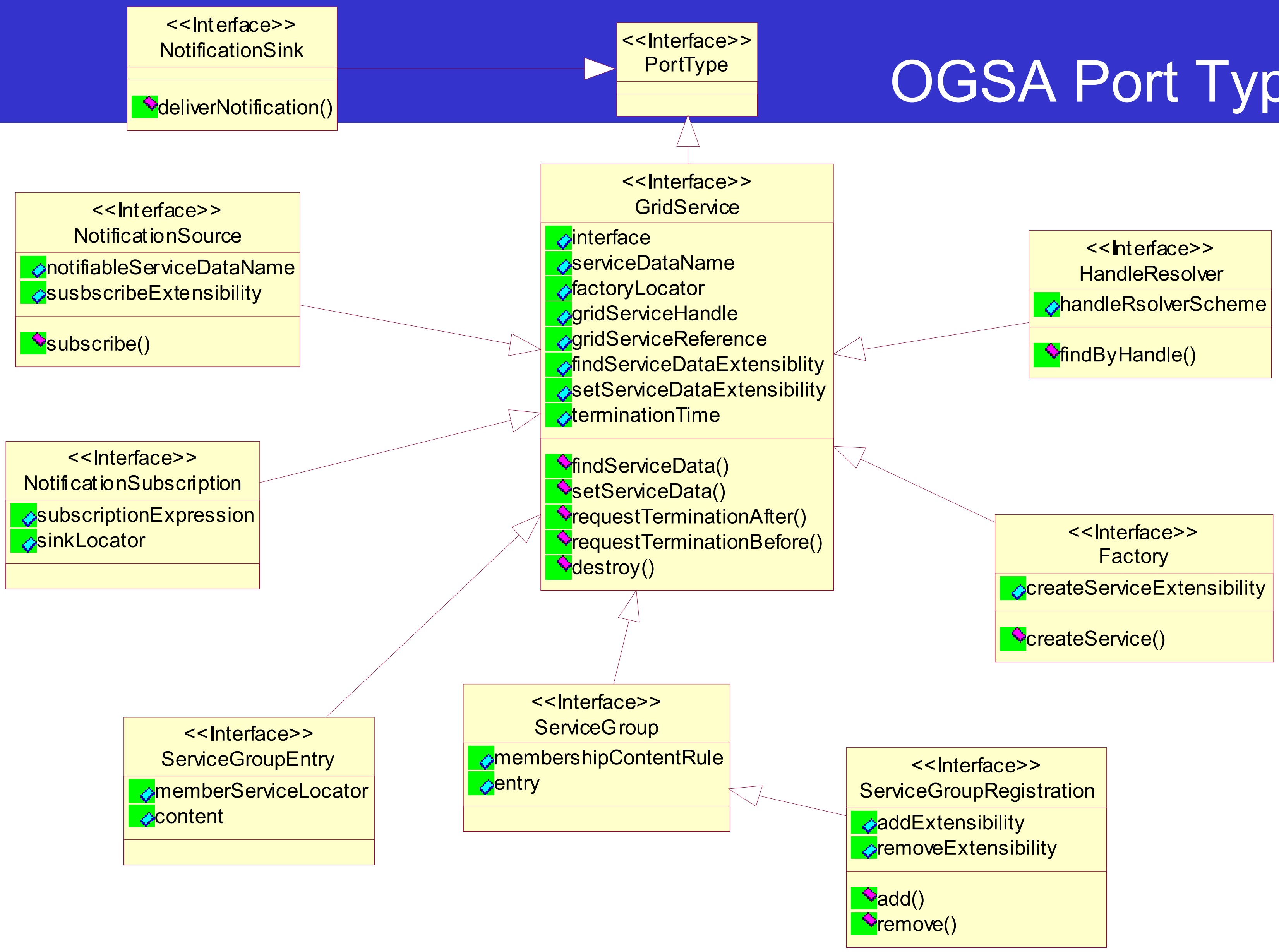

«interface» GridService
«interface» ServiceProperties
«interface» GridServiceCallback
«interface» GridServiceBase
«implements»
«interface» ServiceDataSet
1
1
GridServiceImpl
1
0..*
«interface» OperationProvider
«implements»
1
0..*
ServiceData
PersistentGridServiceImpl
FactoryProvider
ServiceGroupProvider
NotificationSourceProvider
1
1
«interface» FactoryCallback
«implements»
DynamicFactoryCallbackImpl

You don't have to be able to read this but understand that there is a set of classes that Globus define that support Grid Service instances

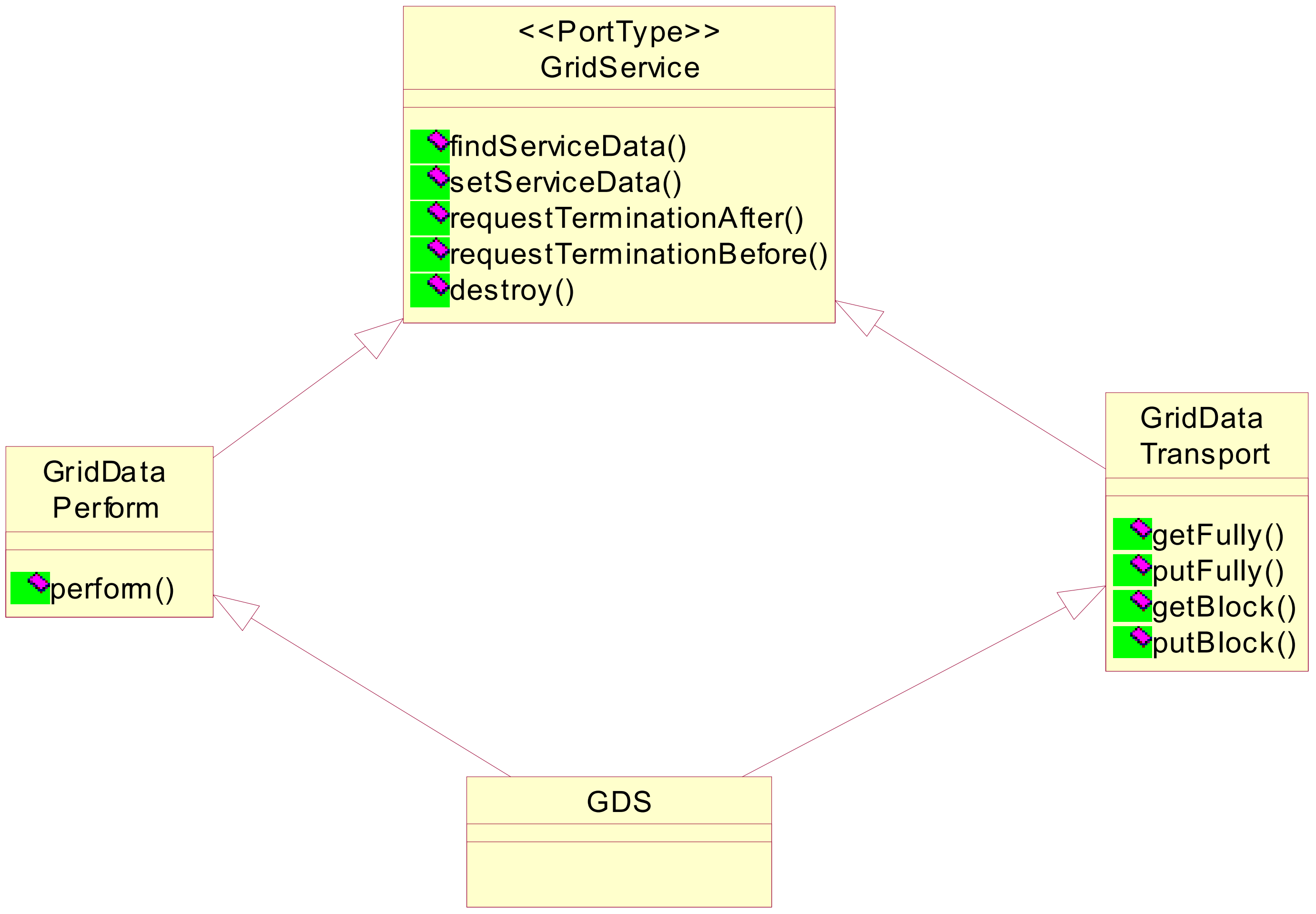
Anatomy Of A Grid Service



OGSA Port Types



OGSA-DAI Port Types



Java Services

- Service (Component) is implemented as a Java class
- Implements the portType interfaces and extends some base class

```
public class GDSService
               extends        GridServiceImpl
               implements     GDSPortType
```

- Here GT3.0 GridServiceImpl implements common GridService interface function
- Other common functions are reused through delegation
- This class is instantiated in order to create a service instance

The OGSA-DAI Project

- *OGSA - Data Access and Integration*
  - Jointly funded by the UK DTI eScience Programme and industry
- Provides data access and integration functions for computing Grids using the OGSI framework.
- Closely associated with GGF DAIS working group
- Project team members drawn from
  - Commercial organisations and
  - Non-commercial organisations
- Project runs until July 2003
  - Support DB2, Oracle, MySQL, Xindice

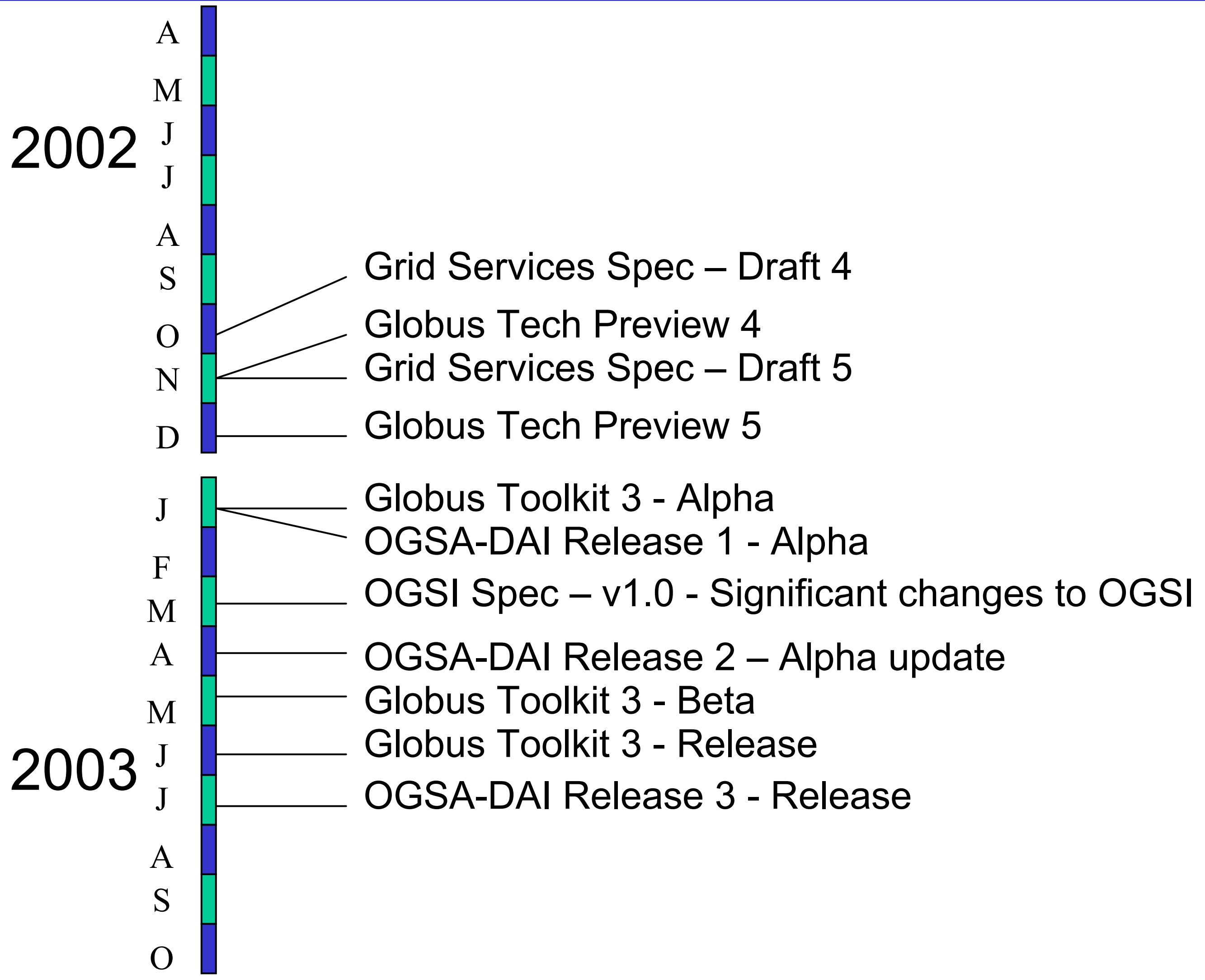
Phase 1

- Phase 1 – March to September 2002
  - GGF DAIS Workgroup Grid Database Spec
  - Architectural Framework
  - Release 0 - Software Prototypes
    - EPCC (XML Database) – OGSI compliant
    - IBM UK (Relational Database) – non-OGSI
  - Functional Scope for Phase 2

Phase 2

- Release 1 – Jan 2003
  - Basic infrastructure and services. Combine the efforts of Phase 1 and get the team going in one direction
- Release 2 – Apr 2003
  - More functionality and changes to match *Grid Service Specification* as was then (now OGSI)
- **Release 3 – July 2003**
  - Final release of Phase 2 to coincide with the full Globus GT3 release

Timeline



2002
A
M
J
J
A
S
O
N
D
Grid Services Spec – Draft 4
Globus Tech Preview 4
Grid Services Spec – Draft 5
Globus Tech Preview 5
J
F
M
A
M
J
J
A
S
O
2003
Globus Toolkit 3 - Alpha
OGSA-DAI Release 1 - Alpha
OGSI Spec – v1.0 - Significant changes to OGSI
OGSA-DAI Release 2 – Alpha update
Globus Toolkit 3 - Beta
Globus Toolkit 3 - Release
OGSA-DAI Release 3 - Release

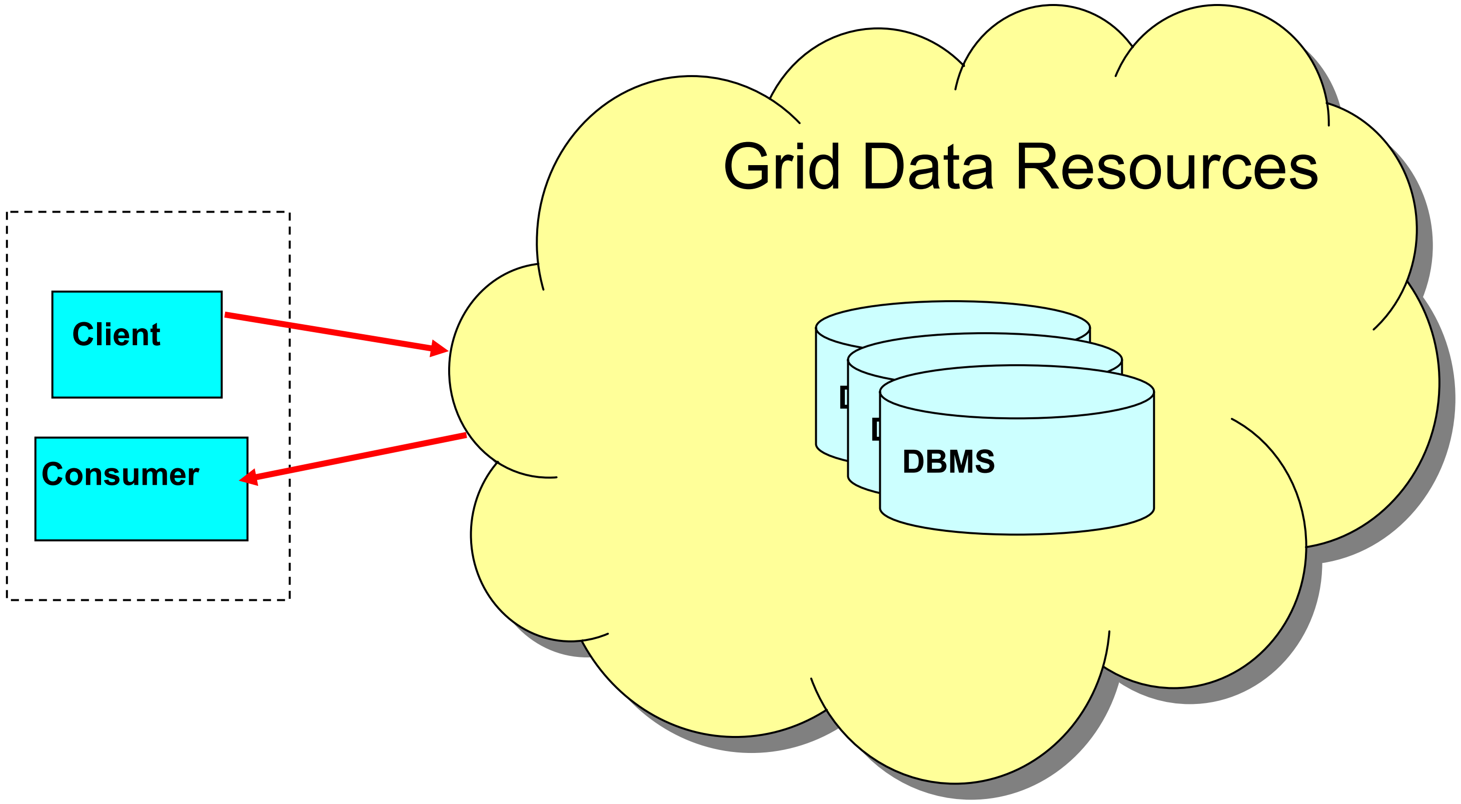
Grid Technology Repository

- Place for people to publish and discover work related to Grid Technologies
- International community-driven effort
- OGSA-DAI registered with the GTR
  - Visible UK contribution
  - Free publicity
- More information from:
  - http://gtr.globus.org

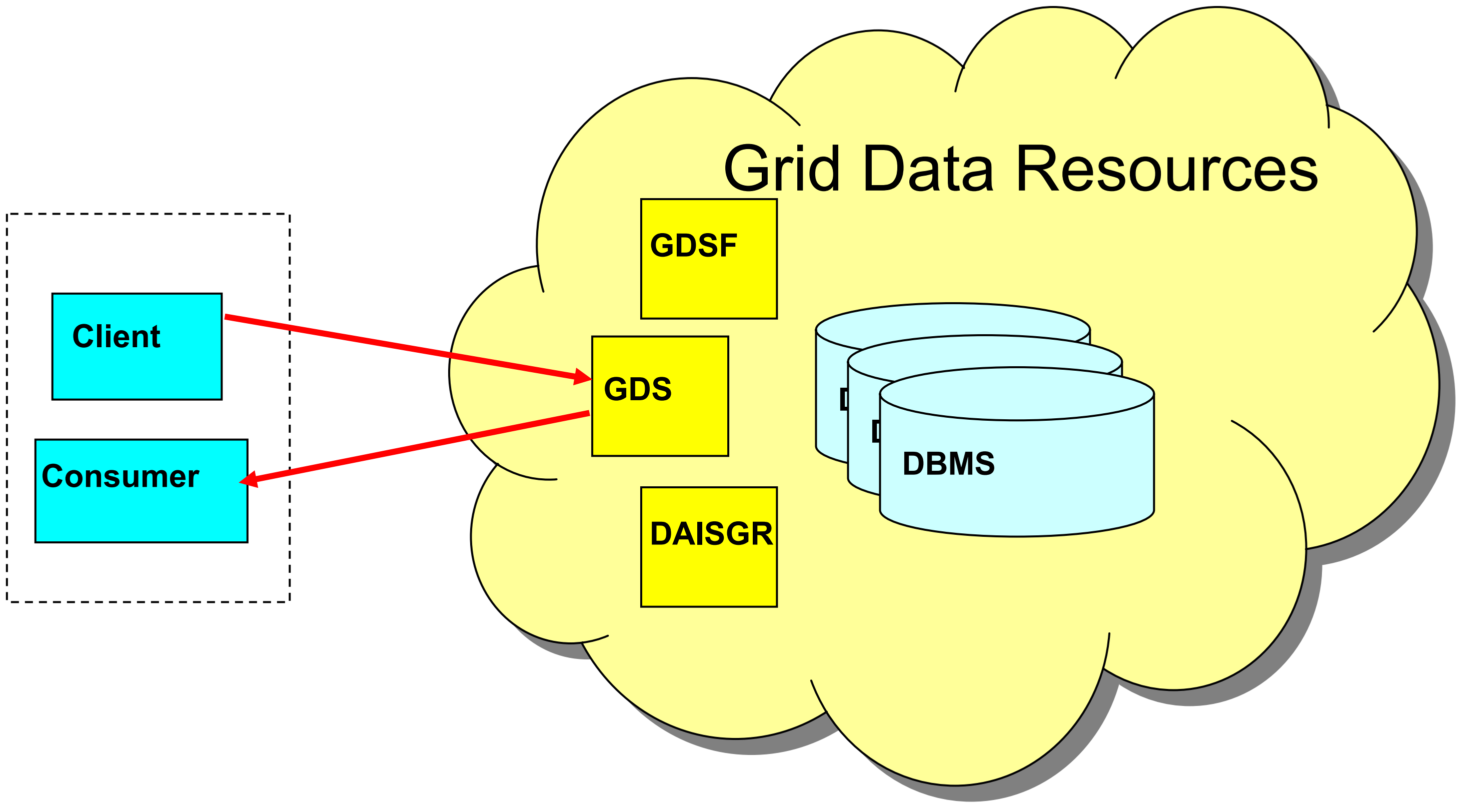
“Buy not Build”

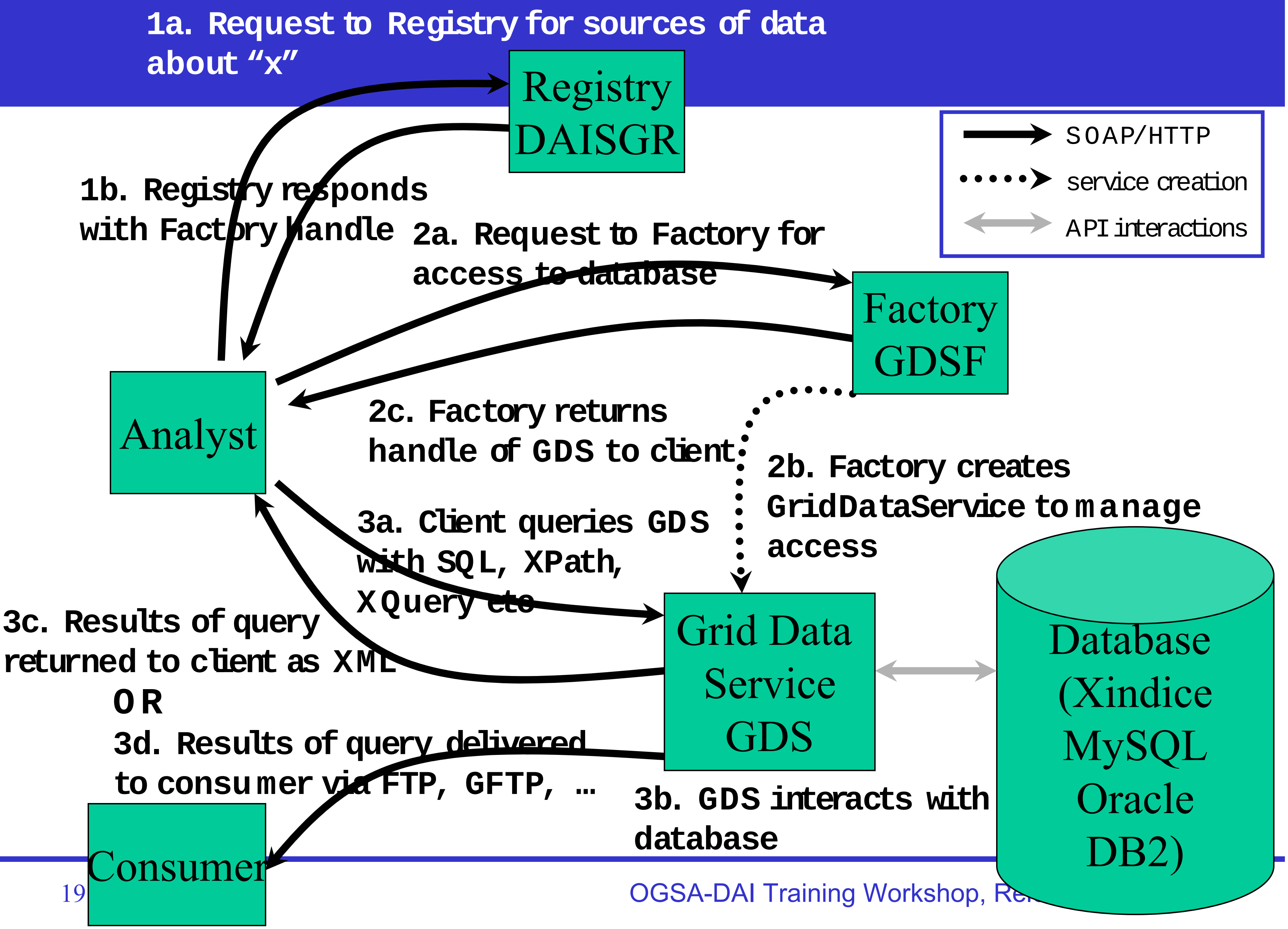
- OGSA/OGSI
- Query Language
- Data Format
- Data transport
- Data Description Schema
- Replication
- …

10000 Feet

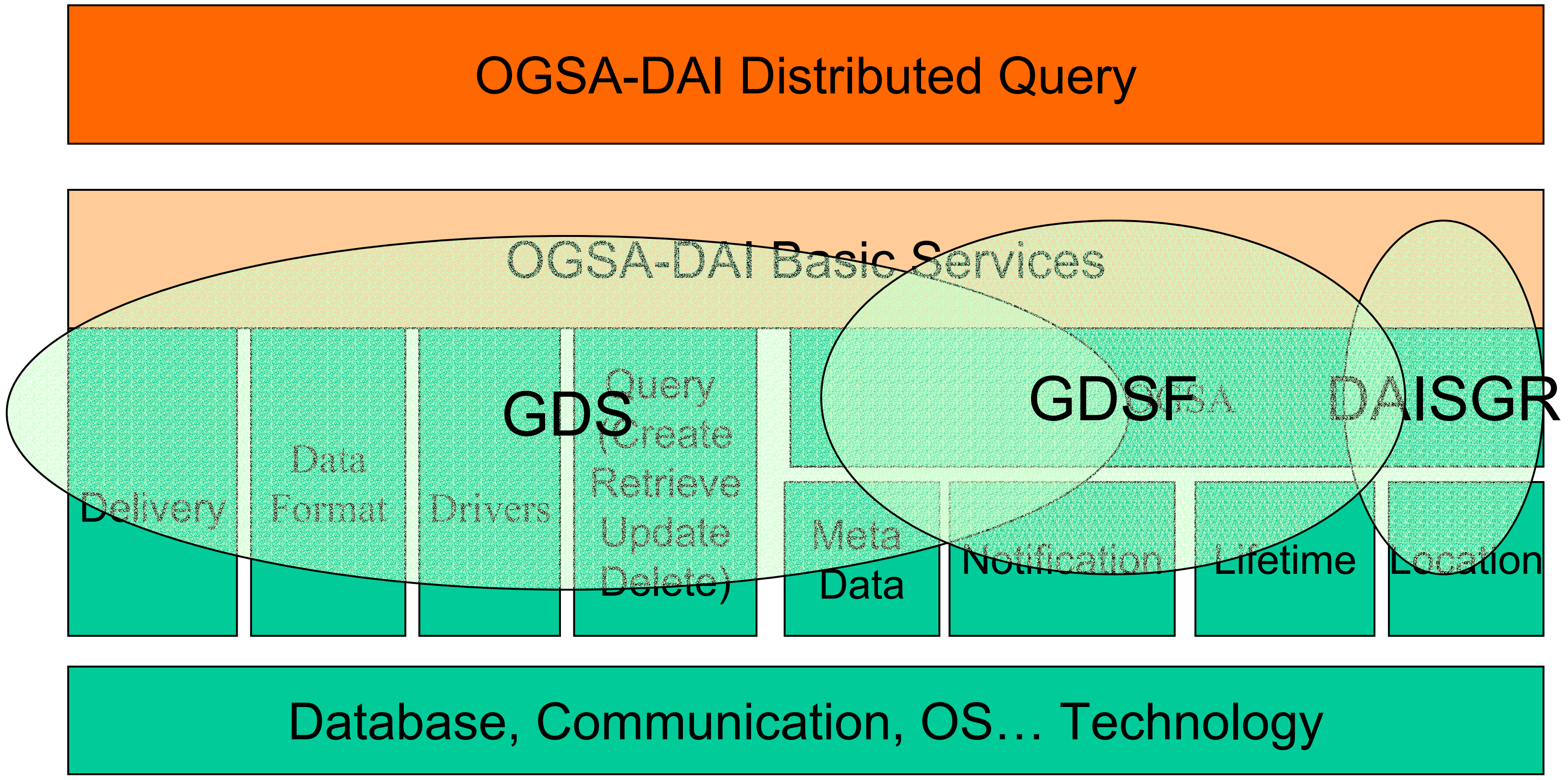


10000 Feet
With OGSA-DAI Services

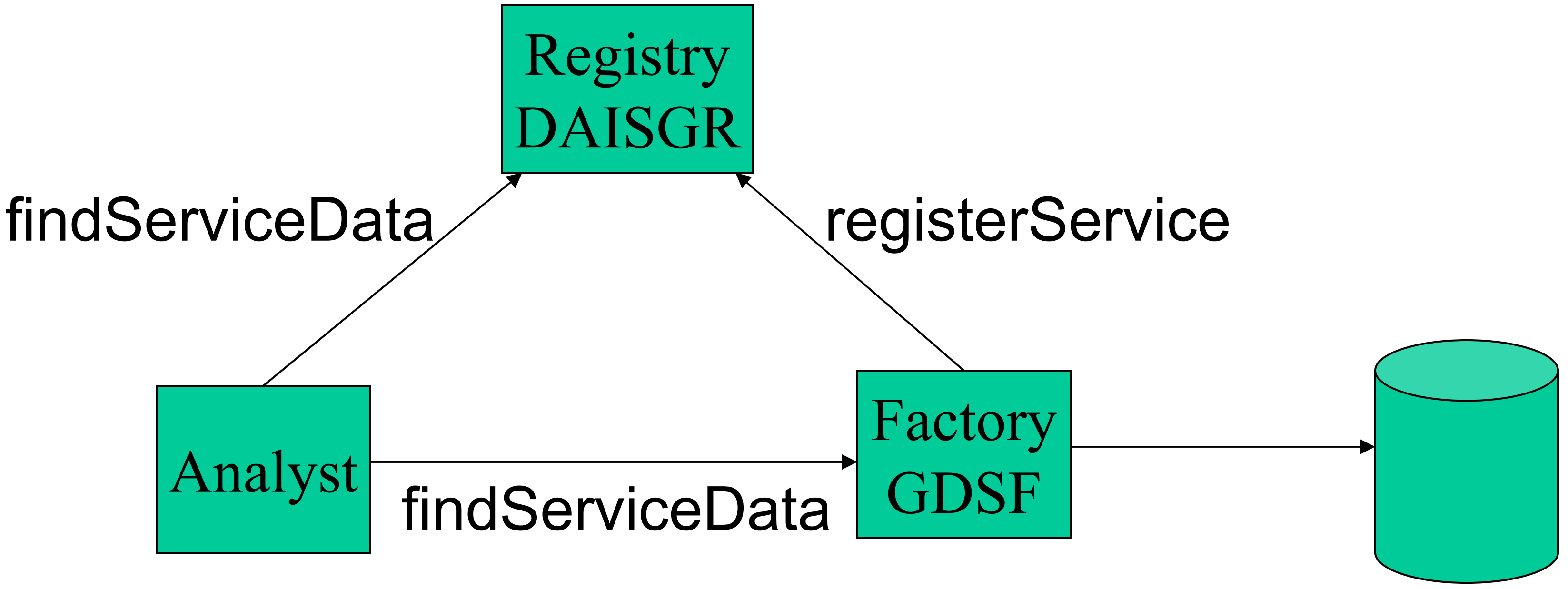



1a. Request to Registry for sources of data about "x"
Registry DAISGR
SOAP/HTTP
service creation
API interactions
1b. Registry responds with Factory Handle
2a. Request to Factory for access to database
Factory GDSF
Analyst
2c. Factory returns handle of GDS to client
2b. Factory creates GridDataService to manage access
3a. Client queries GDS with SQL, XPath, XQuery etc
3c. Results of query returned to client as XML
OR
3d. Results of query delivered to consumer via FTP, GFTP, …
Grid Data Service GDS
Database (Xindice MySQL Oracle DB2)
3b. GDS interacts with database
Consumer

OGSA-DAI Basic Services

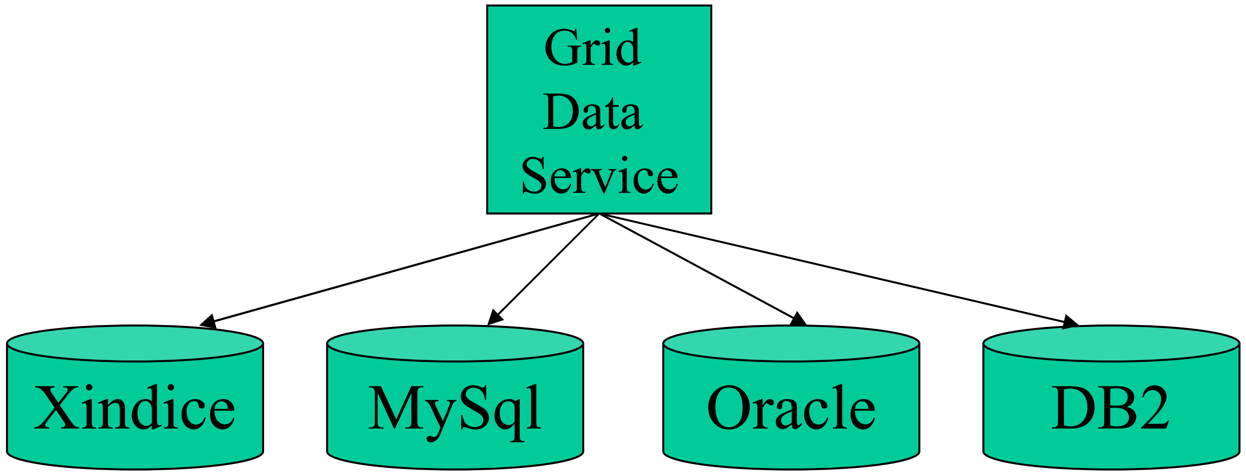


Location



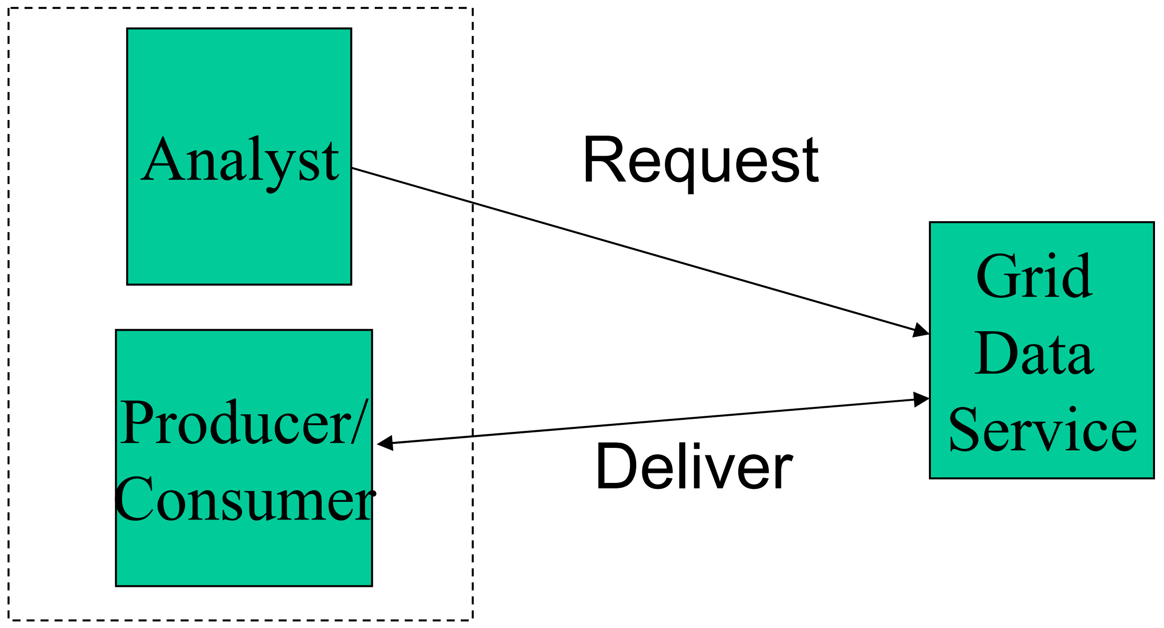
- Data resource publication through registry
- Data location hidden by factory
- Data resource meta data available through Service Data Elements

Heterogeneity



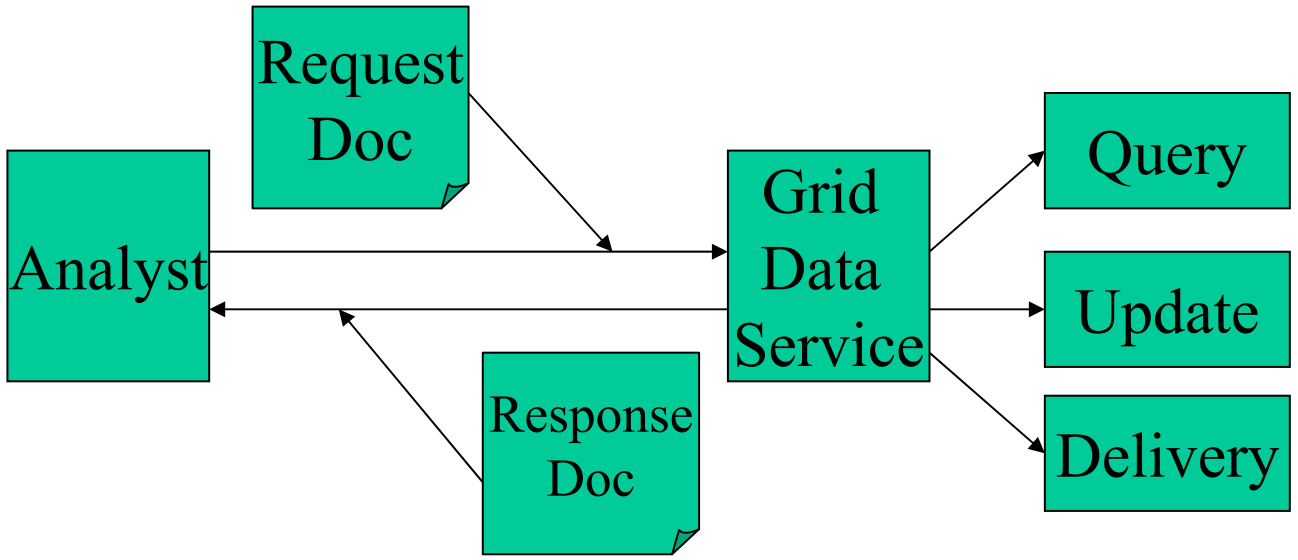
- Data source abstraction behind GDS instance
  - Plug in “data resource implementations” for different data source technologies
  - Does not mandate any particular query language or data format

Scale



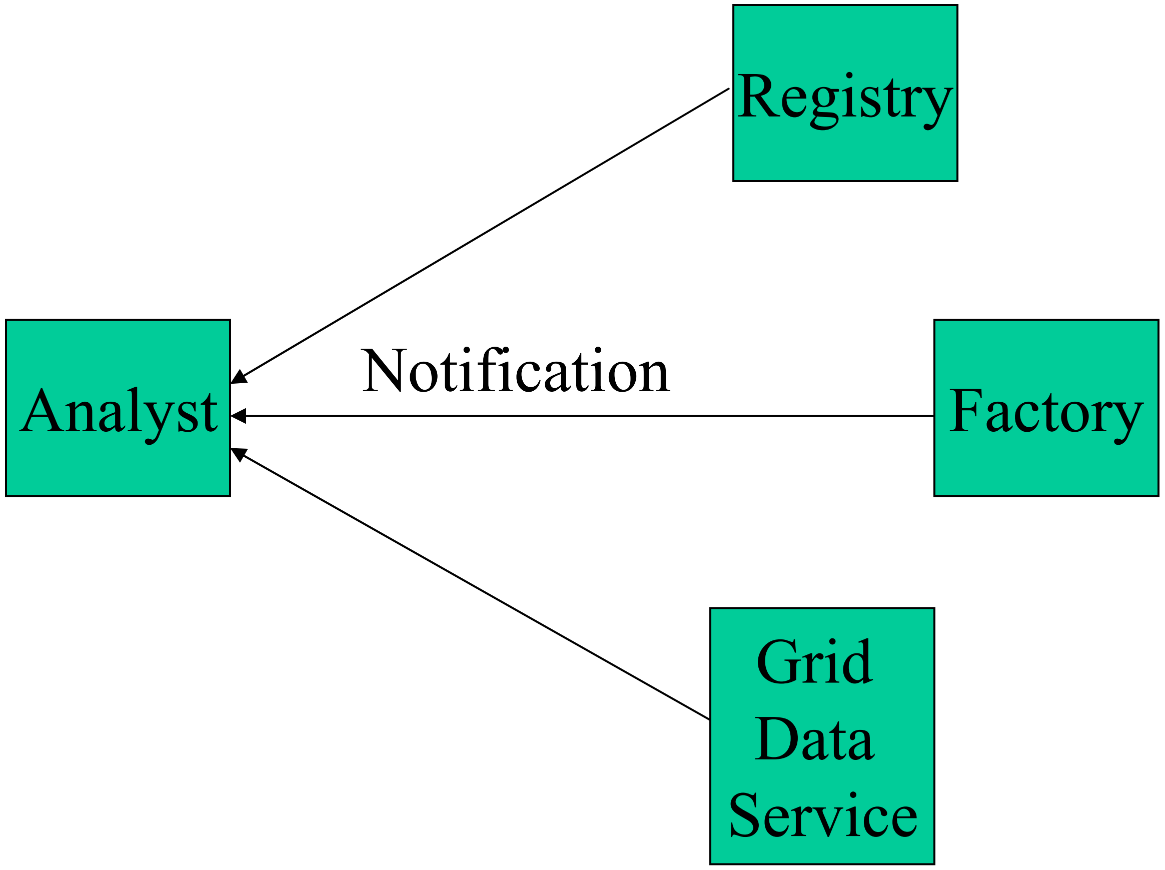
- Delivery configured as part of request
- Asynchronous delivery with varying modes/transports
  - "Zero copy deliver"
- OGSA-DAI will not specify transport mechanism but support existing

Flexibility



- Data source abstraction behind GDS instance
  - Document based interface
    - Document sharing, operation optimization
  - Combines statement with other, plugin, operations/activities
    - delivery, data transformation, data caching
  - Ongoing activity is represented in state of the service
    - running query, cached data, referenced data

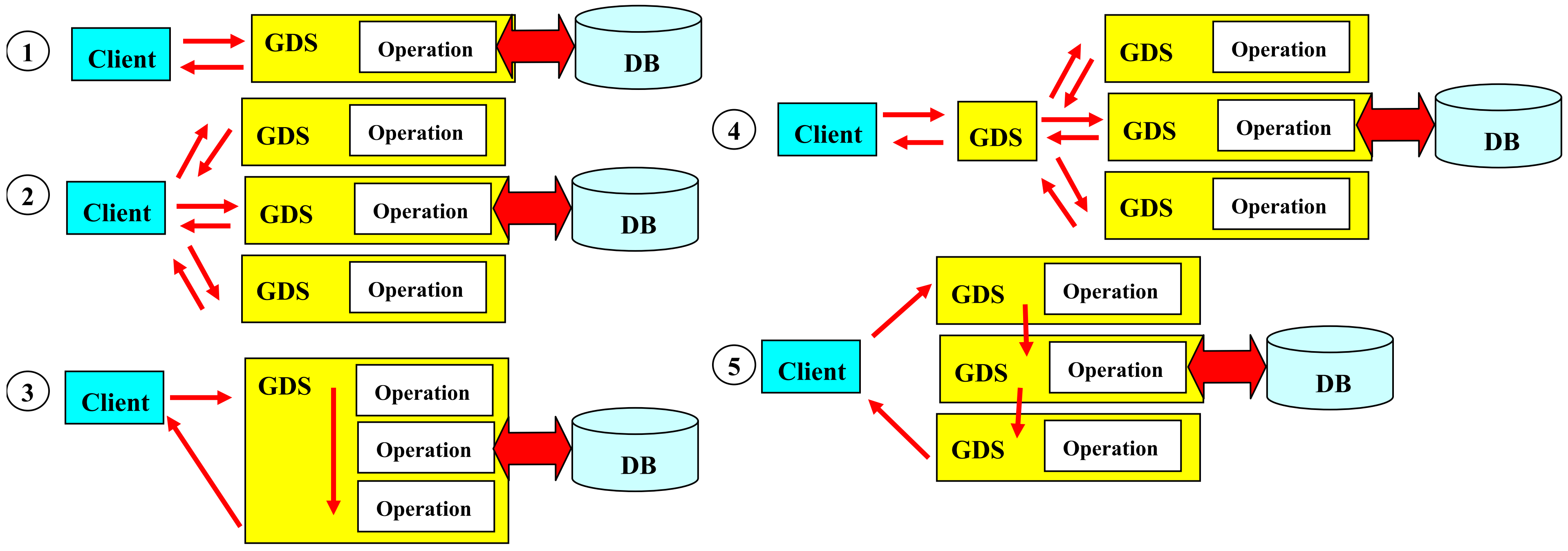
Dynamism



Management, Ownership, Accounting etc.

- We rely on OGSA/I for much common distributed computing function
- Any OGSA-DAI specific function will be compatible with OGSA/I approach
- Not much has been done to date

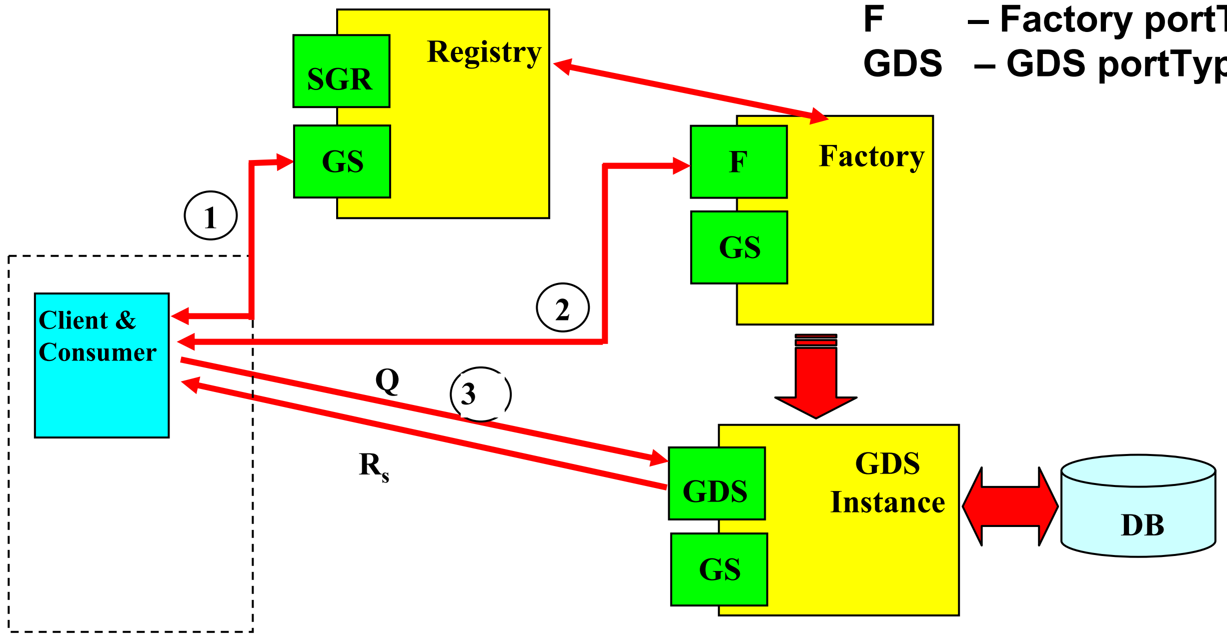
GDS Composition



Release 1

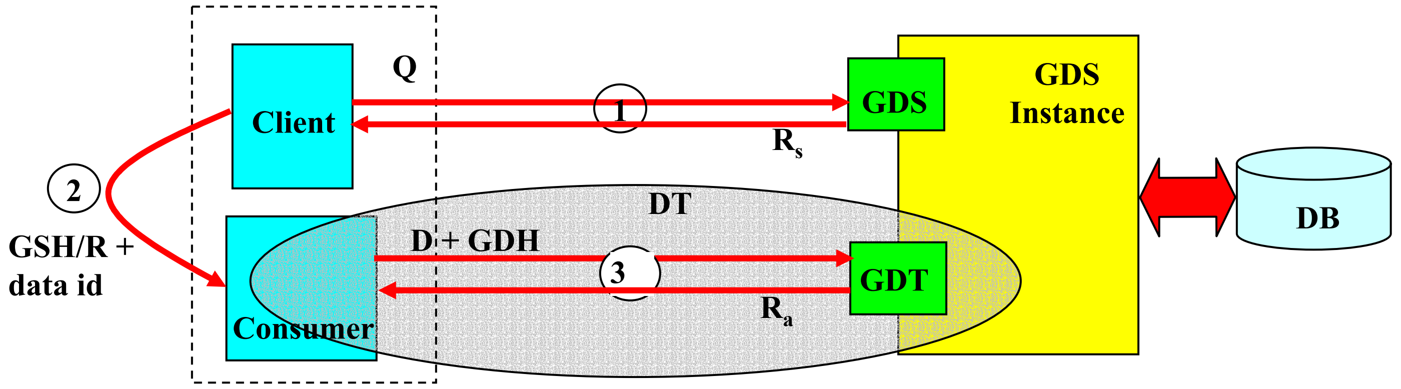
- Simple synchronous interaction with a data source using a GDS as a proxy.

SGR	– ServiceGroupRegistration portType
GS	– GridService portType
F	– Factory portType
GDS	– GDS portType

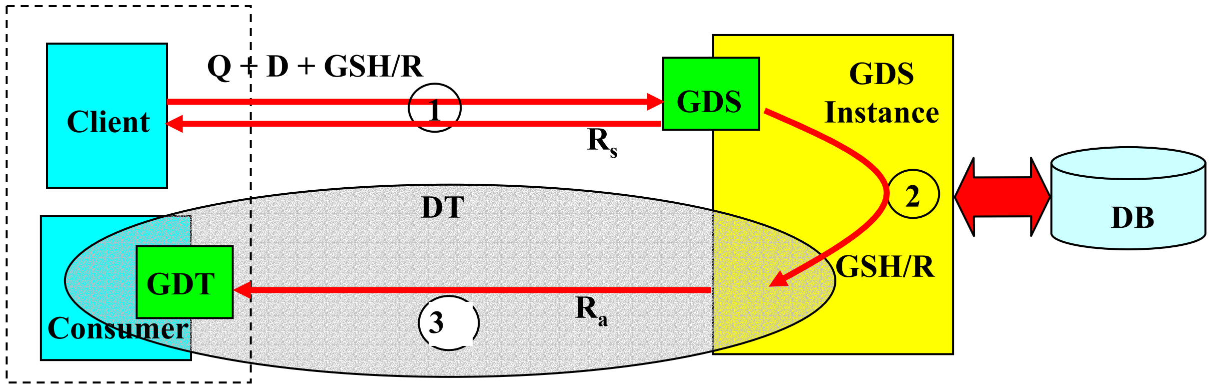


Release 3

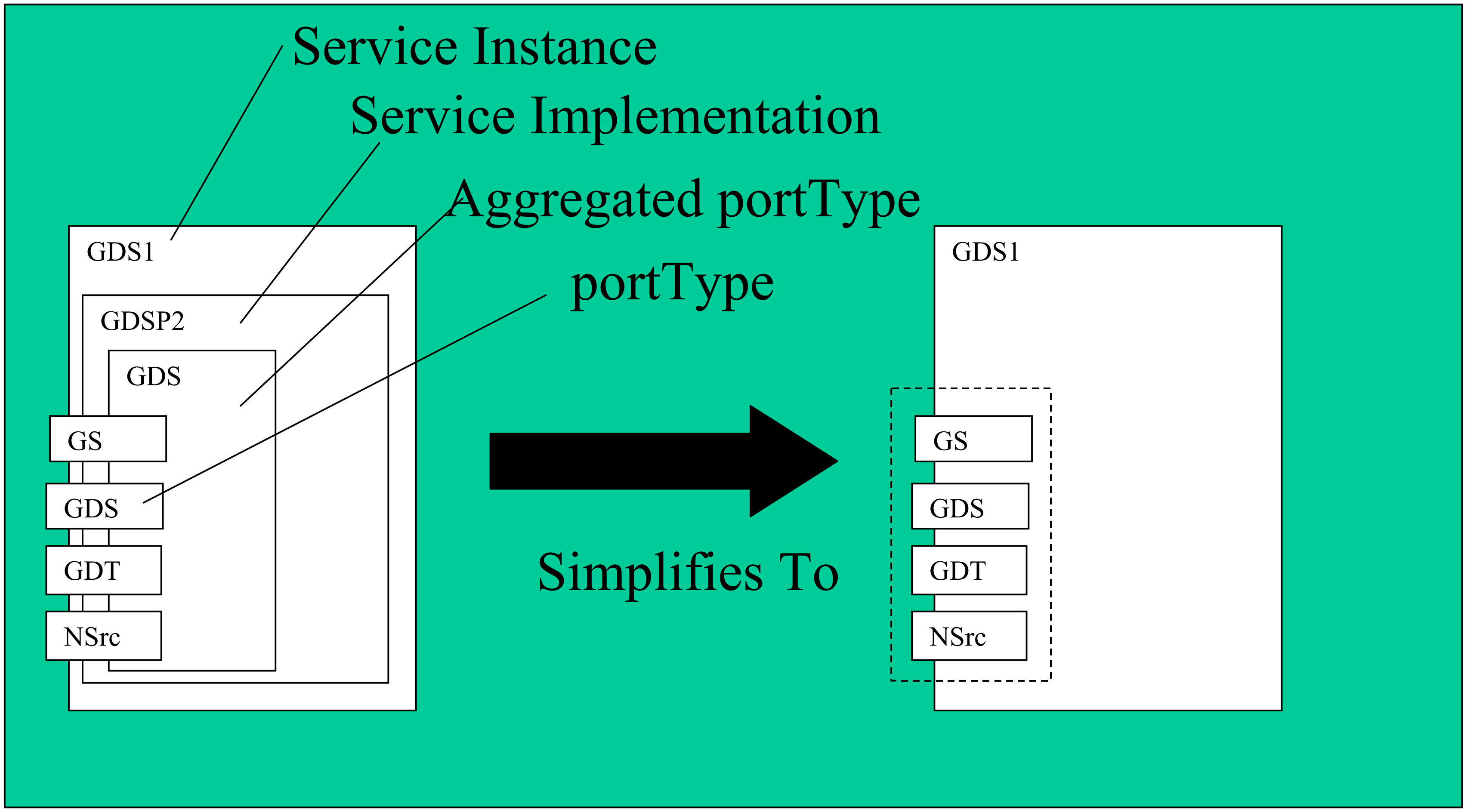
Asynchronous delivery – Pull



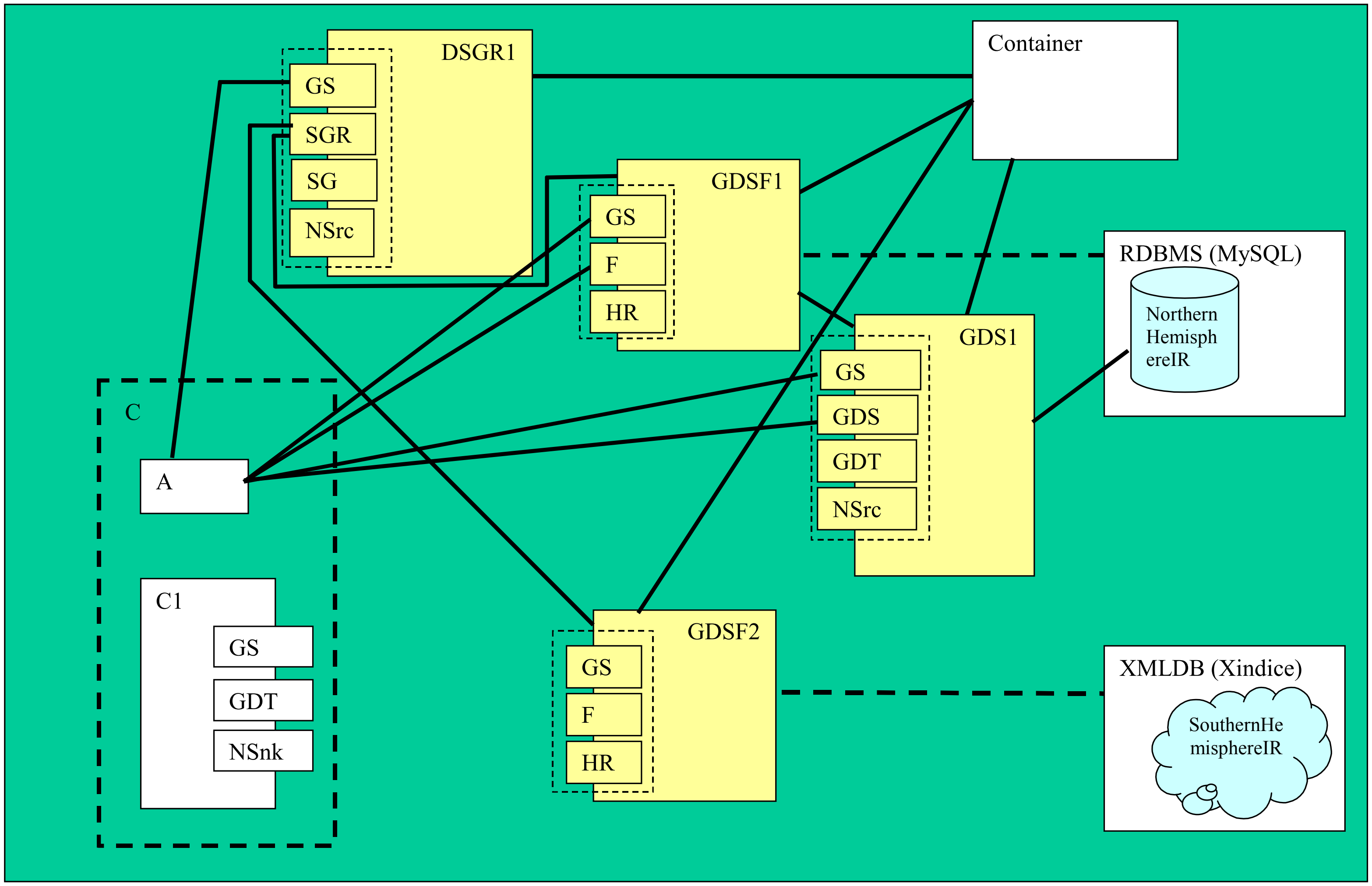
Asynchronous delivery – Push



Notation


Service Instance
Service Implementation
Aggregated portType
portType
GDS1
GDSP2
GDS
GS
GDS
GDT
NSrc
Simplifies To
GDS1
GS
GDS
GDT
NSrc

Overview – Release 3 (R3)



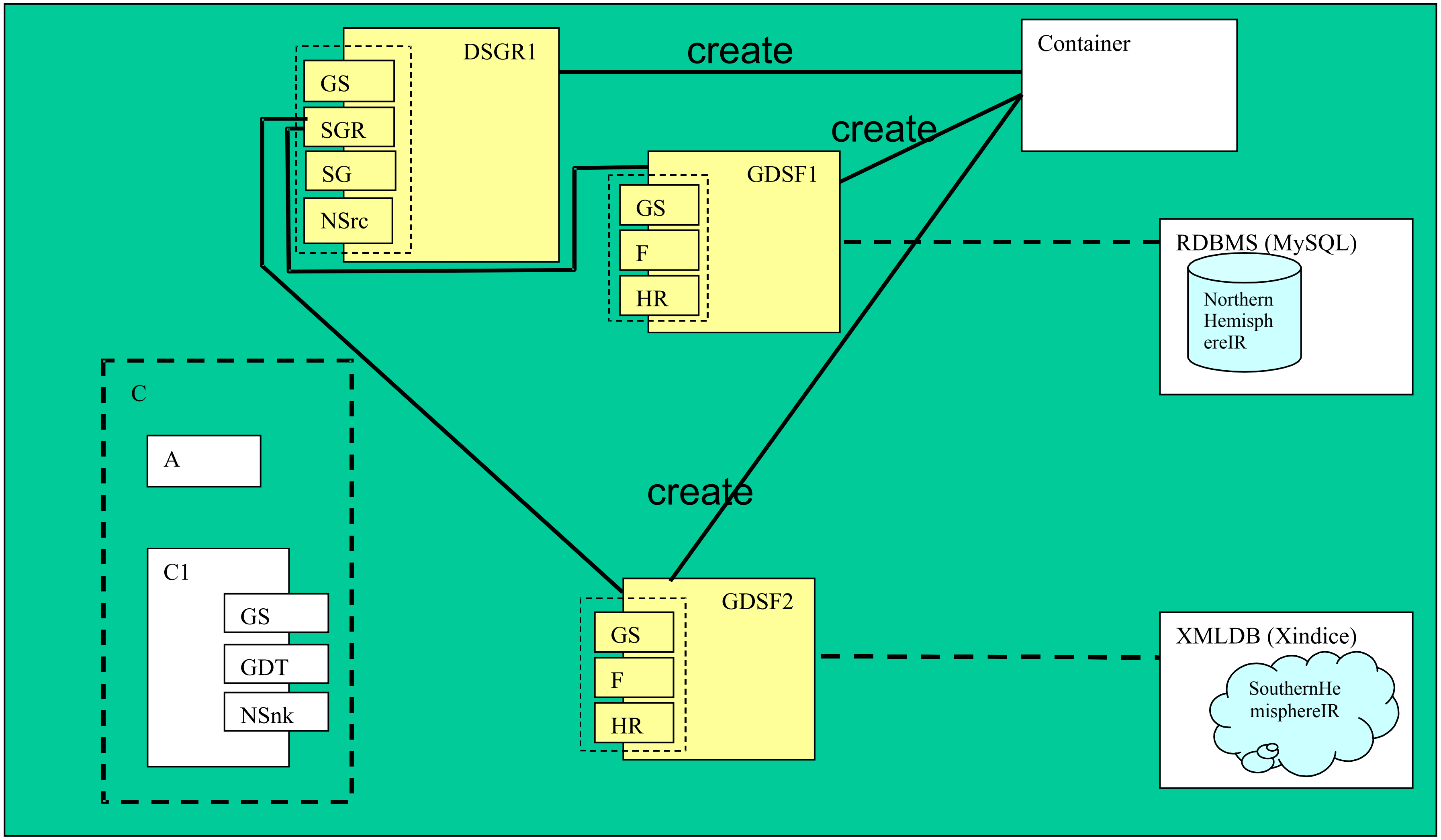
Scenario 1 (synchronous delivery)

- An analyst wants to perform a SQL query across a dataset with a known name and schema
  - Container starts
  - Analyst Starts
  - Analyst identifies factory that supports required statement type
  - Analyst uses factory to create GDS instance and obtains GSH
  - Analyst maps GSH to GSR using factory
  - Analyst formulates a GDS perform document containing the query
  - Analyst passes GDS perform document to GDS instance
  - GDS instance returns data in response
  - Analyst removes GDS instance

Scenario 2 (asynchronous delivery)

- An analyst wants to perform an XPath query across a dataset with a known name and schema
  - Container starts
  - Analyst Starts
  - Analyst identifies factory that supports required statement type
  - Analyst uses factory to create GDS instance and obtains GSH
  - Analyst maps GSH to GSR using factory
  - Analyst formulates a GDS perform document containing the query and the URL of the consumer
  - Analyst passes GDS perform document to GDS instance
  - GDS instance returns report to analyst
  - GDS instance delivers data to specified consumer
  - Analyst removes GDS instance

Container Start



DAIServiceGroupRegistry

- Allows OGSA-DAI services to:
  - Make clients aware of their existence.
  - Make clients aware of their capabilities, services or the data resources they manage.
  - Be shared amongst multiple clients.
- Allows clients to:
  - Search for DAI services meeting their requirements.

PortTypes

- Most-derived portType:
  - DAIServiceGroupRegistry.
- Aggregates OGSI portTypes:
  - GridService:
    - Query registered services via **findServiceData**.
  - NotificationSource:
    - Subscribe to changes in DAISGR state via **subscribe**.
  - ServiceGroup:
    - Group together DAI services.
  - ServiceGroupRegistration:
    - Add and remove DAI services to and from the DAISGR via **add** and **remove**.

GridDataServiceFactory

- Exposes a data resource to clients.
- Allows clients to request creation of Grid Data Services which can be used to interact with the data resource.

GridDataServiceFactory PortTypes

- Most-derived portType:
  - GridDataServiceFactory.
- Aggregates OGSI portTypes:
  - GridService:
    - Query the data resource exposed by the GDSF via **findServiceData**.
  - Factory:
    - Create a GDS to allow interaction with a data resource via **createService**.
  - NotificationSource:
    - Subscribe to changes in DAISGR state via **subscribe**.

GridDataService PortTypes

- Most-derived portType:
  - GDSPortType – GridDataService
- Aggregates OGSI and OGSA-DAI portTypes:
  - GridService:
    - Query the data resource exposed by the GDSF via **findServiceData**.
  - GridDataPerform:
    - Interact with the data resource represented by the GDS via **perform**.
  - GridDataTransport
    - Give data to or receive data from the GDS data either in one complete chunk or in separate sub-chunks via **putFully**, **putBlock**, **getFully** and **getBlock**.

Behind the scenes: Data Resources

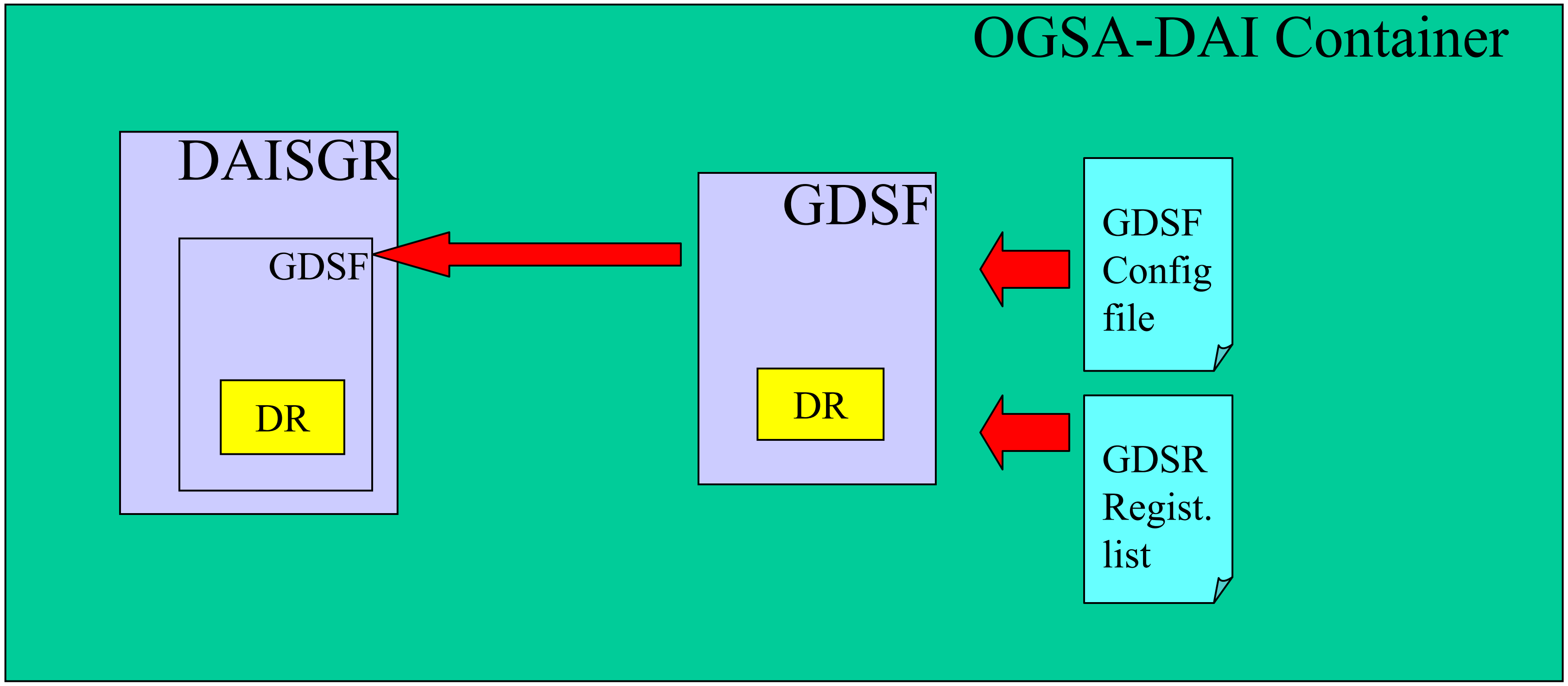
- Data Resources in OGSA-DAI represent a data source/sink
- Data Resources are typified by:
  - Way of communicating with the data resource
  - Location, i.e. properties about the container managing access to the data source/sink and information about its capabilities
  - The actual data source/sink
  - The resource, an instantiation/view/sample obtained from the data source/sink

Data Resources in OGSA-DAI

- An OGSA-DAI Factory is configured with exactly one data resource
  - Done in the factory configuration file
  - Data resource confined to a static named object defined in the Factory configuration file
  - In the future hope to make this more dynamic
- A GDS created by a factory
  - Can only be associated with the data resource known to the factory
  - Can only be associated with one data resource

WSDD Container Config

- Creates persistent registry
- Creates persistent factory
  - Defines configuration files to read in



WSDD Container Config

```
<service name="ogsadai/GridDataServiceFactory" provider="Handler" style="wrapped"
use="literal">
 <parameter name="ogsadai.gdsf.config.xml.file" value="dataResourceConfigRel.xml"/>
 <parameter name="ogsadai.gdsf.registrations.xml.file"
value="registrationList.xml"/>
 <parameter name="name" value="Grid Data Service Factory"/>
 <parameter name="operationProviders"
value="org.globus.ogsa.impl.ogsi.FactoryProvider"/>
 <parameter name="persistent" value="true"/>
 <parameter name="instance-schemaPath" value="schema/ogsadai/gds/gds_service.wsdl"/>
 <parameter name="instance-baseClassName"
value="uk.org.ogsadai.service.gds.GridDataService"/>
 <parameter name="baseClassName"
value="uk.org.ogsadai.service.gdsf.GridDataServiceFactory"/>
 <parameter name="schemaPath"
value="schema/ogsadai/gdsf/grid_data_service_factory_service.wsdl"/>
 <parameter name="handlerClass" value="org.globus.ogsa.handlers.RPCURIProvider"/>
 <parameter name="instance-name" value="Grid Data Service"/>
 <parameter name="className"
value="uk.org.ogsadai.wsdl.gdsf.GridDataServiceFactoryPortType"/>
 <parameter name="allowedMethods" value="*"/>
 <parameter name="factoryCallback"
value="uk.org.ogsadai.service.gdsf.GridDataServiceFactoryCallback"/>
 <parameter name="activateOnStartup" value="true"/>
</service>
```

Factory Configuration XML

- Defines components that constitute a data resource
  - **DataResourceManager:** contains DBMS specifics, such as driver class and physical location, and can implement connection pooling
  - **RoleMaps:** maps grid credentials to database roles
  - **DataResourceMetadata:** metadata such as product information and relational or XMLDB specific information
  - **ActivityMaps:** activities i.e. operations supported by the data resource; each activity is mapped to its implementing class and a schema

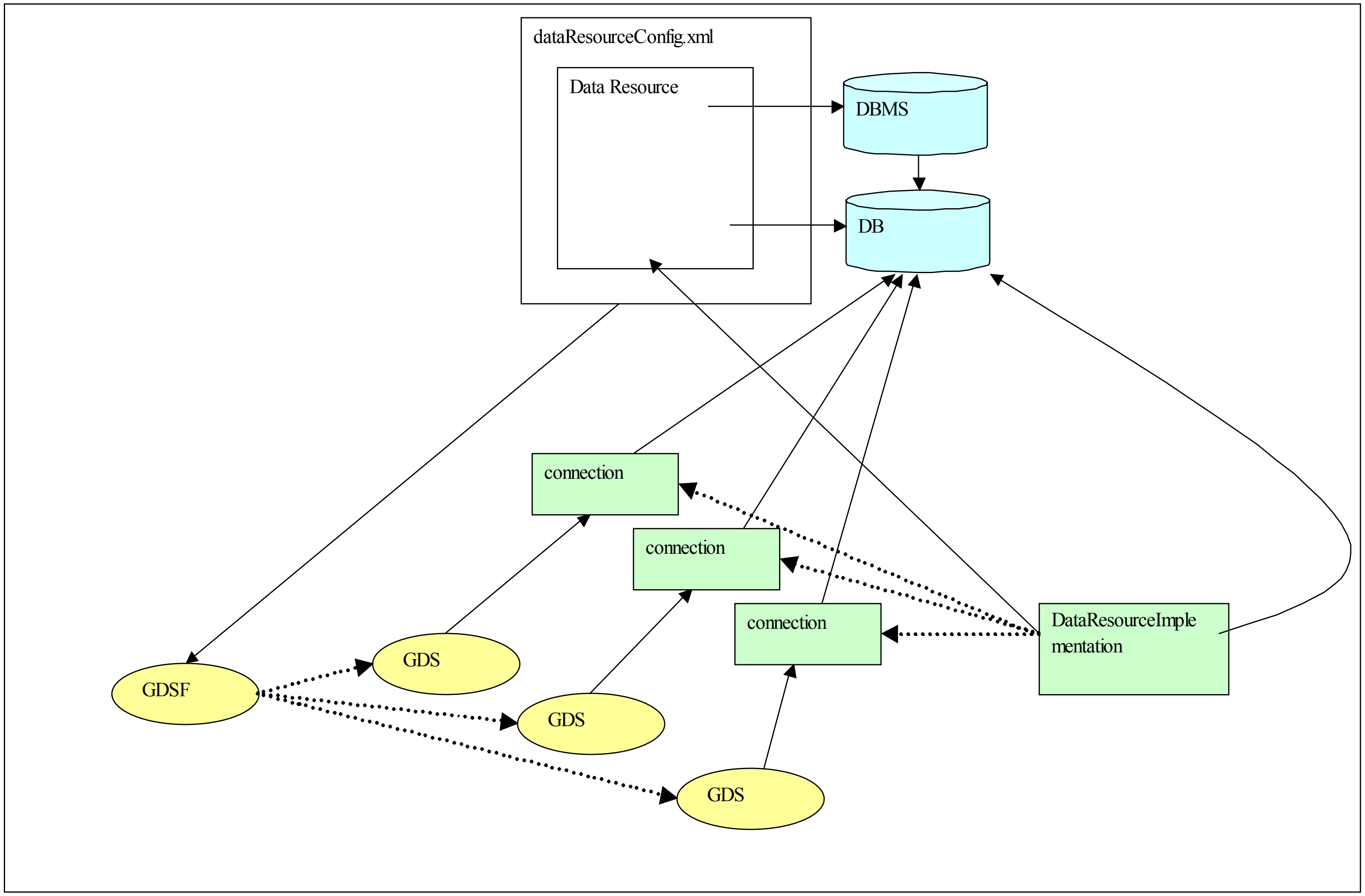
Factory Configuration XML Skeleton

```
<dataResourceConfig
      xmlns="http://ogsadai.org.uk/namespaces/2003/07/gdsf/config">

   <documentation> A sample config file. </documentation>

   <activityMap name="sqlQueryStatement> . . .
   </activityMap>

   <dataResourceMetadata>
      . . .
   </dataResourceMetadata>

   <roleMap name="Name" . . . />

   <driverManager . . .>
      <driver> . . .
      </driver>
   </driverManager>

</dataResourceConfig>
```

Driver Manager

- *DriverManager* objects encapsulate the data resource, e.g.
  - Provide connection pooling to databases
  - Allows a single collection of objects to be shared across any number of GDS instances
  - GDS connection capabilities to generate dynamic information capabilities, e.g. obtain the database schema
- GDSF constructs and populates these objects
- The *DriverManager* mapping element relates the data resource defined in the GDSF configuration file to a Java implementation class
- Currently have generic classes for
  - JDBC databases
  - XML:DB databases (i.e. Xindice)

Data Resource Implementation Mapping



Factory Configuration: DriverManager

```
<driverManager
      driverManagerImplementation="uk.org.ogsadai.porttype.gds.
                    dataresource.SimpleJDBCDataResourceImplementation">
  <driver>
    <driverImplementation>org.gjt.mm.mysql.Driver</driverImplementation>
    <driverURI>
        jdbc:mysql://localhost:3306/ogsadai
    </driverURI>
  </driver>
</driverManager>
```

Factory Configuration: DataResourceMetadata

```
<dataResourceMetadata>

  <productInfo>
    <!--  This element and its contents are optional. -->
    <productName>MySQL</productName>
    <productVersion>4</productVersion>
    <vendorName>MySQL</vendorName>
  </productInfo>

  <relationalMetaData>
    <databaseSchema
           callback="uk.org.ogsadai.porttype.gds.
                     dataresource.SimpleJDBCMetaDataExtractor" />
  </relationalMetaData>

  <!-- User can define own metadata -->

</dataResourceMetadata>
```

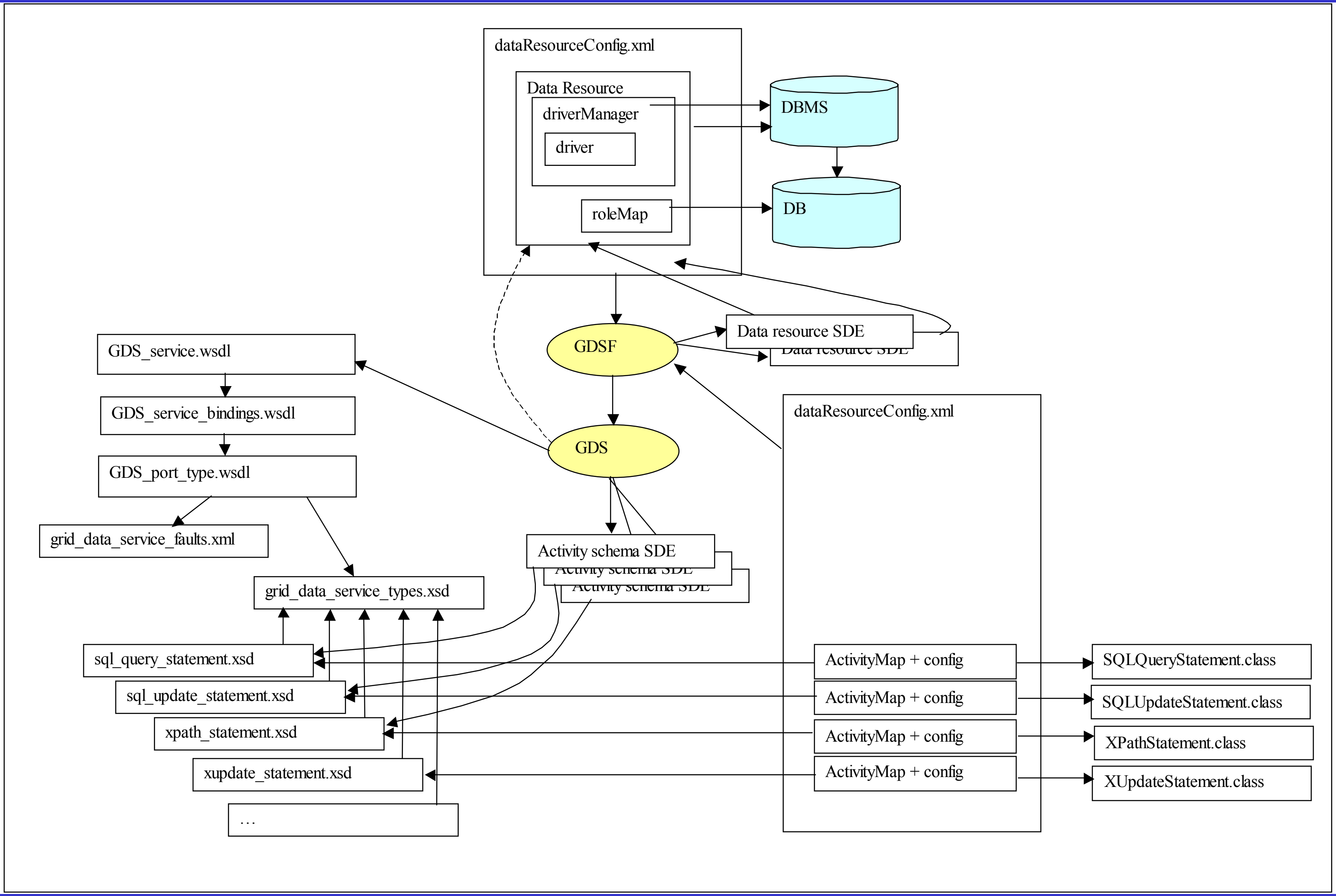
Activities

- Activities are tasks/operations that can be performed by a GDS on a data resource
  - Clearly data resources can support subset of activities, e.g. cannot run an SQL query on a Xindice database
  - The Factory identifies the activities supported by the data resource at configuration time

Activity Mapping

- The Activity Map file relates each named activity to
  - a Java implementation class
  - XML Schema that corresponds to activity
- Maps activities to data resources
  - Unless you are writing your own activity you should not need to modify this file

Activity Mapping II



Activity Map Example

```
<activityMap name="sqlUpdateStatement"
   implementation="uk.org.ogsadai. … .SQLUpdateStatementActivity"
   schemaFileName="http://localhost:8080/…/sql_update_statement.xsd"/>
<activityMap name="sqlStoredProcedure"
   implementation="uk.org.ogsadai. … .SQLStoredProcedureActivity"
   schemaFileName="http://localhost:8080/…/sql_stored_procedure.xsd"/>
<activityMap name="deliverFromURL"
    class="uk.org.ogsadai. … .DeliveryFromURLActivity"
    schemaFileName="http://localhost:8080/…/deliver_from_url.xsd" />
 <activityMap name="deliverToURL"
    class="uk.org.ogsadai. ….DeliveryToToURLActivity"
    schemaFileName=" http://localhost:8080/…/ deliver_to_url.xsd" />
```

Factory Configuration: RoleMaps

- Rolemapper maps grid credentials to database roles
- Java implementation *SimpleRolemapper* is provided with the release:
  - maps the distinguished name of the user to a username and password
  - Username and password are provided in a separate file

```
<roleMap name="SimpleRolemapper"
         implementation="uk. … .SimpleFileRoleMapper“
         configuration="examples/ExampleDatabaseRoles.xml"
/>
```

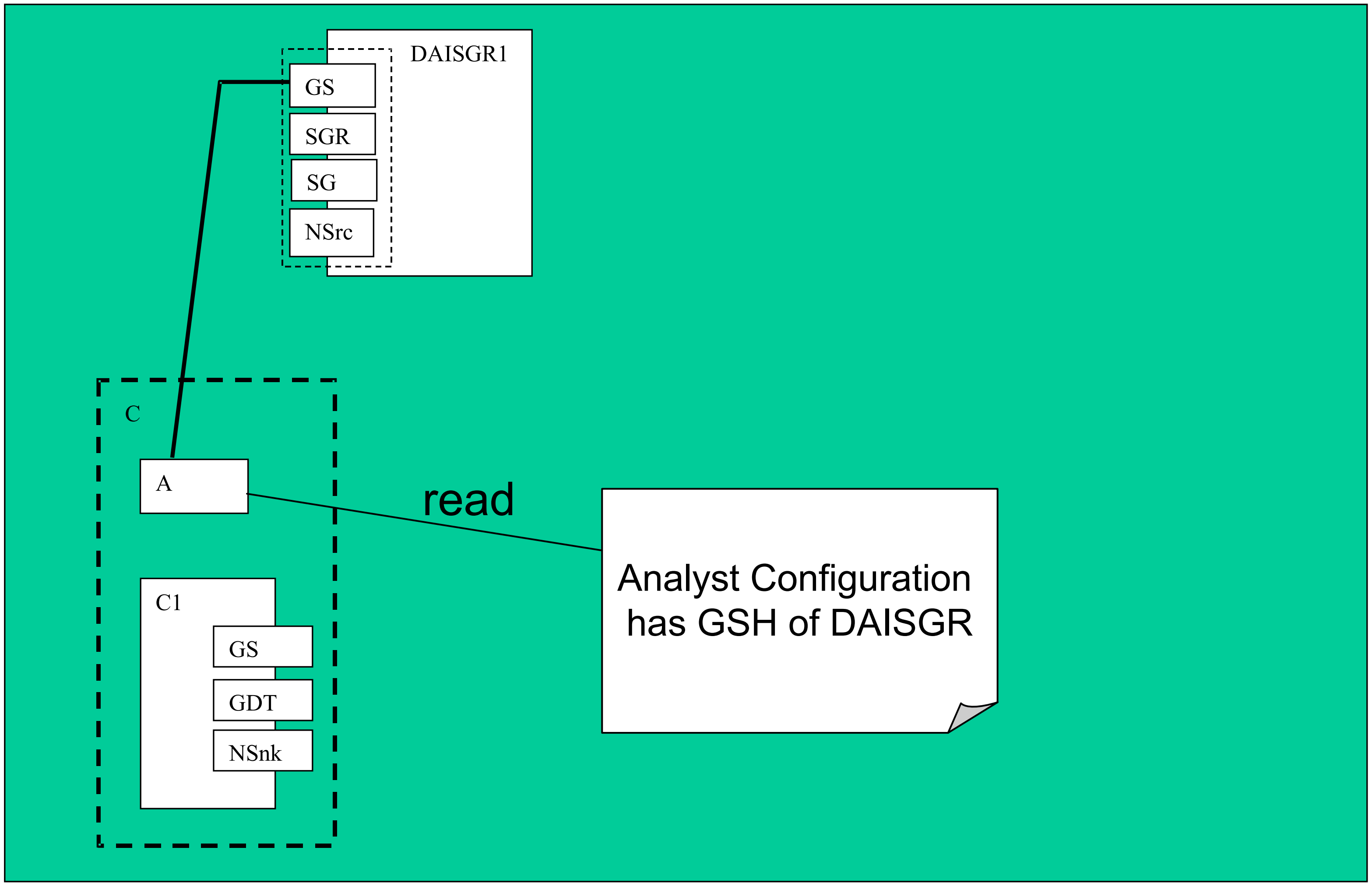
Factory Registration

- Through meta-data (SDEs) factory exposes
  - details from the configuration file, i.e.
    - data manager information
    - activities supported
    - relational metadata: database schema
  - Metadata about components (not shown earlier)
- Registration file allows GDSF to register with a DAISGR

Factory RegistrationList

```
<gdsf:gdsfRegistrationList … >

  <gdsf:gdsfRegistration name="defaultRegistration‵
   gsh="http://localhost:8080/ogsa/services/ogsadai/Grid
   DataServiceRegistry"/>

   <!-- can have more entries here -->

</gdsf:gdsfRegistrationList>
```

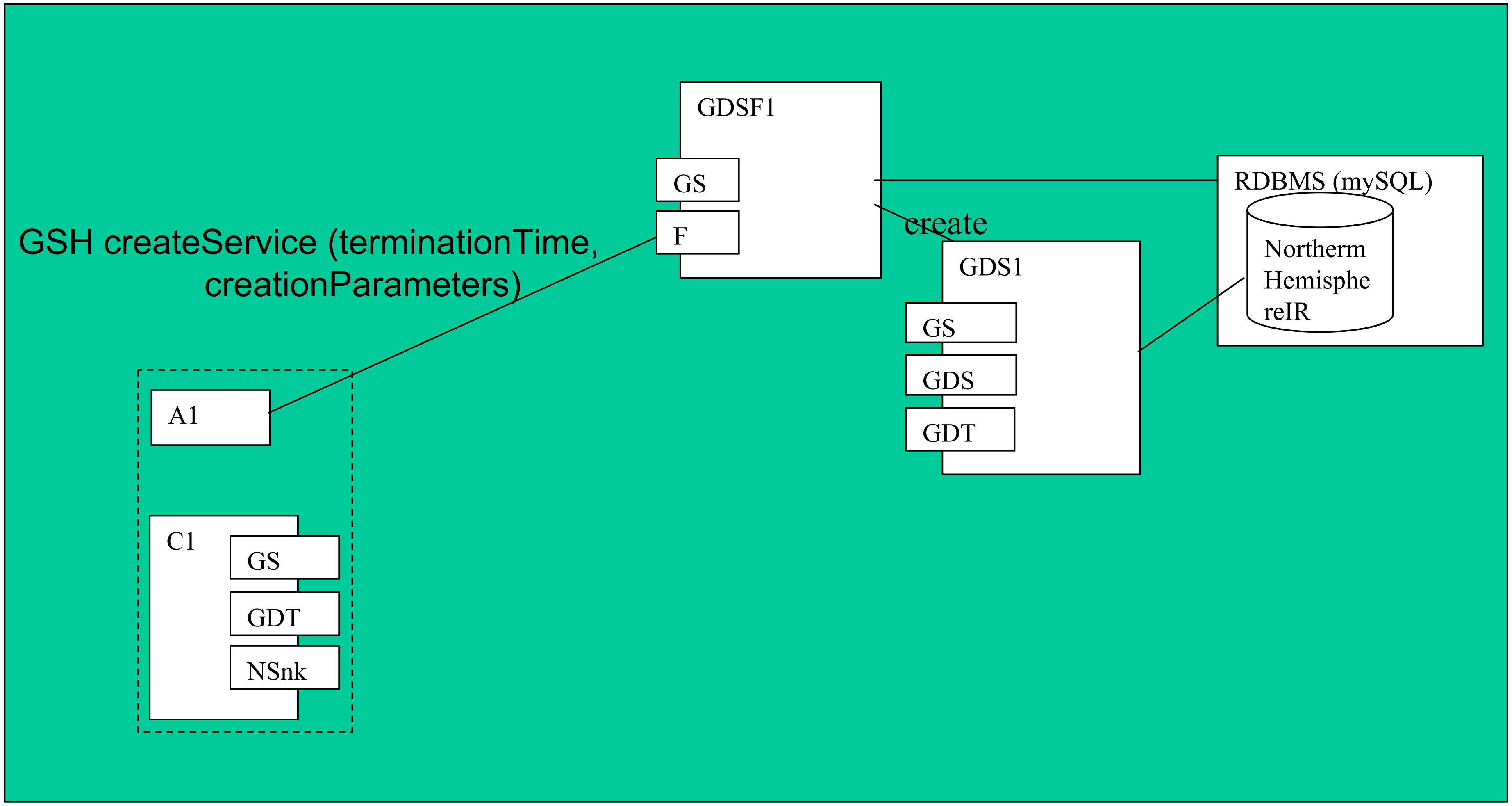
Analyst Starts and Identifies Factory



Registry Query

- Query for registered
  - GridServices
  - GridDataServices
  - GridDataServiceFactories
- XPath queries possible, for example
  - //path/data[@name="NorthernHemisphereIR"]
- Registry must be able to apply this and resolve it to a matching factory instance
- Factory registers its GSH on startup (if specified in the configuration)

Analyst Uses Factory Instance To Create GDS Instance



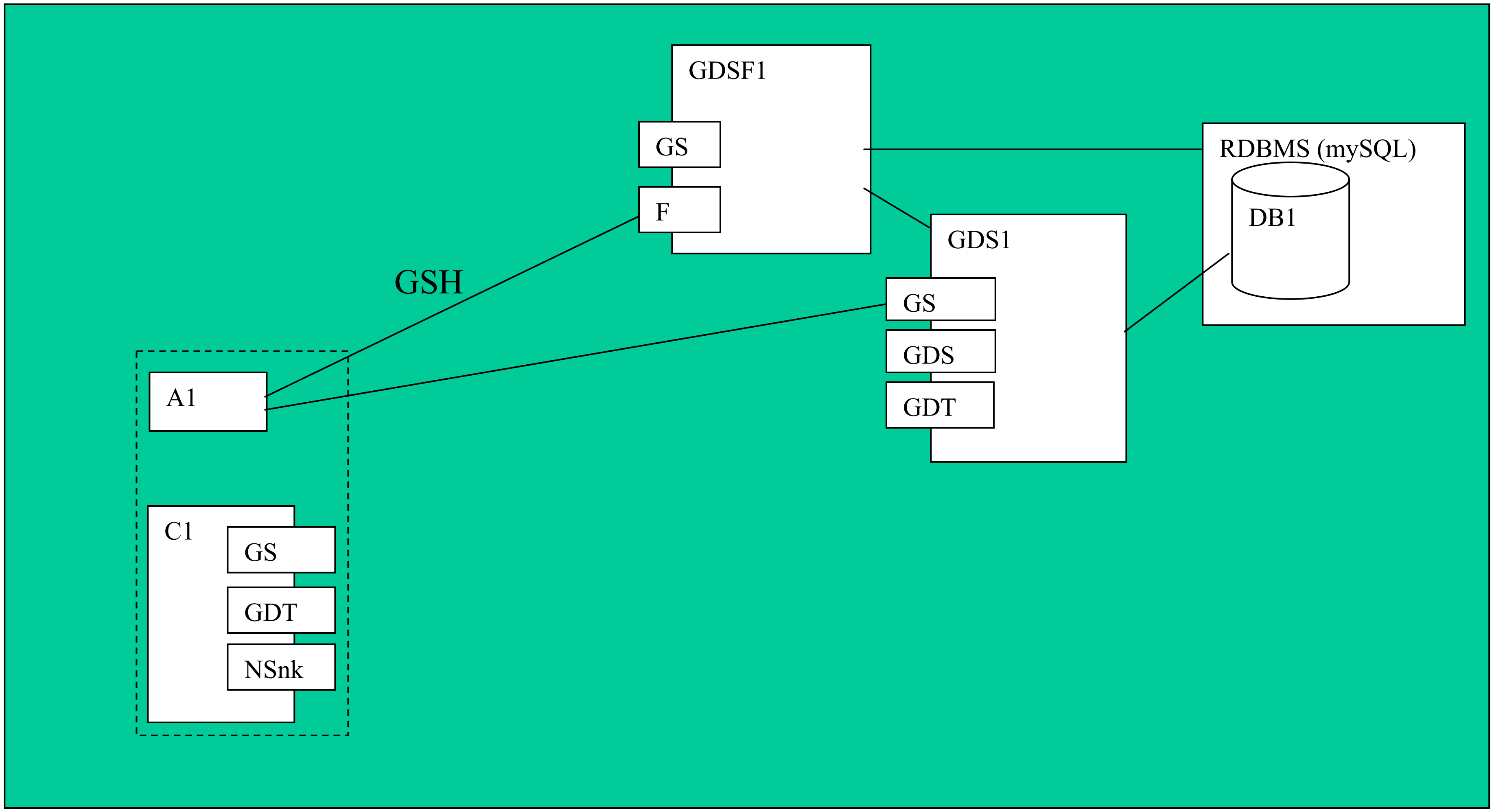
GDSF Creation Parameters

- In Release 3 the creation parameters are empty
- GDSF is associated with exactly one Data Resource
- GDSF will create a GDS configured for this Data Resource

GDSF Configures GDS Instance

- GDS is configured using information from the GDSF configuration
- Interfaces used to configure GDS are not exposed
  - They are particular to the implementation of GDSF and GDS
- Client requests actions to be taken by the GDS on the data resource by using a GDS-Perform document

Analyst maps GDS GSH



GDS-Perform document

- GDS Perform document contains activities and an optional documentation element
- Output from one activity can be used by another activity
- Any hanging outputs will be delivered with the SOAP response (synchronous)
- Using delivery activities, the output of a query can be delivered asynchronously (via HTTP, FTP, GridFTP)

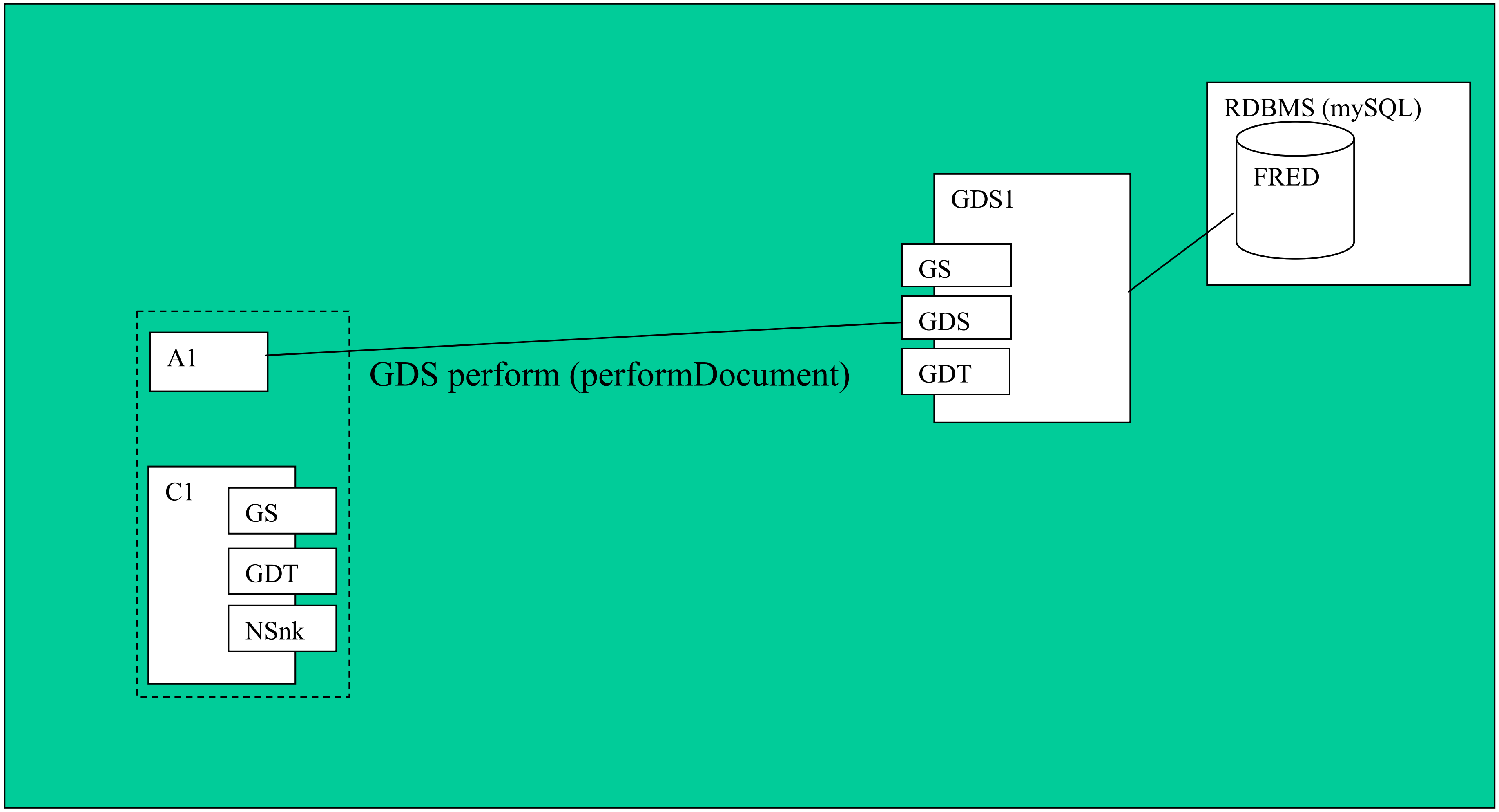
Analyst Formulates Query As GDS Perform Document

```
<gridDataServicePerform
     xmlns="http://ogsadai.org.uk/namespaces/2003/07/gds/types">
   <documentation>
      Select with data delivered with the response
      request stored then executed.
   </documentation>
   <sqlQueryStatement name="statement">
   <expression>
      select * from littleblackbook where id=10
   </expression>
   <webRowSetStream name="statementresult"/>
   </sqlQueryStatement>
</gridDataServicePerform>
```

GDS Perform Document Schema

- The WSDL for the GDS portType specifies the general schema that the perform method accepts
- The complex type ActivityType forms a base for extension by all activities
- The GDS configuration defines the operations that a GDS will perform
- The GDS will generate the GDS perform document schema on request based on the specified configuration

Analyst Passes Request to GDS and Retrieves Data From Response



GDS Response Documents

GDS response document contains:

- A named *response* element referencing a *request*
- For each activity in the request, a *result* element, referencing the name of the activity, which contains the result data
  - *sqlQueryStatement*
  - *xPathStatement*
  - *zipArchive*
  - …

The Data In The Response

```
<gridDataServiceResponse
   xmlns="http://ogsadai.org.uk/namespaces/2003/07/gds/types">
  <result name="statement" status="COMPLETE"/>
  <result name="statementresult" status="COMPLETE">
    <![CDATA[<?xml version="1.0" encoding="UTF-8"?>
    <!-- DOCTYPE RowSet PUBLIC '-//Sun Microsystems,
        Inc.//DTD RowSet//EN`
        'http://java.sun.com/j2ee/dtds/RowSet.dtd' -->
    <RowSet>
        . . .
    </RowSet>
  </result>
</gridDataServiceResponse>
```

Analyst Removes GDS Instance

- This is done either
  - by the GDS instance itself when the lifetime expires, i.e.
    - the container removes any Grid services whose lifetimes have expired
  - directly through the “Destroy” method

To Date

- Have assumed that OGSA/OGSI is a good thing
  - OGSA-DAI
  - Have adopted the OGSI approach
- Have first concentrated on data access
  - Data integration, for example, distributed query, pipelines, comes later
- Working Closely with GGF DAIS Working Group on *Grid Database Service Specification*
- Intentions to be a reference implementation

http://ogsadai.org.uk/

- Releases
- Support from the UK Grid Support Centre