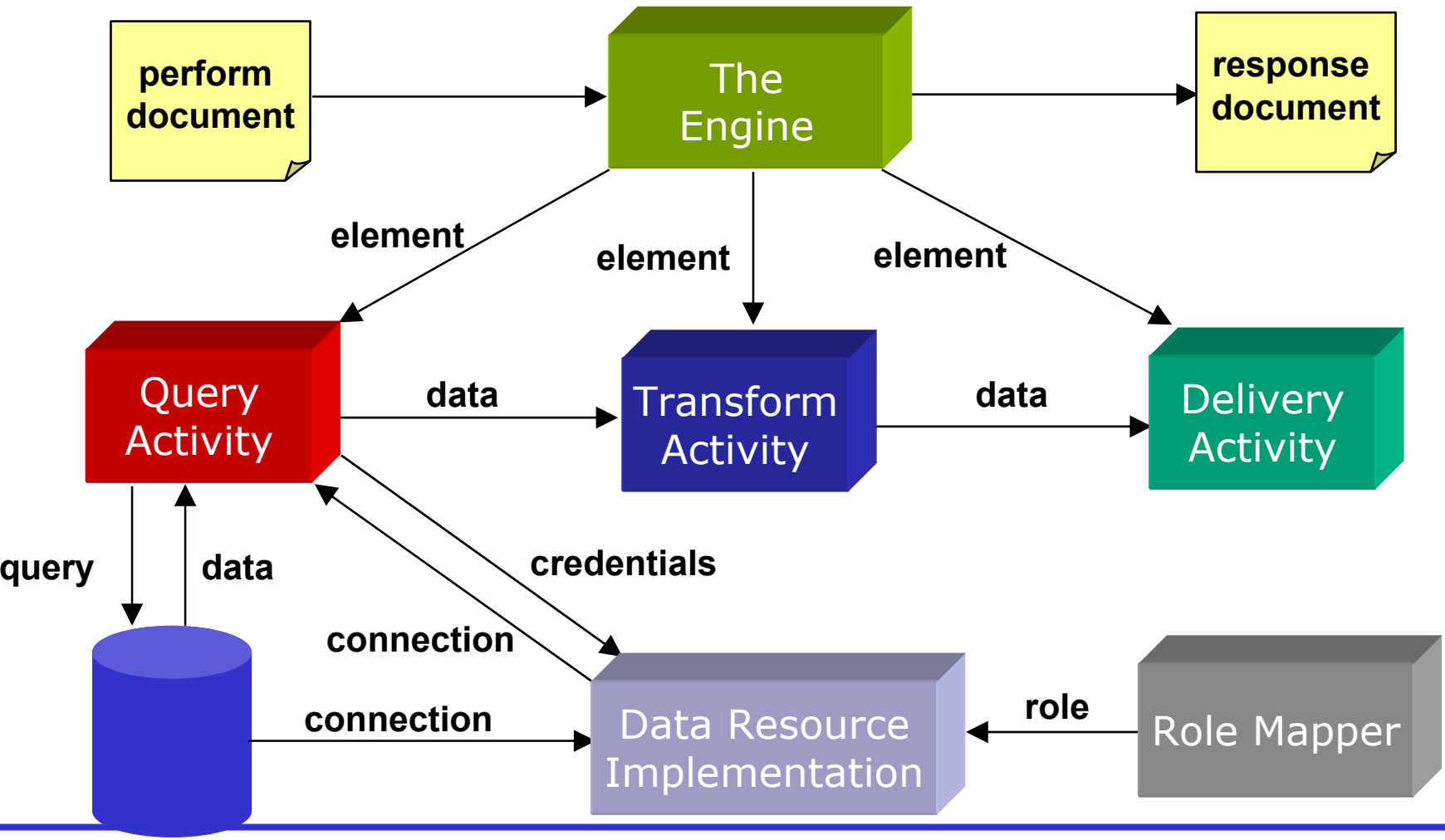
OGSA-DAI Internal Architecture

Andrew Borley
borley@uk.ibm.com

Overview

- Low-level components of a Grid Data Service
  - Engine
  - Activities
  - Data Resource Implementations
  - Role Mappers
- Extensibility of OGSA-DAI architecture
  - Interfaces
  - Abstract classes
  - Implementations

GDS Internals



- GDS has a document based interface
  - Consumes perform documents
  - Produces response documents
  - More sophisticated behaviour possible
    - Third party data delivery, get data, talk to other GDSs, …
- Motivation for using a document interface
  - Change in behaviour ≠> interface change
  - Reduce number of operation calls
  - Extensible

The GDS Engine

- Engine is the central GDS component
- Dictates behaviour when perform documents are submitted
  - Parses and validates perform document
  - Identifies required activities implementations
  - Processes activities
  - Composes response document
  - Returns response document to GDS

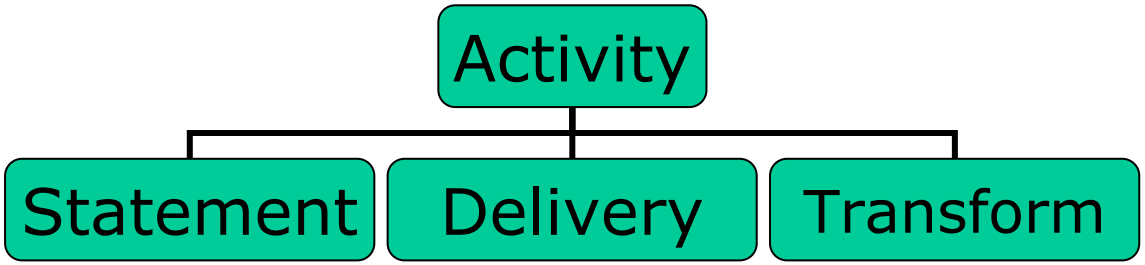
- Perform documents
  - Encapsulate a serialisation of multiple interactions with a service into a single interaction
  - Abstract each interaction into an “activity”
  - Data can flow from one activity to another
  - No control constructs present
    - no conditionals, loops or variables
- Not intended for human consumption
  - Currently hand-crafted (!)
  - Intend to be machine generated
    - Client toolkit will do this - soon to be (formally) released

Activities

- An Activity dictates an action to be performed
  - Query a data resource
  - Transform data
  - Deliver results
- Engine processes a sequence of activities
- Subset of activities available to a GDS
  - Specified in GDSF Configuration
- Data can flow between activities

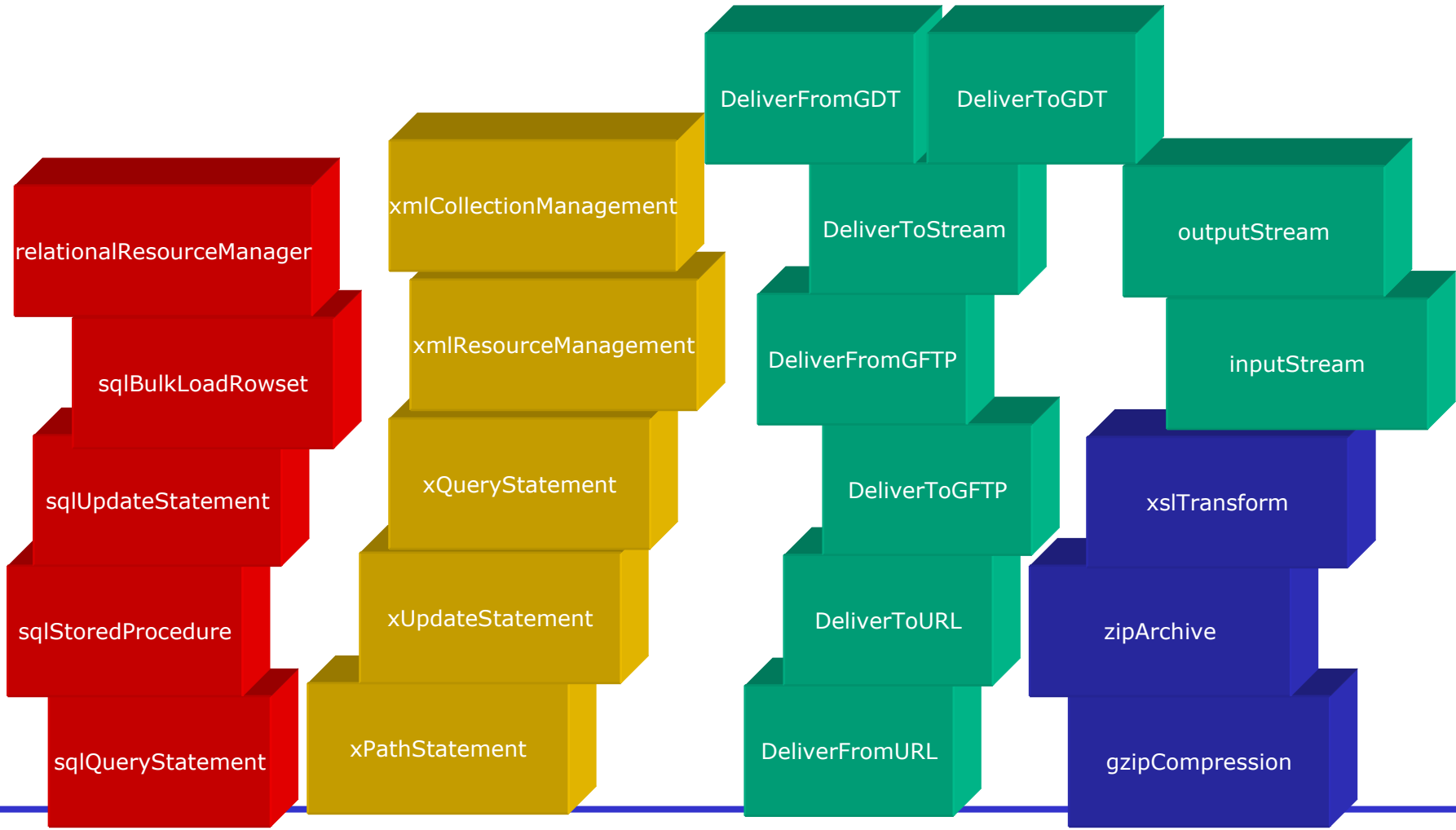
Activity Taxonomy

- Activities fall into three main functional groups



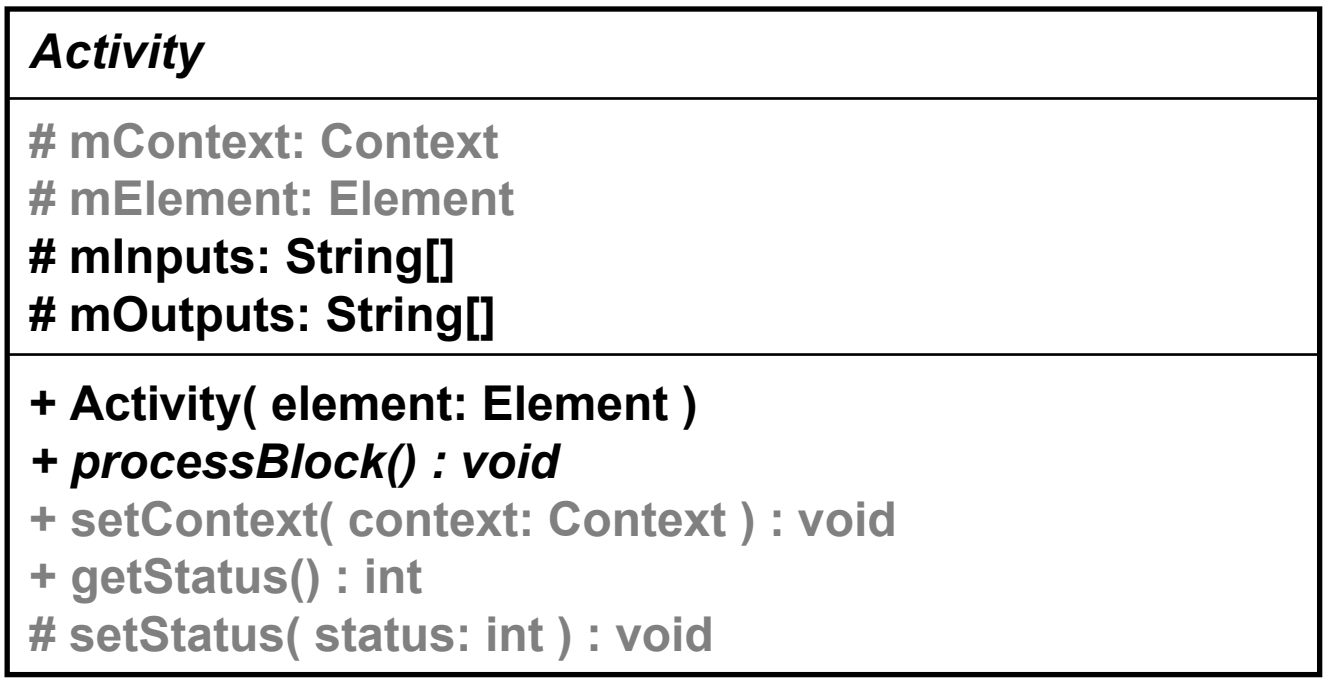
- Statement
  - Interact with the data resource, e.g. direct an SQL query to a DBMS
- Delivery
  - Deliver data to a third party
- Transform
  - Perform transformations on data, e.g. XSL Transform, compression

Building Blocks
Predefined Activities



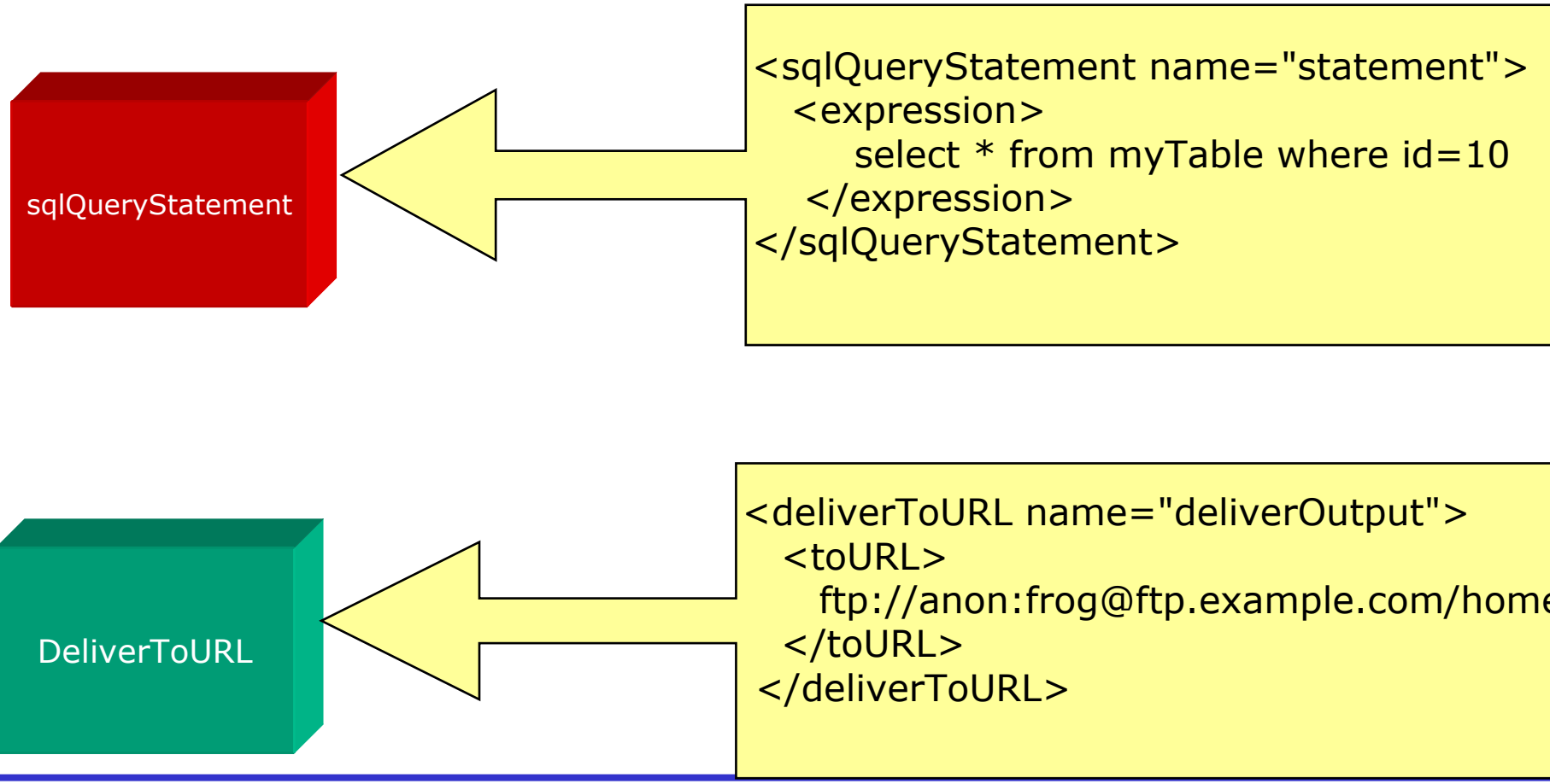
Activity Implementations

- Extensibility point
- All Activity implementations extend the abstract Activity class

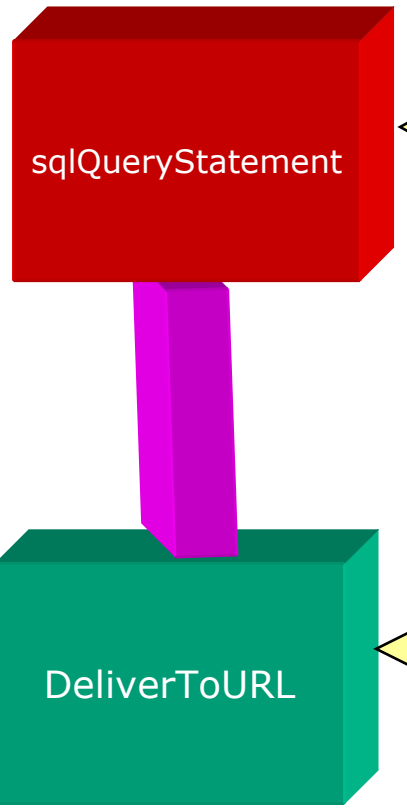


- Users can develop additional activities
  - To support different query languages
    - XQuery
  - To perform different kinds of transformation
    - STX
  - To deliver results using a different mechanism
    - WebDAV
- An activity requires
  - XSD schema — sql_query_statement.xsd
  - Java implementation — SQLQueryStatementActivity.java

Connected Activities



Connected Activities cont.




```
<sqlQueryStatement name="statement">
  <expression>
     select * from myTable where id=10
  </expression>
  <resultSetStream name="output1"/>
</sqlQueryStatement>
```

```
<deliverToURL name="deliverOutput">
  <fromLocal from="output1"/>
  <toURL>
     ftp://anon:frog@ftp.example.com/home
  </toURL>
</deliverToURL>
```

The Perform Document

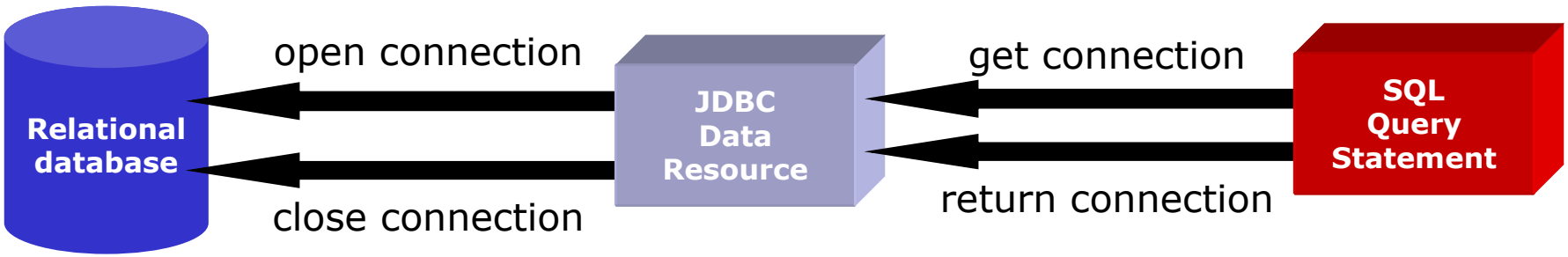
```
<?xml version="1.0" encoding="UTF-8"?>
<gridDataServicePerform
      xmlns="http://ogsadai.org.uk/namespaces/2003/07/gds/types"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://ogsadai.org.uk/namespaces/2003/07/gds/types
       ../../../../schema/ogsadai/xsd/activities/activities.xsd">

 <documentation>
  This example performs a simple select statement to retrieve
  one row from the test database. The results are delivered
  within the response document.
 </documentation>

 <sqlQueryStatement name="statement">
  <expression>
      select * from littleblackbook where id=10
   </expression>
  <resultSetStream name="output"/>
 </sqlQueryStatement>

 <deliverToURL name="deliverOutput">
  <fromLocal from="output"/>
  <toURL>ftp://anon:frog@ftp.example.com/home</toURL>
 </deliverToURL>

</gridDataServicePerform>
```

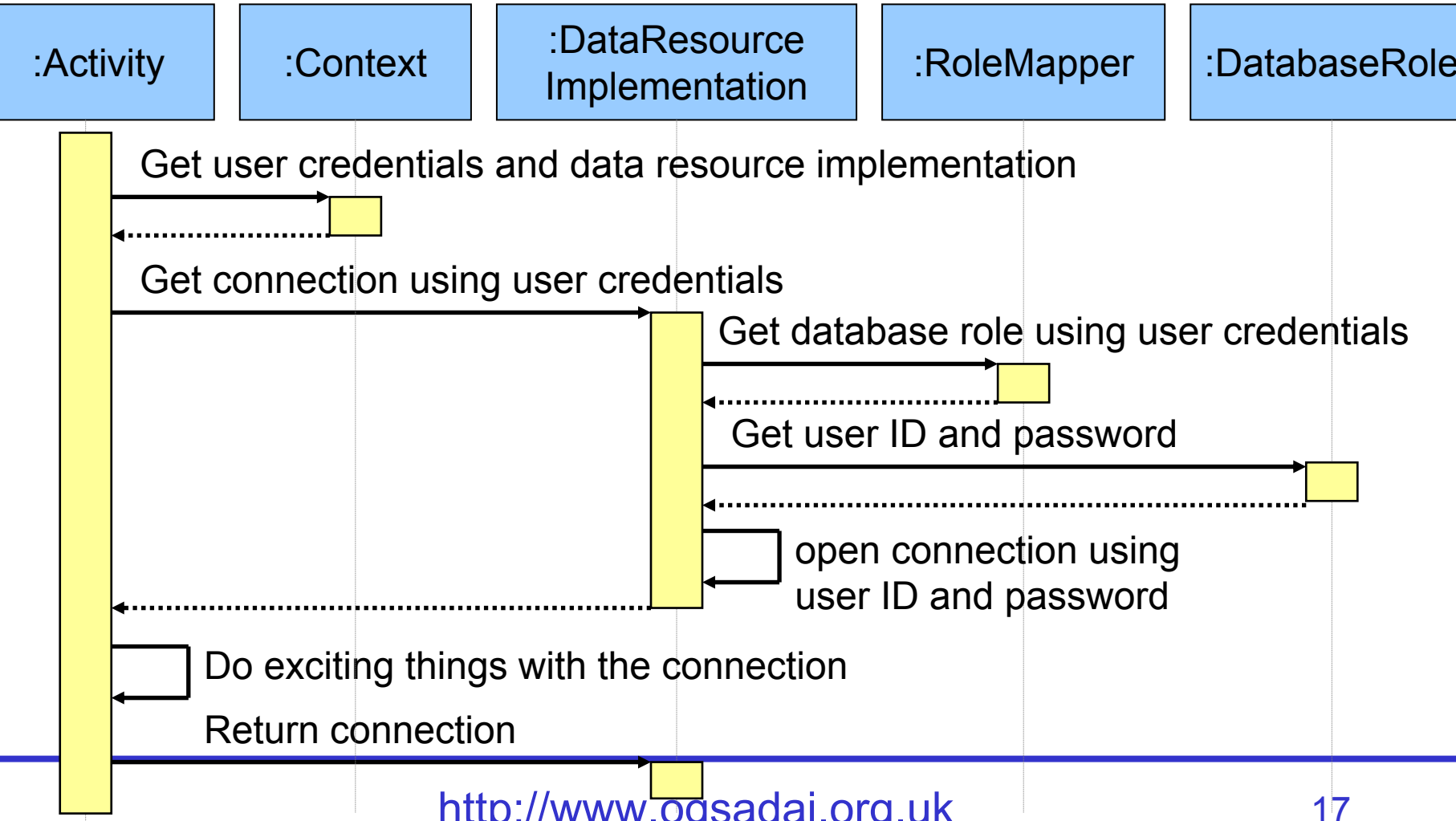
Activity Inputs and Outputs

- Activities read and write blocks of data
  - Allows efficient streaming between activities
  - Reduces memory overhead
- A block is a Java Object
  - Untyped but usually a String or byte array
- Interfaces for reading and writing
  - BlockReader and BlockWriter

Data Resource Implementations

- Governs access to a data resource
  - Open/close connections
  - Validate user credentials using a RoleMapper
  - Facilitate connection pooling
- Provided for JDBC and XML:DB




:Activity
:Context
:DataResource Implementation
:RoleMapper
:DatabaseRole
Get user credentials and data resource implementation
Get connection using user credentials
Get database role using user credentials
Get user ID and password
open connection using user ID and password
Do exciting things with the connection
Return connection

Advantages of the Activity Model

- Avoid multiple message exchanges
- Extensible
  - Developers can add functionality
  - Could import third party trusted activities
- Allows for optimisation
  - GDS engine can optimise internals

Issues with Activity Model

- Incomplete syntax
  - No typing data streams
  - Typing inputs and outputs
    - How do you specify how many inputs/outputs an activity can have?
    - How do you determine the data types that can be accepted?
- Keeping implementation and XML Schema fragment in synch
- Semantics not specified
- Puts workload on the server
- DAIS has factored out the perform document from the draft specs

Summary

- The Engine is the central component of a GDS
- Activities perform actions
  - Querying, Updating
  - Transforming
  - Delivering
- Data Resource Implementations manage access to underlying data resources
- Architecture designed for extensibility
  - New Activities
  - New Role Mappers
  - New Data Resource Implementations

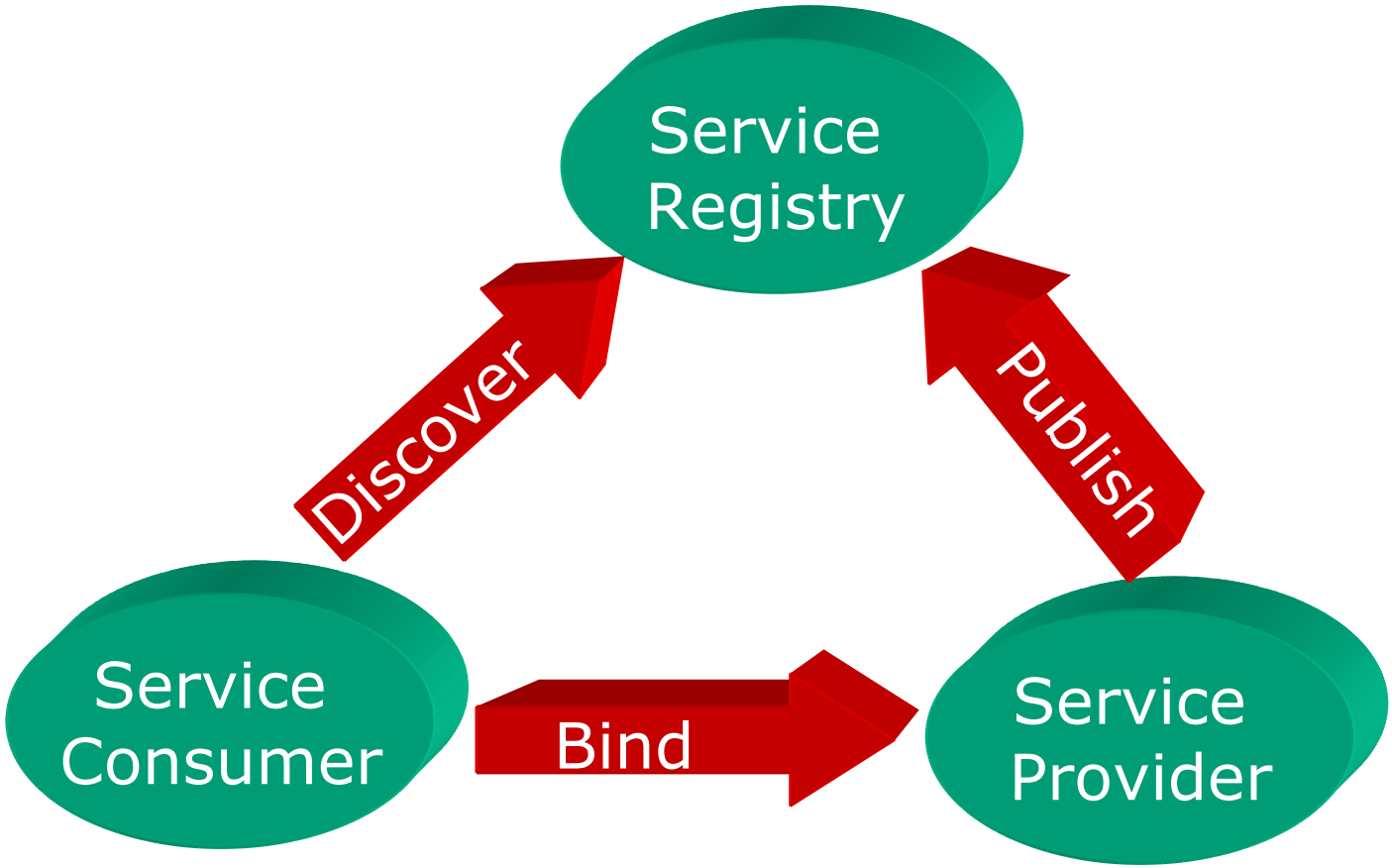
Questions?

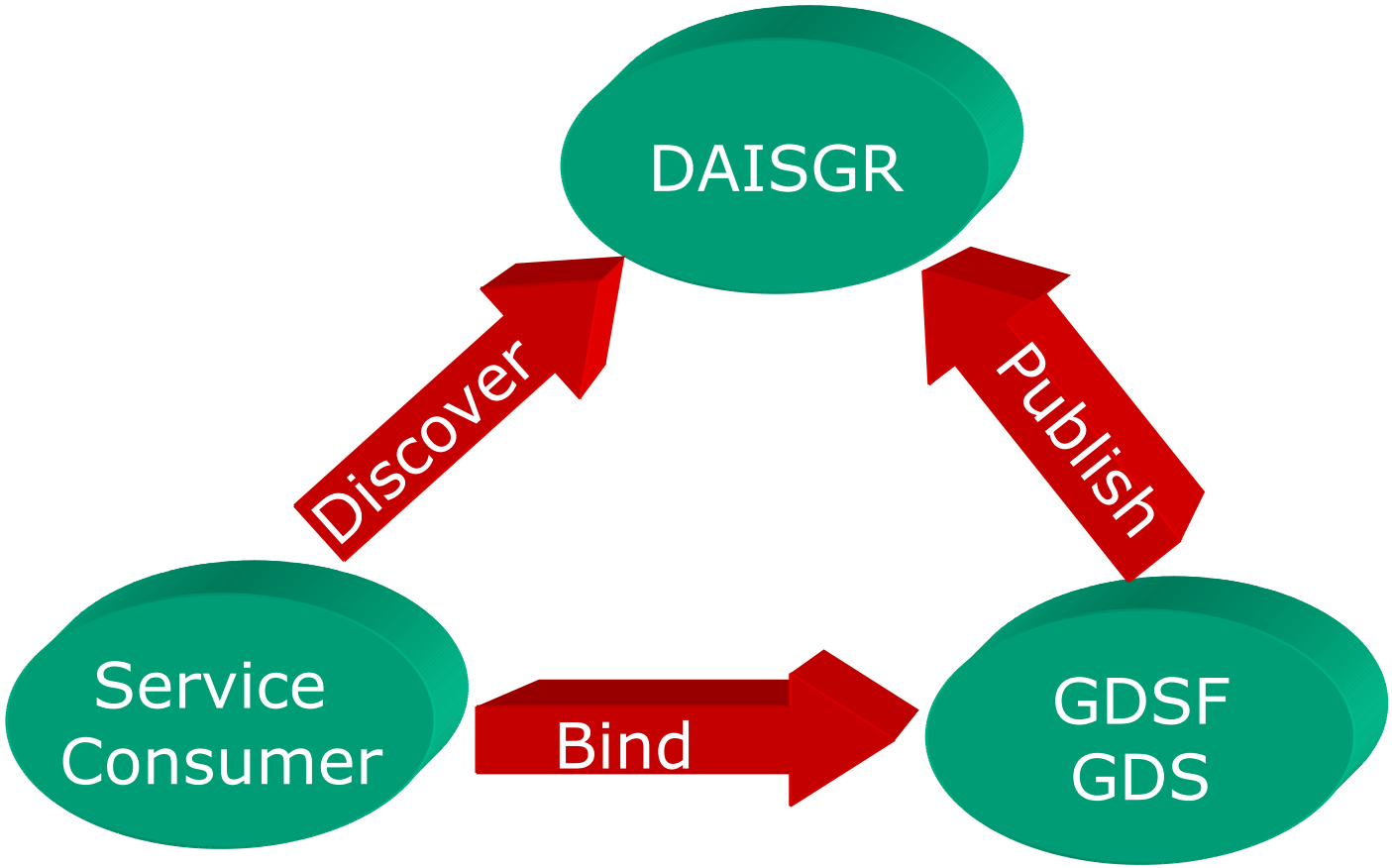
OGSA-DAI High-level Architecture

Andrew Borley
borley@uk.ibm.com

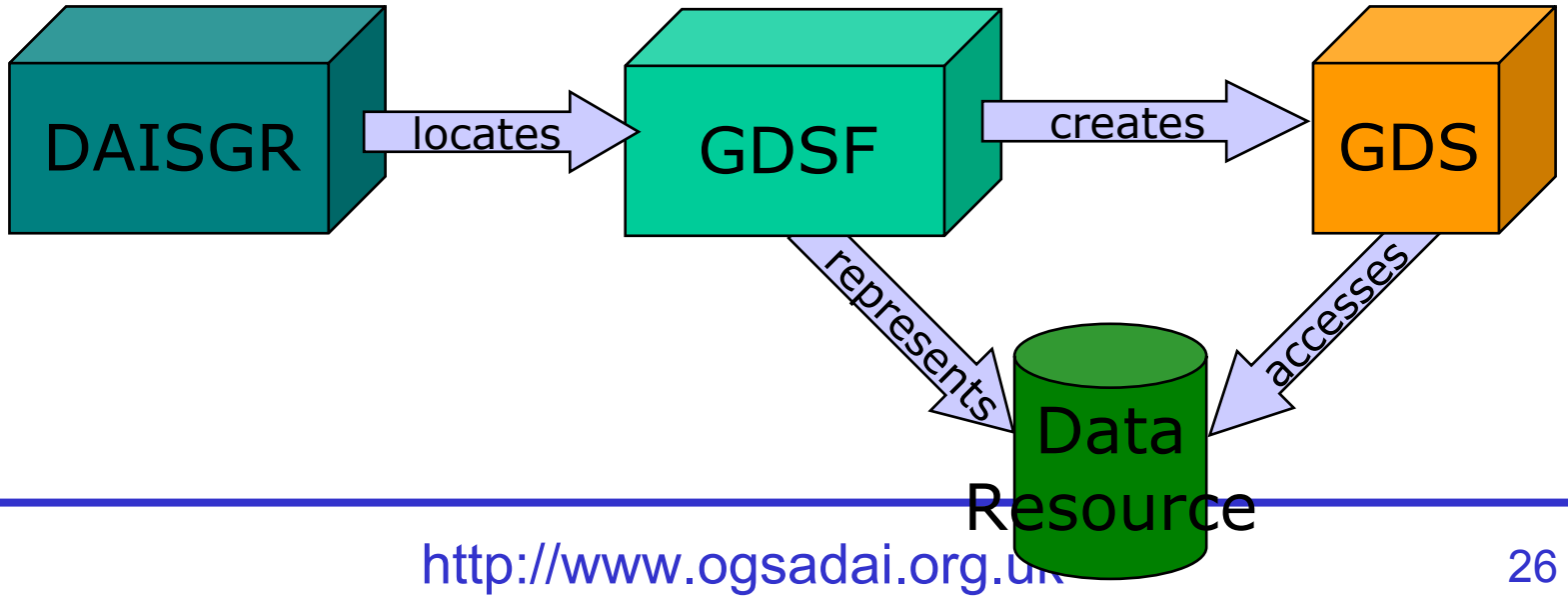
- Web Services → Grid Data Services
- Data Access & Integration Service Group Registry (DAISGR)
- Grid Data Service Factory (GDSF)
- Grid Data Service (GDS)
- Use-case Scenarios

Web Service Architecture




DAISGR
Discover
Publish
Service Consumer
Bind
GDSF
GDS

- OGSA-DAI uses three main service types
  - DAISGR (registry) for discovery
  - GDSF (factory) to represent a data resource
  - GDS (data service) to access a data resource



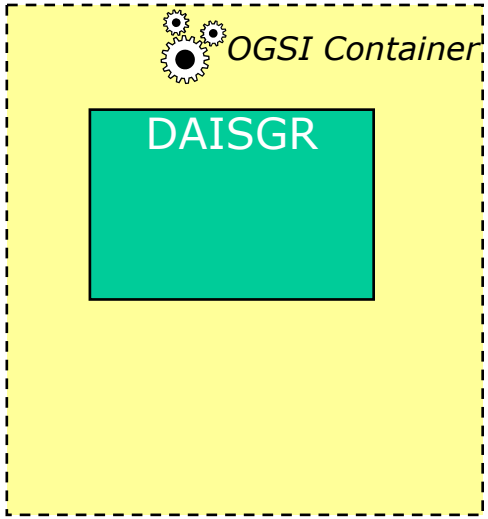
GDSF and GDS

- Grid Data Service Factory (GDSF)
  - Represents a data resource
  - Persistent service
    - Currently static (no dynamic GDSFs)
      - Cannot instantiate new services to represent other/new databases
  - Exposes capabilities and metadata
  - May register with a DAISGR
- Grid Data Service (GDS)
  - Created by a GDSF
  - Generally transient service
  - Required to access data resource
  - Holds the client session

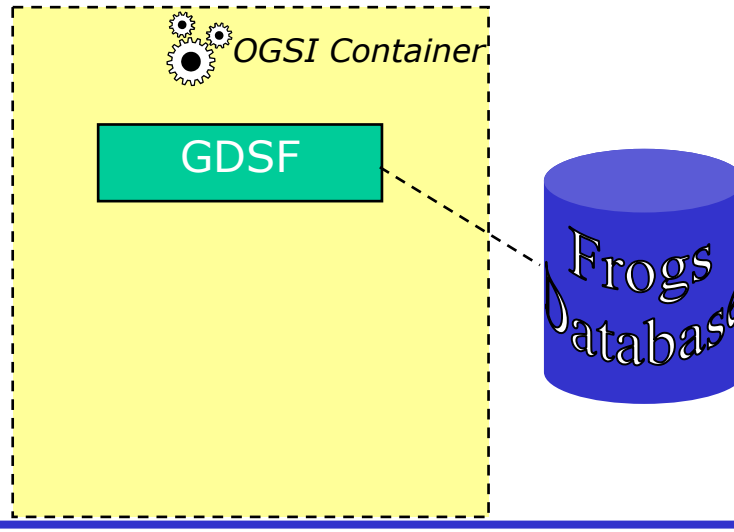
DAI Service Group Registry (DAISGR)

- Persistent service
- Based on OGSI ServiceGroups
- GDSFs may register with DAISGR
- Clients access DAISGR to discover
  - Resources
  - Services (may need specific capabilities)
    - Support a given portType or activity

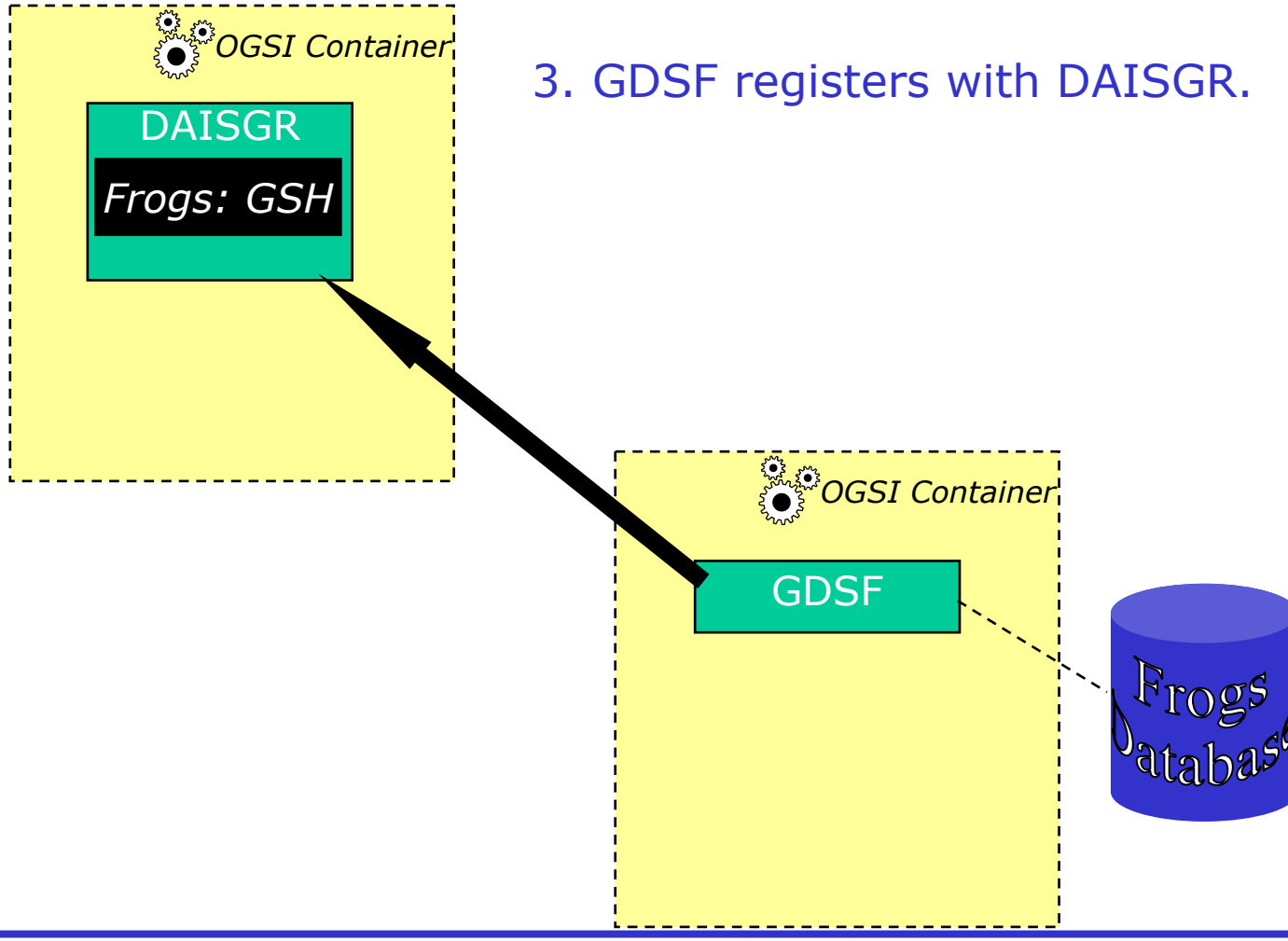
Interaction Model: Start up



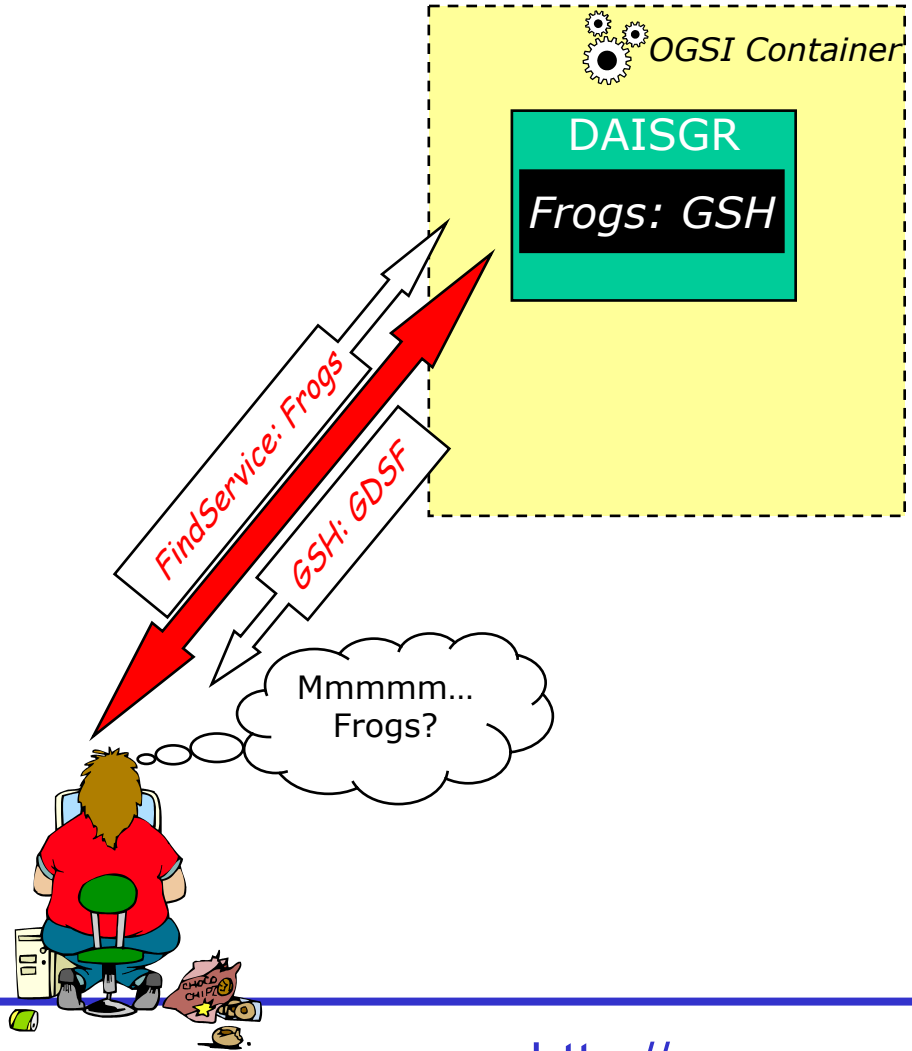
1. Start OGSI containers with persistent services.
2. Here GDSF represents Frog database.



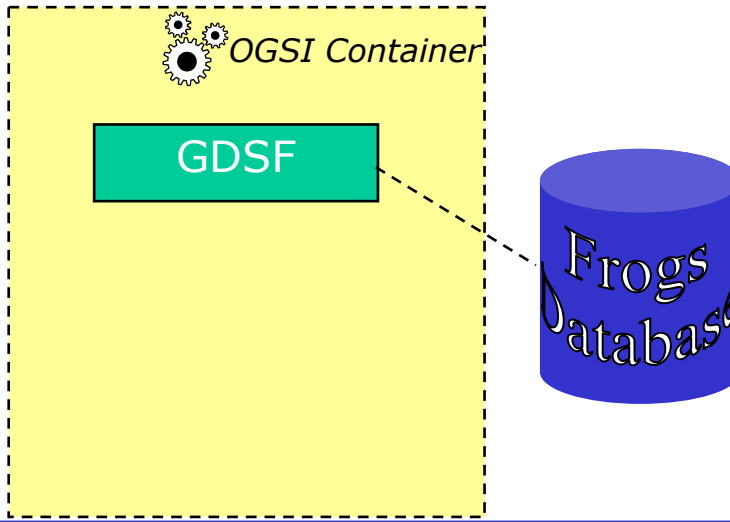
Interaction Model: Registration



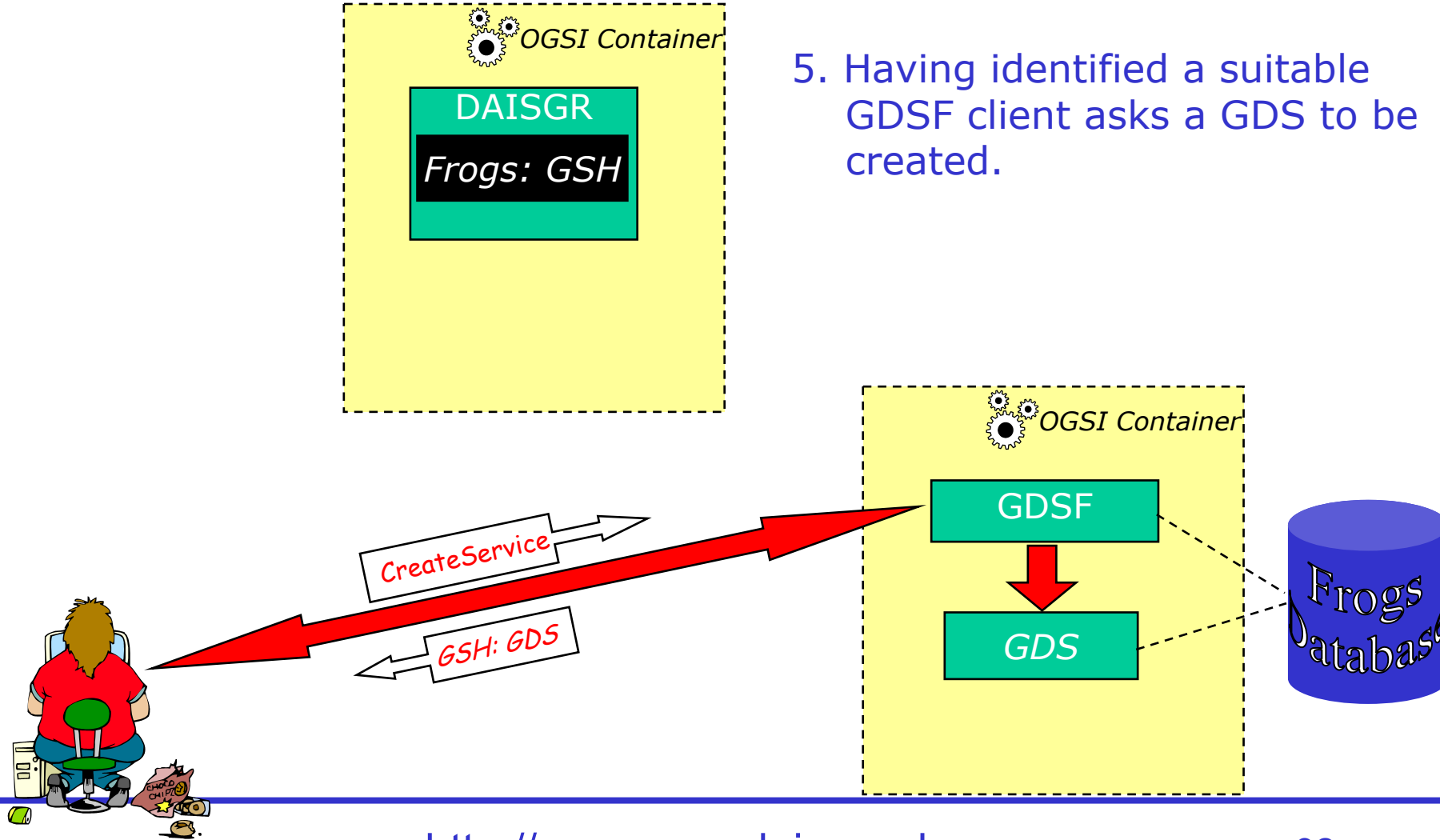
Interaction Model: Discovery



4. Client wants to know about frogs. Can:
   (i) Query the GDSF directly if known or
   (ii) Identify suitable GDSF through DAISGR.

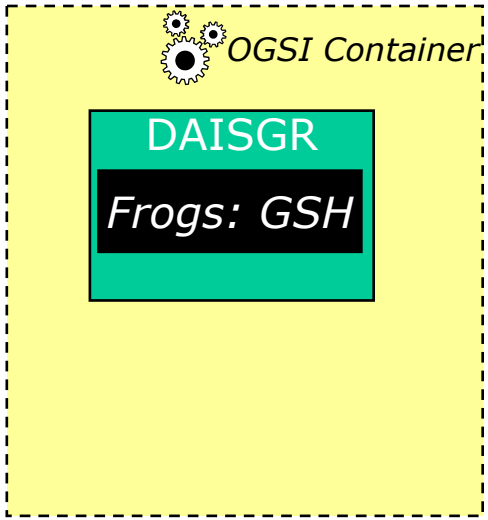


Interaction Model: Service Creation

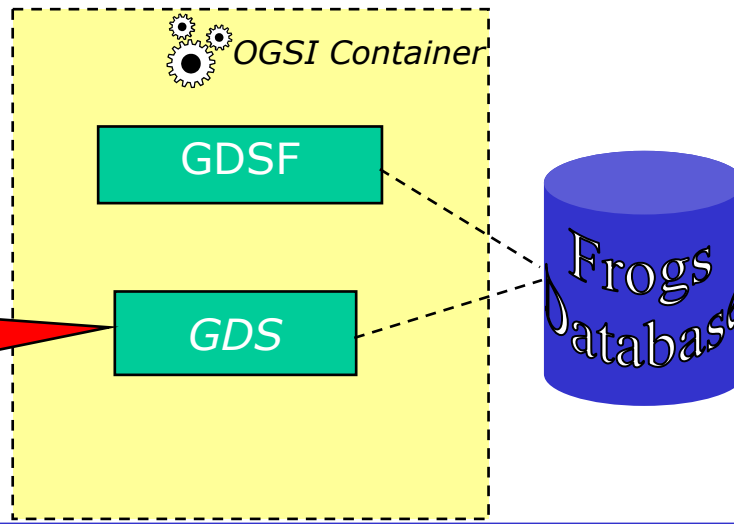


5. Having identified a suitable GDSF client asks a GDS to be created.

Interaction Model: Perform



6. Client interacts with GDS by sending Perform documents.
7. GDS responds with a Response document.
8. Client may terminate GDS when finished or let it die naturally.



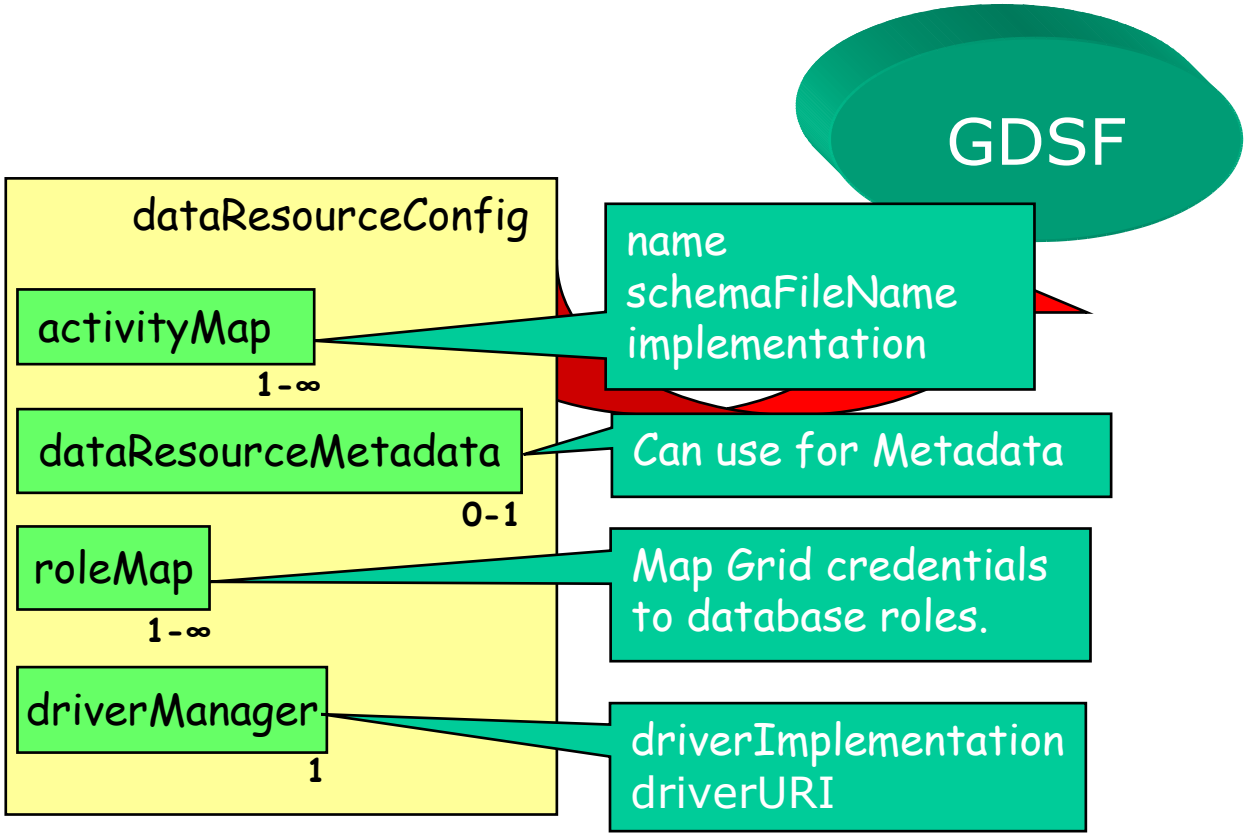
Perform Document

Response Document

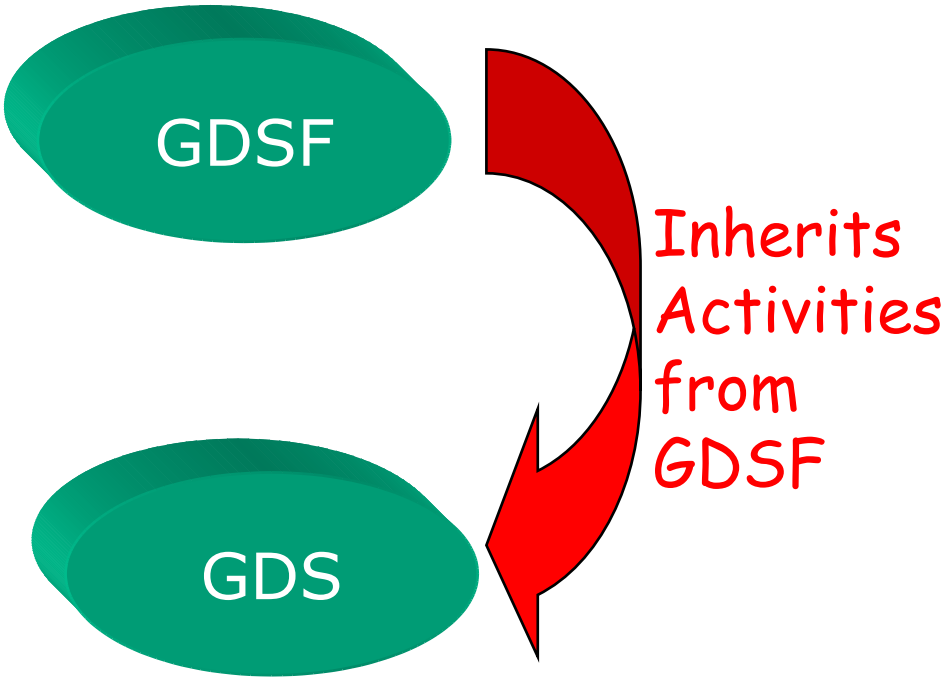
Interaction Model: Sum up

- Only describe an access use case
  - Client not concerned with connection mechanism
  - Similar framework could accommodate service-service interactions
- Discovery aspect is important
  - Probably requires a human
  - Needs adequate definition of metadata
    - Definitions of ontologies and vocabularies - not something that OGSA-DAI is doing …

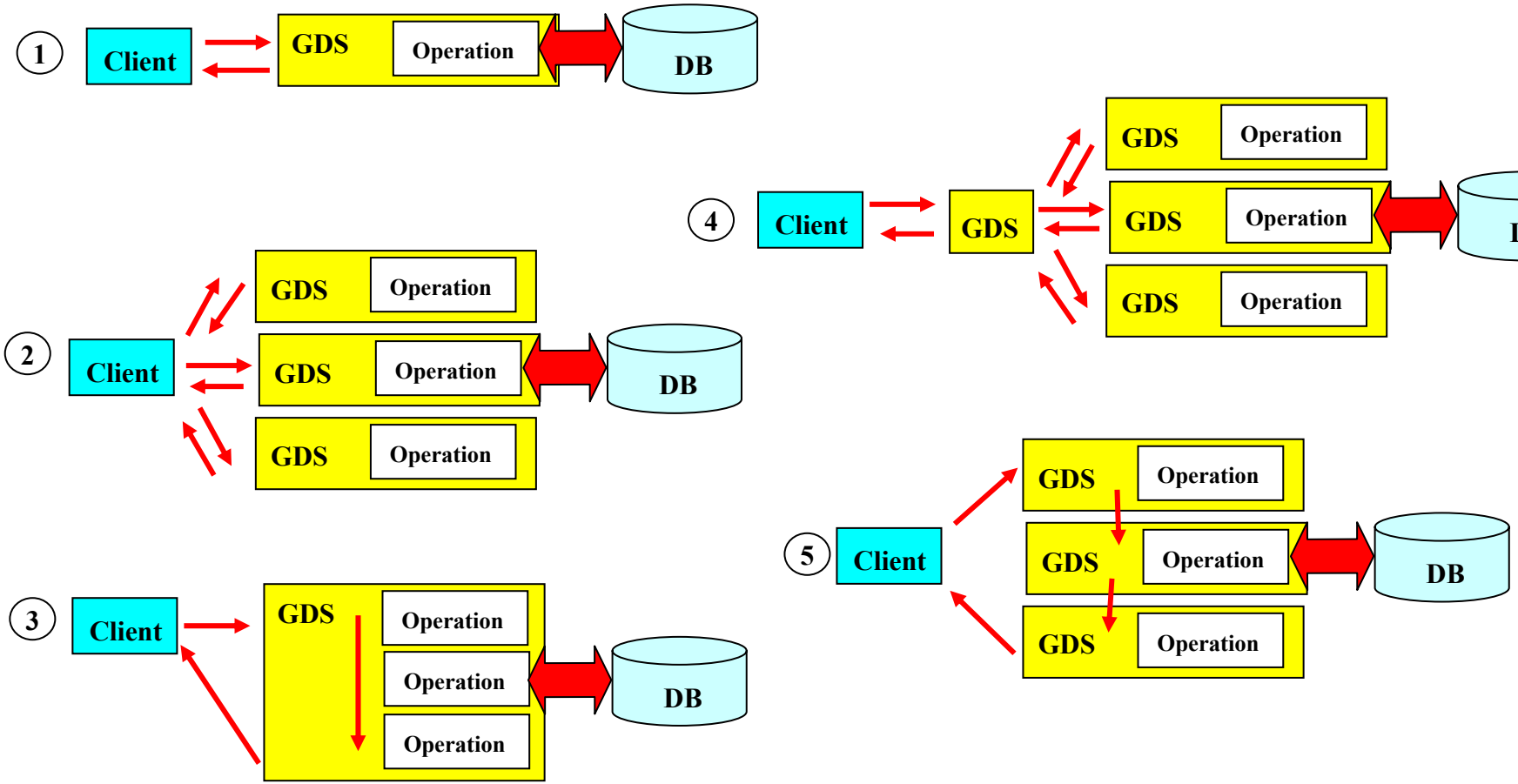
GDSF Configuration



GDSF Configuration 2



GDS Composition



Summary

- Assumed OGSA/OGSI is a good thing
- First concentrated on data access
  - Data integration comes later
- Working Closely with GGF DAIS Working Group
- OGSA-DAI did not attempt to do workflow
  - Came dangerously close
  - Would have been a mistake to do so
    - Emerging standards
    - Do not want to re-invent the wheel
- Framework will change with DAIS