

Grid Computing

High Performance Computing

- Supercomputer (Tightly coupled)
- Clusters (loosely coupled)
- Grids (Virtual Organisations)
 - Virtualization of HPC applications as Grid Services

Parallel Processing

- Shared Memory
- Distributed Memory (network)
- Hybrid-Distributed-Shared Memory (today mostly used, multiple Symmetric Multi-Processing Nodes [SMPs])

→ Parallel Programming (Threads , Parallel Computing → Shared Memory, Distributed Computing → Message Passing, Hybrids)

Voluntary Computing (Global Distributed Computing)

- SETI @ HOME (2,3 Mill. Single Processor years since 1999)
- Google (150 000 000 queries per day)

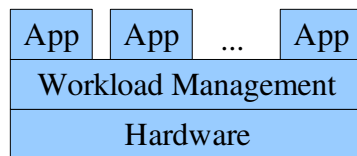
Grids

- Virtual Supercomputer (Integrates computers, networks, Databases, scientific instruments)
- Virtual Organisation
- coordinates resources that are not subject to centralized control
- uses standard, open, general-purpose protocols and interfaces
- delivers non-trivial Quality of Service (QoS)
 - Support dynamic negotiation of QoS
 - provide guarantees with respect to time/costs
 - web service level agreements (WSLA) = contract on price, performance, etc.
 - applications specific performance and pricing dependent on meta data
 - Advanced resource reservation

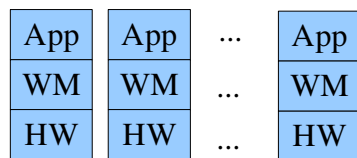
Grids as Service Orientated Architecture (SOA)

- Attributes
 - Modularity
 - composability

- encapsulation
- loose-coupling,
- abstraction
- reusability
- discoverability
- Technical Specifications
 - Services descriptions: WSDL
 - Encoding Messages from/to services: SOAP
 - Security: WS-Security
 - Address services: WS-Addressing
 - Manipulate state: WS-Resource Framework
 - Deliver notifications: WS-Notification
- Evolution of Enterprise IT Architecture

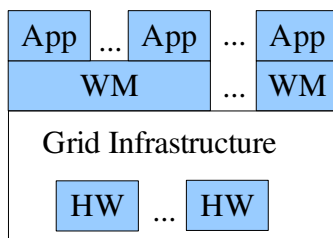


(a) Centralisation



(Vertical decoupling)

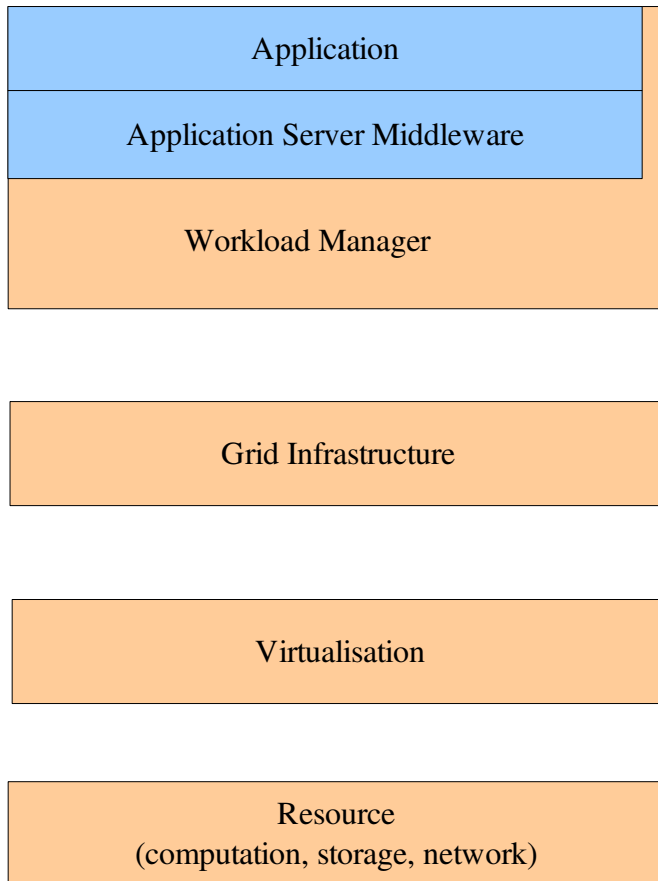
(b) Disintegration



(Horizontal Integration)

(c) Reintegration

- SOA Architecture Stack



Workload Manager

Scheduling of granularity-varied tasks on behalf of multiple applications → *Workflow*
Orchestration of Batch System (=Job scheduler beyond single OS) & Grid infrastructure (=distributed applications)

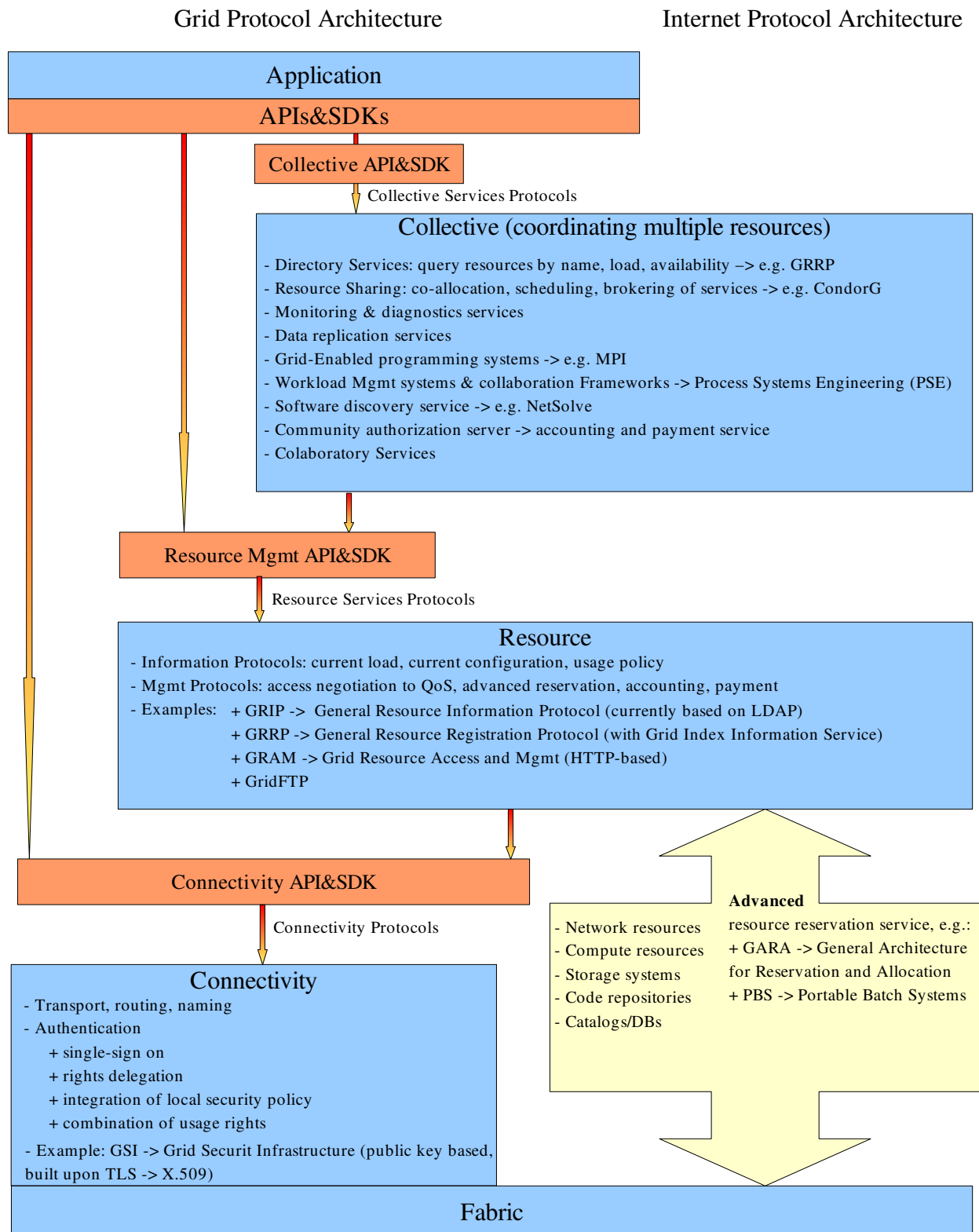
Virtualisation

Physical resource abstraction
Execution Virtualisation (e.g. VMWare, Xen, ...)
Virtual Machine Based (e.g. J2EE, .NET)
Communication Virtualisation (VLANs, VPNs, ...)

Resource Management

Resource Modelling through discovery, provision and Mgmt Of QoS
Monitoring & Notification by monitoring state of resource and notifying of state changes → Logging of Events and State transitions
Allocation of resources by negotiating the required service level(s) and dynamic creation of service level agreement → Assures QoS
Life-Cycle Management by provisioning/decommissioning by automatically configuring the allocated resources for application use
Accounting & Auditing by tracking shared resource usage and charging users/applications → Provides mechanisms for transferring cost among user communities

- Virtual Organisations (Vos)
 - Participants of VO provide resources to VO restricted by its usage policies & rules
 - VO structure may vary over time → Structure Management by defining/querying relationships within the VO
 - Interoperability through Services, Protocols, APIs
- Layered Grid Architecture (see „Grid infrastructure“ above) vs. Internet Architecture

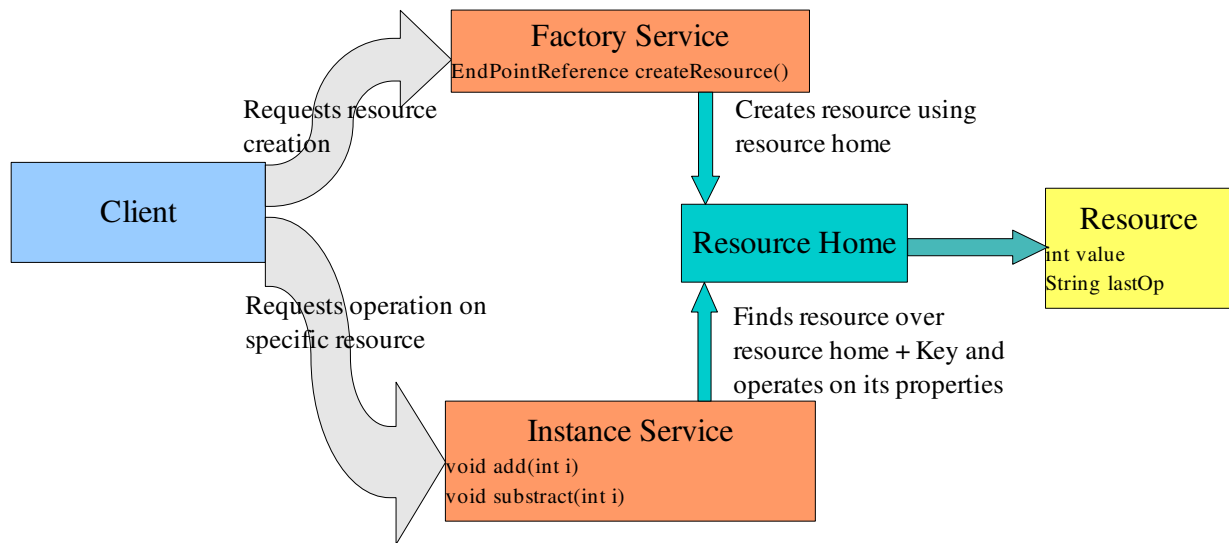


- **OGSA** – Open Grid Service Architecture used by GT4 (Globus Toolkit)
 - Based on web services and SOA
 - NOT object-orientated & NOT layered
 - replaces OGSII (= OGS-Infrastructure) used by GT3
 - too object-orientated (state and service is not separated)
 - not working well with existing web service tools
 - Standardizes (stateful) grid services compatible with web services are structured into 8 OGSA service categories:
 - Infrastructure services
 - Resource management services
 - Data services
 - Context services
 - Information services
 - Self-Management services
 - Security Services
 - Execution management services:
 - Mgmt of
 - Units of work
 - Provisioning
 - Life time
 - Models Resources
 - Service Container:
 - static information such as executables, OS version, policies, security, libraries installed
 - dynamic information such as load, QoS information
 - Persistent State Handle Service (for Job Migration)
 - tracking state
 - holds resource handle (abstract name + resource state)
 - holds resource address (e.g. file name, primary key, ...)
 - Job Mgmt and Monitoring
 - Job submission description language
 - Job Manager
 - Candidate Set Generator
 - Execution Planning Service
 - Reservation Service
 - Resource selection (decide where to execute unit of work)

- Interaction with other OGSA Services
 - Deployment & Configuration
 - Naming Service
 - Information Service
 - Fault Detection & Recovery Service
 - Auditing, billing logging service
 - Accounting
- **WSRF** – Web Services Resource Framework
 - Converging of grid and web technologies → WSRF
 - Distinction of
 - Resources with resource key (Stateful entities acting on services) and
 - Web services (Stateless)
 - Example:
 1. Request the server holding the 'weather service' from the discovery service on server A
 2. Receive the response that the 'weather service' is located on server B
 3. Request 'weather service' description; i.e.
 4. Receive WSDL description → Client stub is generated
 5. send SOAP (= invocation protocol) request, which was marshaled/serialized from the 'local invocation request' by the client stub, invoking getWeather() with parameter 'europe' at the weather service on server B
 6. The server stub, which is generated in advance from WSDL by SOAP engine (e.g. Apache Axis), receives SOAP request and unmarshals/deserializes it to 'local language'
 7. Request is processed and SOAP response is generated and sent back to client
 8. client stub receive SOAP response and unmarshals it to 'client dialect', e.g. 'cloudy, likely to rain'
 - Advantages:
 - platform & language independent
 - XML & HTTP (= transport protocol → to pass firewalls!)
 - Disadvantages:
 - Lack of versatility
 - NO persistence (non-transient), notifications, transactions, life-cycle mgmt, ...
 - Mgmt of WS-Resources = Web Service[URI+WS-Addressing] (stateless) + Resource (stateful)
 - Defines how to make stateless services stateful → Specifies stateful web services, which are required by OGSA
 - WSRF Specification

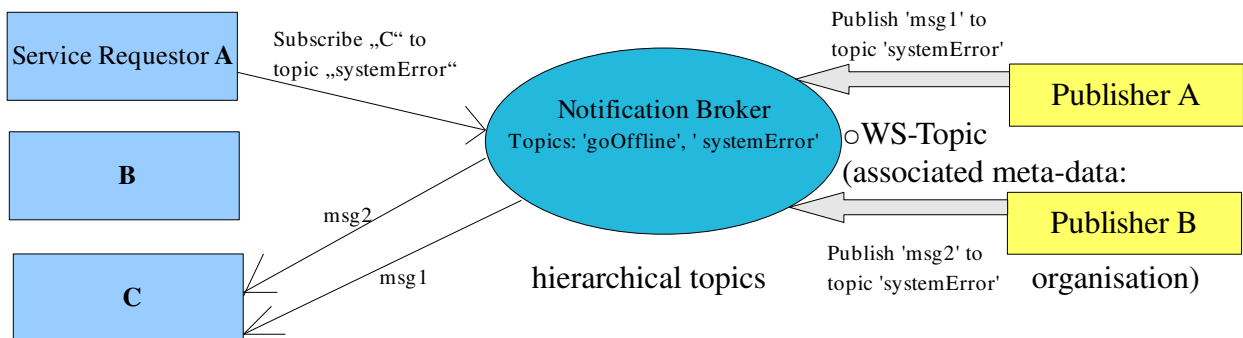
- WS-Resource Properties(RP)
 - specified how to define/access properties
 - defined in XML resource properties document: each element = RP
 - referred to by WSDL 1.1 service *portType*
 - PULL: Query RP document by different query engines
 - PUSH: services use WS-Notification to inform interested parties about changes of their RPs
- WS-Resource Lifetime
 - Basic Mgmt of WS-Resources
 - Resource creation (Factory pattern) → Resource identity
 - Resource destruction (immediate or scheduled destruction of old resources)
 - Soft-State Lifetime mgmt: lifetime extensions possible
 - Example: Job in batch submission system: submit new job as resource creation request → resource creation → job finished → resource destruction
- WS-Service Group
 - information about collection of services/resources represented itself as a resource
 - ServiceGroupRegistration: add group member using WS invocation
 - MembershipContentRules: Restrictions on services that become part of service group
 - WS-Notification on service group changes
 - Example: Resource registries
- WS-Base Faults
 - XML based fault transmission standard associated with WSDL operation
 - with standard data types: Originator, Timestamp, ...
- WS-Renewable Reference
 - implemented by adding policy information to WS-Addressing
 - used to renew/transform stale EPRs (End-Point references) to a fresh one (e.g. resource moved to different host)
- WS-Addressing
 - Route Messages (identify source/destination + authentication method)
 - More than URI : EPR → complex type in WS-Addressing schema
 - Reference URI
 - Reference properties
 - Reference parameters
 - Port type
 - Service Name
 - WS-Policy

- WS-Addressing uses the WS-Resource factory pattern to create EPRs:

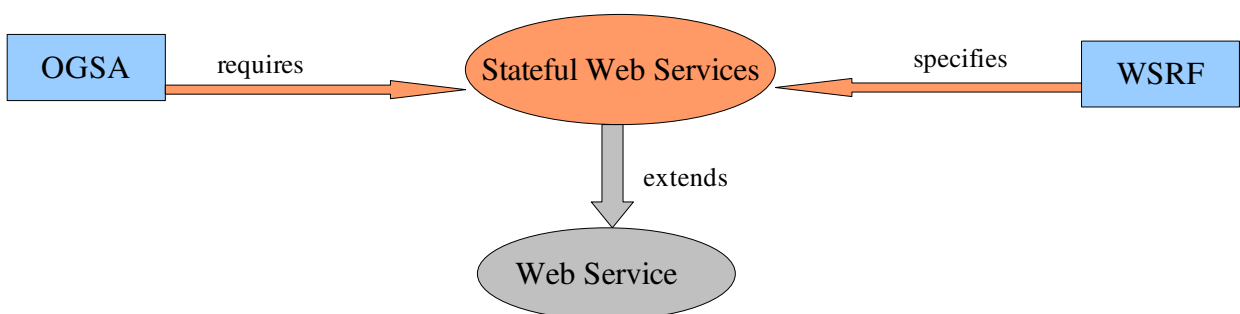


■ WS-Notification

- Notification producers (need NOT be a Web service)
- Notification subscribers (clients = service/subscription requestors) using Xpath expression to issue subscribe request for a specific topic of interest
- 'Subscriptions' are WS-Resources:
 - WS-Base Notifications (roles, concepts, patterns, ...)
 - WS-Brokered Notifications (intermediary Broker):



• OGSA vs. WSRF



- Globus GRAM – Grid Allocation and Management → STATEFUL Job Control

- Resource Allocation
- Process/Job Creation (Factory)
- Process/Job Monitoring
- Process/Job Management
- Maps requests to commands on local schedulers and local computers using Resource Specification Language (RSL)

■ Example using job staging and RFT (Reliable File Transfer) options:

```
<job>
  <executable>/bin/echo</executable>
  <directory>/tmp</directory>
  <argument>Hello</argument>
  <stdout>job.out</stdout>
  <stderr>job.err</stderr>
  <fileStageOut>
    <transfer>
      <sourceUrl>file:///tmp/job.out</sourceUrl>
      <destinationUrl>
        gsiftp://host.domain:2811/tmp/stage.out
      </destinationUrl>
      <rftOptions>
        <subjectName>
          /DC=org/CN=Stuart Martin 564728
        </subjectName>
        <parallelStreams>4</parallelStreams>
      </rftOptions>
    </transfer>
  </fileStageOut>
</job>
```

- Provides Job Submission Model:

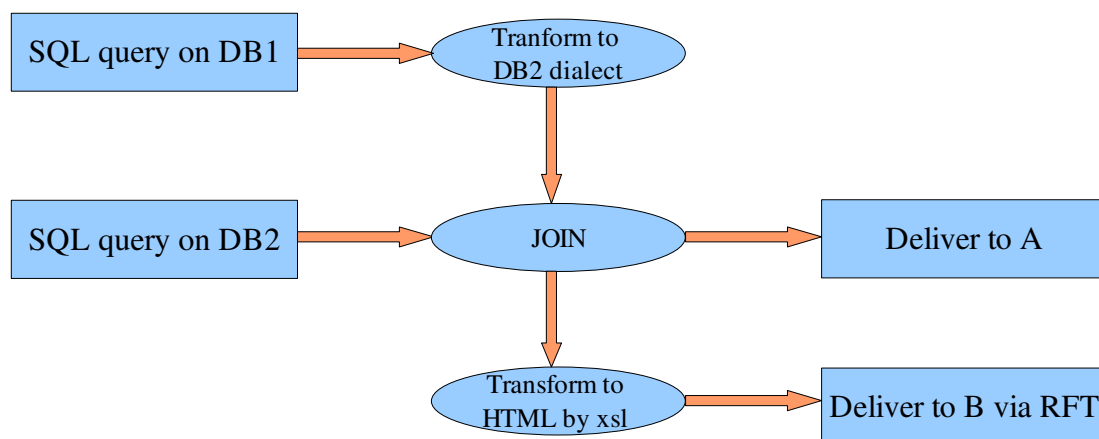
- Create and manage one job on a resource
- Submit and wait
- File based (std in/out/err)
- supported by all batch scheduler (= job schedulers)
- More Complex than RPC:
 - Interception possible (before and after message submission)
 - Complex life-cycle (staging/execution/cleanup states)
 - Job life-cycle monitoring:
 - Scheduler/GRAM states of job: Done, Failed, Active, Suspended, Cleanup, etc.
 - return code
 - Allows asynchronous job monitoring by WS-Notification (vs. Simple query for current state)

- Supports File Staging (Transfer file 'in'/'out' before/after job execution)
 - Supports File Streaming (Monitoring files during job execution)
 - Supports Credential delegation – e.g. security credentials for use by job process (create/refresh/terminate delegations, optional file staging by GRAM)
- Provides Secure Submission model
 - Secure Submit Protocol
 - PKI authentication
 - Authorization and mapping based on GridID
 - Further authorization based on local userID
 - Secure control/cancel (PKI & rights to owned job, not other jobs)
- Provides Secure Execution Model (after authentication)
 - user-account 'sandboxing' of execution processes according to mapping policy and request details
 - Initializing (client delegated or from adapter scripts created) 'sandbox' credentials
 - Multiple levels of audit (container, sudo, local scheduler)
- Example: Vienna Grid Environment (VGE)
 - dynamic negotiation of QoS
 - Virtualize HPC applications as grid services
 - Installation:
 - provide scripts for starting/killing/status querying of jobs
 - provide additional files if necessary (e.g. Cluster submission scripts)
 - VGE service examples:
 - Photodynamics of Retinal (Human vision)
 - Photodynamics of DNA-Base
- Grid Data Management Services
 - Data Movement
 - GridFTP
 - based on FTP (defined by RFCs)
 - Implement standard FTP
 - extensions while preserving interoperability with existing servers
 - Partial file
 - Parallel data vs. Stripped (= several data channels) data
 - Progress Monitoring
 - Buffering settings
 - at least 2 separate socket connections, optionally 2 completely separate processes:

- Control channel/process (commands, responses) owning a
 - Data channel/process (multiple channels in case of a striped server) consisting of
 - Protocol handler
 - DSI – Data Storage Interface
 - ERET/ESTO – Extended Retrieve/Extended Store (manipulate data before transmission) → currently handled via DSI
 - secure, robust, fast, efficient
 - GT4 provides: FTP server & command line client & library for custom clients
- RFT
 - NOT a web service protocol (no WSDL, SOAP, ...)
 - requires open socket connection throughout transfer
 - provides a RFT service using SOAP (and optional notifications) which persists the state of a transfer in a reliable storage
 - Allows third party transfer (= remote file transfer between servers initiated by a client) – GridFTP extends this by security and authentication for the local initiator/client
- Data Replication (keep track of one or more file copies in grid environment, locate files)
 - RLS - Replica Location Service
 - location on physical storage system specified *physical file name*
 - unique identifier for the content of a file specified by *logical file name* used for:
 - Transformations: “I want to search an astronomical database for galaxies with certain characteristics. If a program that performs this analysis exists, I won’t have to write one from scratch.”
 - Derivations: “I want to apply an astronomical analysis program to millions of objects. If the results already exist, I’ll save weeks of computation”
 - user/service registers files when created and can query for replicas
 - distributed indexing of names, fault tolerant update protocols
 - Components:
 - LRC – Local Replica Catalog: a map of logical names to physical names of replicas of those items
 - RLI – Replica Location Index: collects information on LRC mappings stored by one or more LRCs (→ Distributed RLS: every LRC interacts with every RLI)
 - RLI and RLS interaction
 - LRC periodically sends logical name mappings to a set of RLIs using soft-update (→ assures mapping integrity) protocols
 - RLI collects refresh information sent by LRC and updates the periodically timing

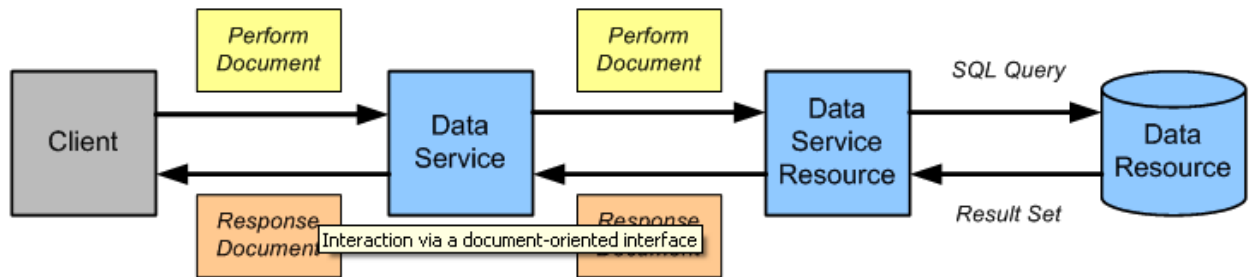
out mappings that are known → RLI responses to querying these mappings

- if RLI fails during an operation, the update information collected makes a reconstruction of the content of the operation possible
- Compression of RLI update traffic with bloom filter: tests if an element is a member of a set → using a bitmap created by a hash function on the logical name of a replicas indicating the members of the LRC
- OGSA DAI → Data Access and Integration middleware to access/integrate data from separate sources via the grid
 - Allows exposure of different types of data sources (XML, DBs, ..) to the grid → Data Integration from various data resources
 - Provides web services to query, update, transform and deliver data
 - Access to data is consistent and data resource-independent incl. optional meta-data
 - Combination of web services to higher level grid services
 - data federation
 - OGSA DAI DQP - distributed query processing: combines result of a distributed query on virtual DB (several physical ones) and delivers a single result to client
 - DQP grid services are realized as mediator over OGSA DAI wrappers
 - parallel processing of queries
 - Provides data storage and analysis via web services
 - Types of DQP services:
 - Coordinators (compile/schedule execution of query)
 - Evaluators (execute query + joining, etc.)
 - Away from handling data sources/structures/transfer/etc. to data processing/analysis
 - Example OGSA DAI Workflow:



- Structure:
 - Data Layer (exposing RDBs, XML-DBs, Files via OGSA DAI)
 - Business layer

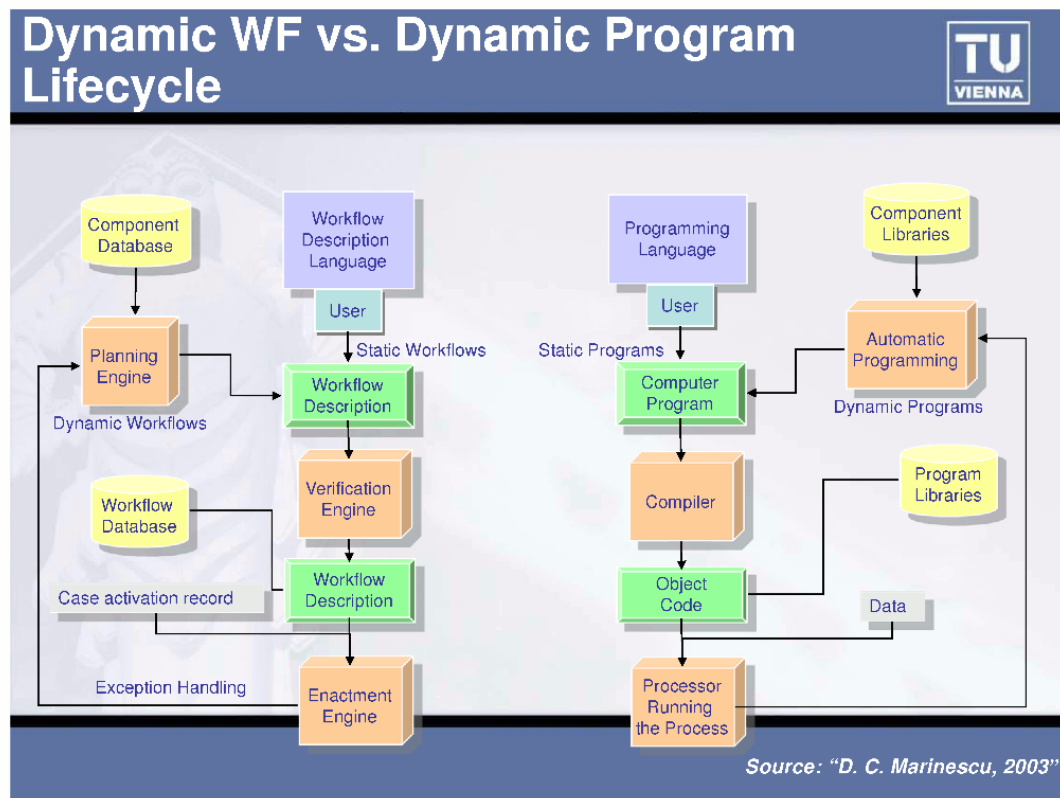
- Provides Business Logic Layer interface
 - bidirectional communication btw. Data and Business Layer
 - provides so called *Data Resource Accessors*, controlling access to the underlying *Data Resource Service*
- Executes perform documents (abstracts interactions with data resource service as activities(=workflow units with specific name)to be performed on behalf of client → SQL queries, XSL transformations, GridFTP delivery) providing NO loops, conditionals, variables, but support flow of data from one activity to the other
- Generate response documents (status of executes, may contain result set)
- Data transport (streaming from/to data service resources from/to clients or data service resources)
- Session Mgmt (state across multiple requests can be maintained)
- Property Mgmt (creation/access/removal of data service RPs)



- Presentation Layer (Data Services: WSRF Data Service & WSIF [Web Service Invocation Framework] Data Service)
- Client Layer (Client application & toolkit providing WSRF or WSI stubs)

Grid Workflows (WF) and QoS

- Creation of complex grid services: automation of execution of several grid applications/services, while defining the order of execution and the data flow and control structures btw. Services
- Originating from scripted based programming (e.g. Pegasus based on scripting) with good performance, but it can be confusing, hard to maintain, limited in expressivity.
-



- Grid WF Taxonomy:
 - WF composition systems
 - WF QoS constraints
 - Information Retrieval
 - WF Scheduling
 - Architecture (centralized, hierarchical, de-centralized)
 - Decision Making (local & global)
 - Planning scheme
- Grid WF Applications
 - PEGASUS - Planning of Execution in Grids
 - maps abstract WF to concrete WF
 - DAGMan guides this mapping
 - Scheduling
 - Decision Making based on
 - TF - transformation catalog: has map of logical transformation names (used to query TC) to physical transformation resources
 - MDF – (Global) Monitoring and discovery service (load, scheduler, disc space, GridFTP)

- Workflow reduction (RLS & RLI → resource might be available already)
- WF execution by Condor-G
 - Queuing service
 - Credential Mgmt
 - Authentication
 - File Transferring
 - Resource description
 - fault tolerance
- Supports WF partitioning
- used in bioinformatics/astronomy/physics
- Triana
 - is PSE – Problem Solving Environment
 - dynamic VOs
 - Peer federations for distributed resources
 - distributed networks (advertise/discovery/communicate)
 - uses GAP- Grid Application Platform Bindings to p2p hosts, web services and OGSA services:
 - JXTA → P2P discovery/communication
 - P2PS (Simplified P2P incl. P2P Advertisement)
 - UDDI registry & WSIF
 - OGSA
- QoS (aware) Grid WF Engine
 - QoS-aware (constraints-aware) WF life-cycle
 - WF scheduling by coordinating tasks and using 'cheapest' services according to WF constraints
 - WF Optimization
 - Analytic (linear programming – suitable for simple WFs with linear constraints and small search spaces)
 - Integer programming
 - Simplex Algorithm
 - AI (suitable for complex WFs with non-linear constraints and large search spaces)
 - Simulated Annealing (approximate to global minimum of a given function in large search space by gradually reducing the 'temperature' T and selecting near-by minimum based on the difference of current function value and T)
 - Genetic Algorithms
 - WF Location affinity
 - Grid site (security, law, performance reasons)
 - Organisation (sensitive business data → hide from competitors)
 - Geographical region (legal requirements, demographic info for medical studies, electronic medical data transfer)
 - High level UML Modelling (QoWL – Quality of Working Life - Modelling)
 - Visual modelling language (improves difficult XML-Based WF Specification)
 - stereotypes & tagged values
 - UML-Based DSL – Domain Specific Language
 - Activity Diagrams