Introduction to Grid Application Development

Vangelis Floros
(efloros@cern.ch)
Application Support Team
NCSR "Demokritos", Institute of Nuclear Physics

www.eu-egee.org

Information Society

Acknowledgments

- **Portion of slides (derived from those) prepared by:**
  - Mike Mineter, NESC
  - Charles Loomis, LAL-Orsay
  - Roberto Barbera and his GILDA team
    University of Catania and INFN
  - EGEE-II NA4 Activity Member's

Contents

- Basic Concepts
- Types of Grid Applications
- Challenges
- Crash introduction to gLite services

- Working from the command line

- Introduction to the basic APIs

Definition, Ian Foster

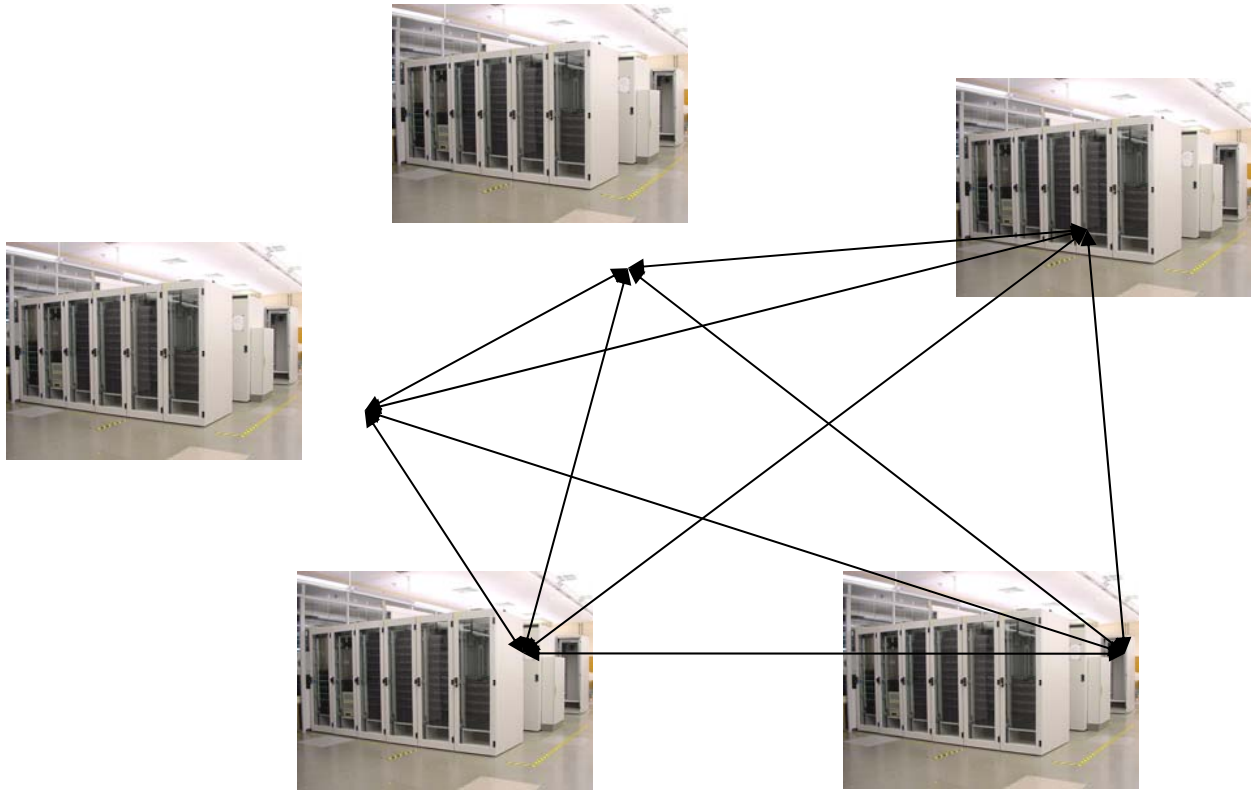
"Grid computing is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations" (I.Foster)

A *Virtual Organisation* is:

- **People from different institutions working to solve a common goal**
- **Sharing distributed processing and data resources**

Focus: Wide area, collaboration, virtual organisations

Practical definition in some areas

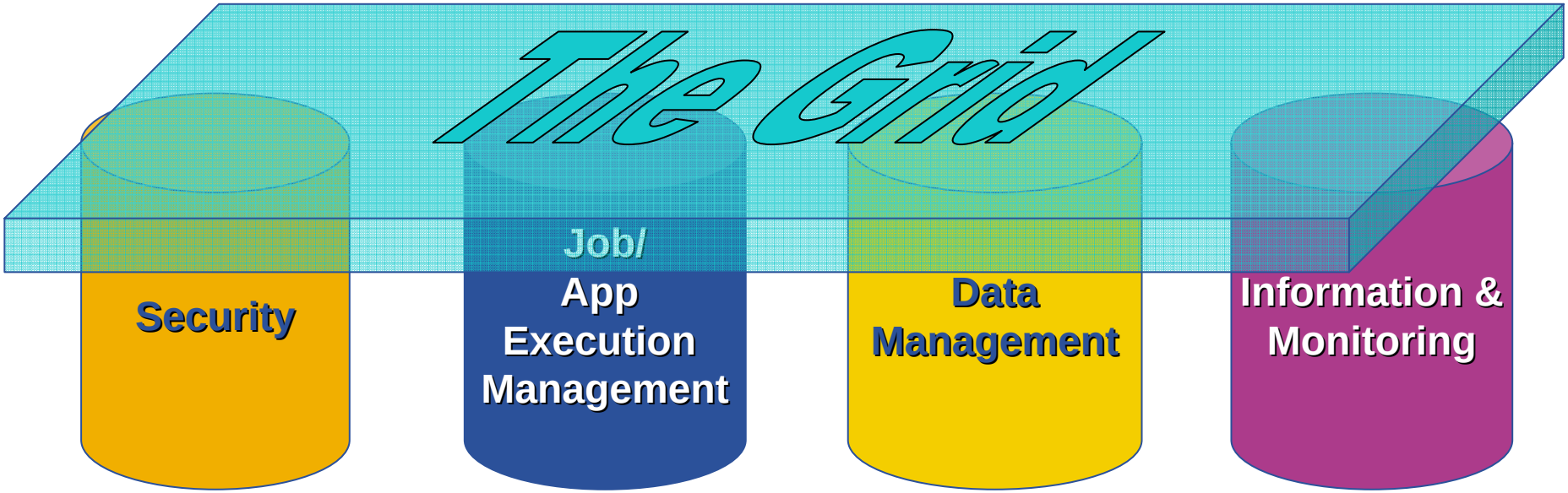


Grid Computing ==

Clustering Clusters; Building a global batch submission system ...

- **Basic Concepts**
- **Types of Grid Applications**
- **Challenges**
- **Crash introduction to gLite services**

The four pillars of Grid Computing



- Authentication
- Authorization
- Confidentiality
- Integrity
- VO management

- Remote execution
- Load balancing
- Interactivity
- Parallelism
- Workflows

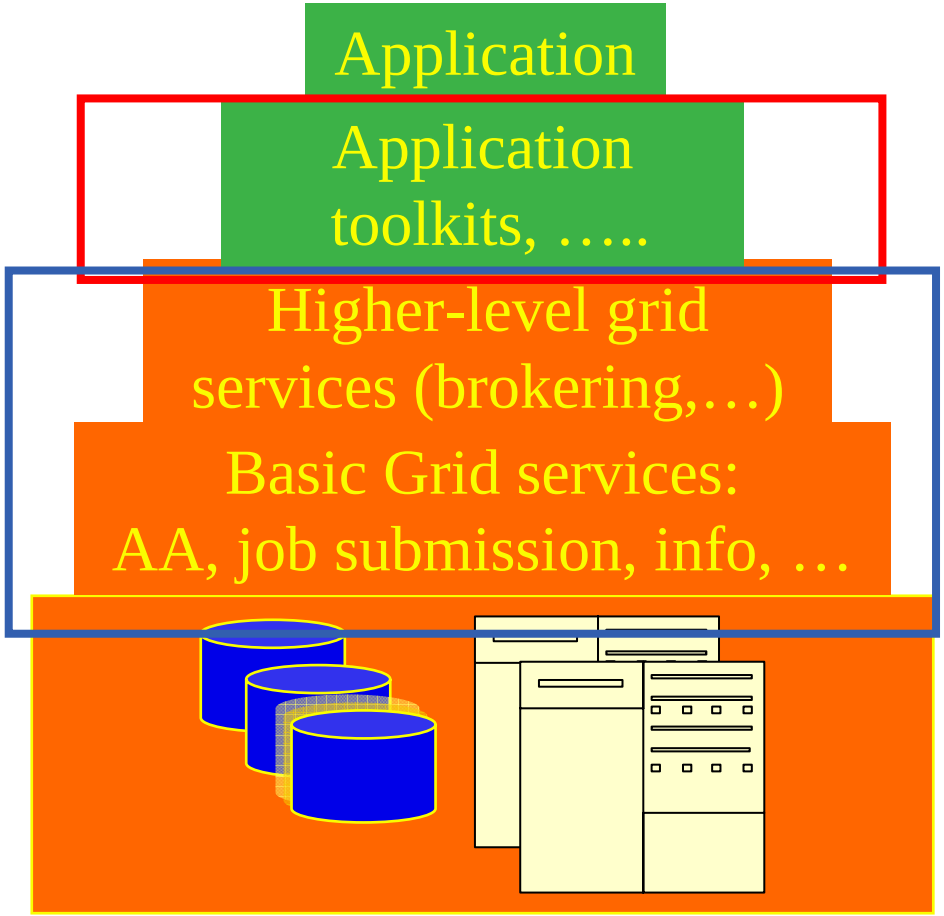
- Data staging
- Bulk data transfers
- Replication
- Metadata

- Resource discovery
- Events & Notifications
- Resource status & monitoring

Grid Application Development

- Application development in the Grid implies the exploitation of APIs, tools and environments that provide the four basic Grid capabilities order to perform complex tasks and achieve diverse goals.
- The extend and approach that the four basic Grid concepts are materialized depends on the specific capabilities of the Grid enabling technologies (in our case the gLite middleware suite)

The vital layer



Where computer science meets the application communities!

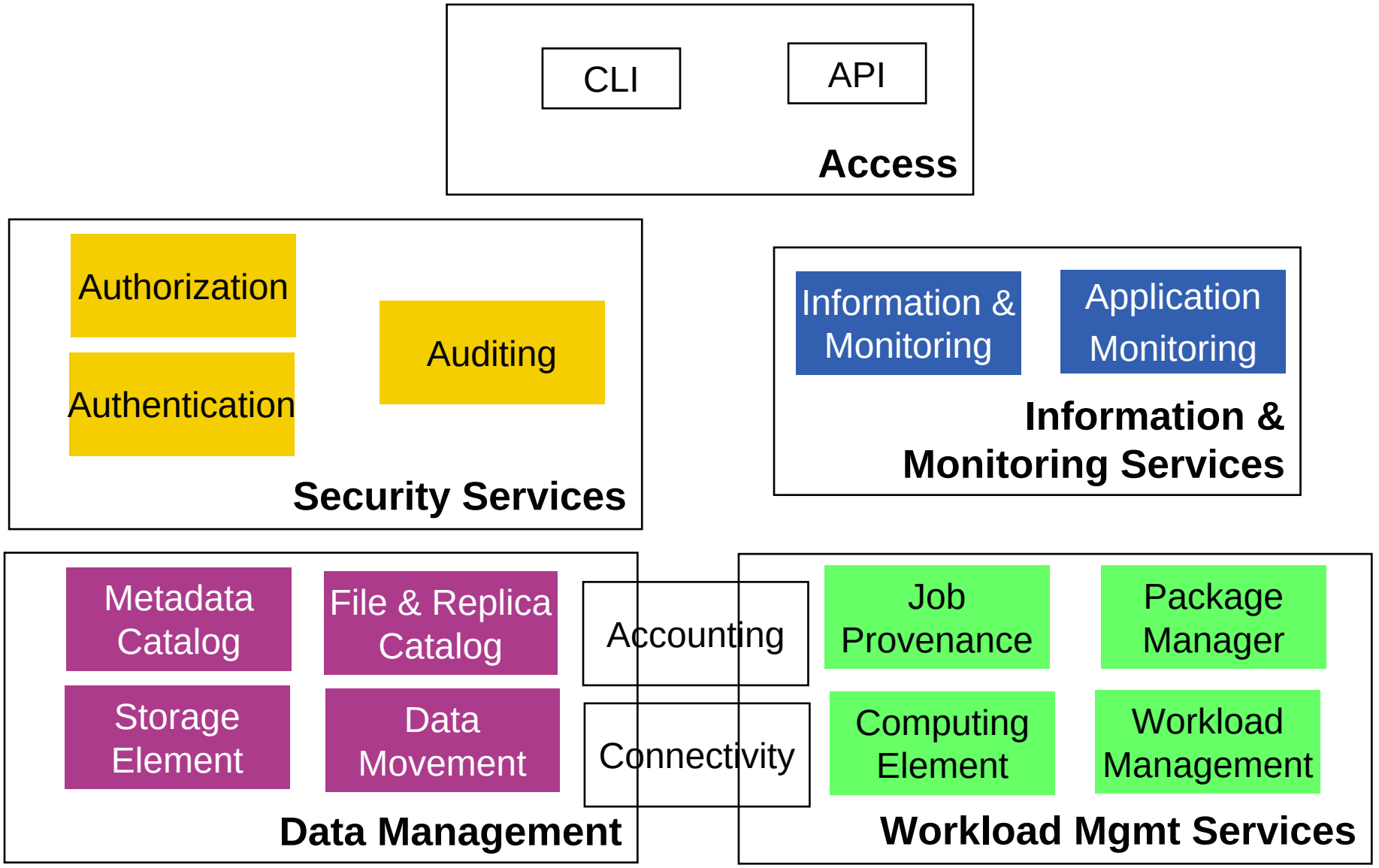
VO-specific developments built on higher-level tools and core services

Makes Grid services useable by non-specialists

Grids provide the compute and data storage resources

Production grids provide these core services.

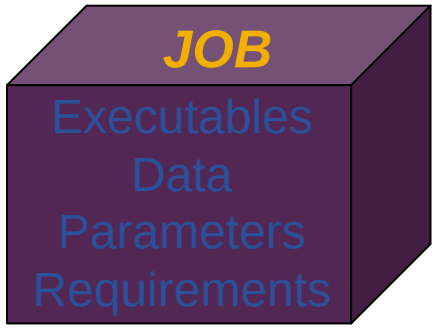
gLite Grid Middleware Services



The Job concept

- **gLite follows the job submission concept for application execution and resource sharing**

- **A job is a self contained entity that packages and conveys all the required information and artifacts for the successful remote execution of an application.**
  - Executable files
  - Input/Output data
  - Parameters
  - Environment
  - Infrastructure Requirements
  - Workflows



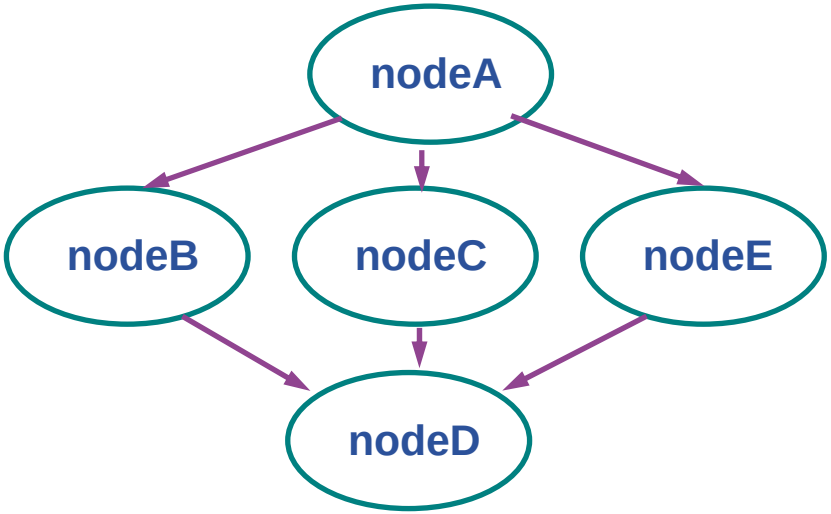
- **Described using the Job Description Language (JDL)**

Complexities of grid jobs

1. **Simple jobs – submitted to WMS to run in batch mode**
2. **Job invokes grid services**
   - To read & write files on SE
   - Monitoring
   - For outbound connectivity (interactive jobs)
   - To manage metadata
   - …
3. **Complex jobs**
   - An environment controls multiple jobs on users' behalf
     - High-level services
     - Portals with workflow
     - Software written for the VO (or by the user)
     - …

Complex Workflows

- **Direct Acyclic Graph (DAG) is a set of jobs where the input, output, or execution of one or more jobs depends on one or more other jobs**
- **A Collection is a group of jobs with no dependencies**
  - basically a collection of JDL's



- **A Parametric job is a job having one or more attributes in the JDL that vary their values according to parameters**
- **Using compound jobs it is possible to have one shot submission of a (possibly very large, up to thousands) group of jobs**
  - Submission time reduction
    - Single call to WMProxy server
    - Single Authentication and Authorization process
    - Sharing of files between jobs
  - Availability of both a single Job Id to manage the group as a whole and an Id for each single job in the group

Pragmatic approach to Grid application development

- Benefits and Restrictions.
- Potential compromises

- Resource sharing (no dedicated resources)
- Explicit and implicit collaboration (working in shared environment)
- Security risks (yes there are!)
- Performance compromises (wrt system responsiveness. Some times too much middleware!)
- Application Models (the application may have to adapt to the grid and not vice versa)

- Basic Concepts
- Types of Grid Applications
- Challenges
- Crash introduction to gLite services

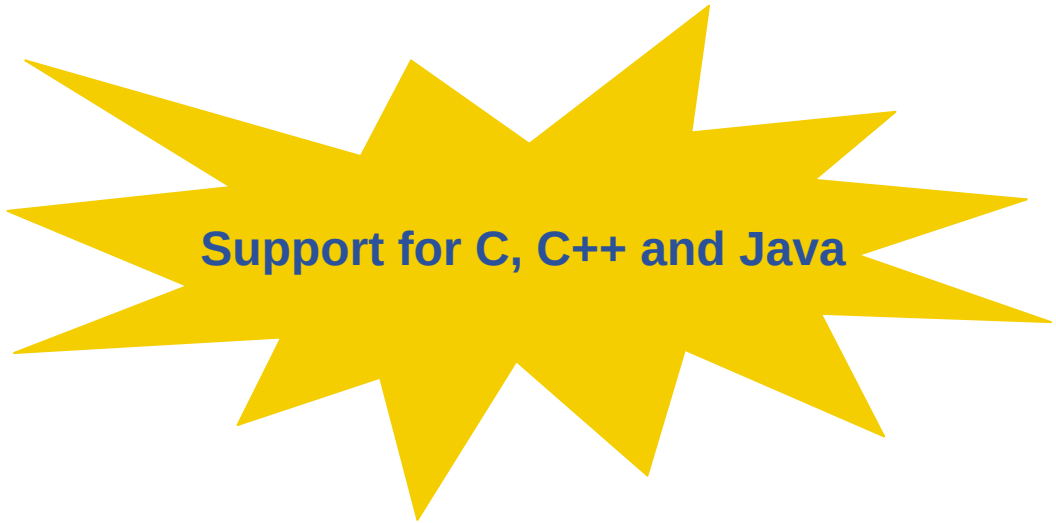
Gridification Levels

- **No development**. Wrap existing applications as jobs. No source code modification is required
- **Minor modifications**. The application exposes minimal interaction with the grid services (e.g. Data Managements)
- **Major modifications**. A wide portion of the code is re-written to adopt to the new environment (e.g. parallelization, metadata, information)
- **Pure grid applications**. Developed from scratch. Extensively exploit existing grid services to provide new capabilities customized for a specific domain (e.g. metadata, job management, credential management)

Application Programming Interfaces

An **Application Programming Interface** (**API**) is the interface that a computer system, library or application provides in order to allow requests for services to be made of it by other computer programs, and/or to allow data to be exchanged between them.

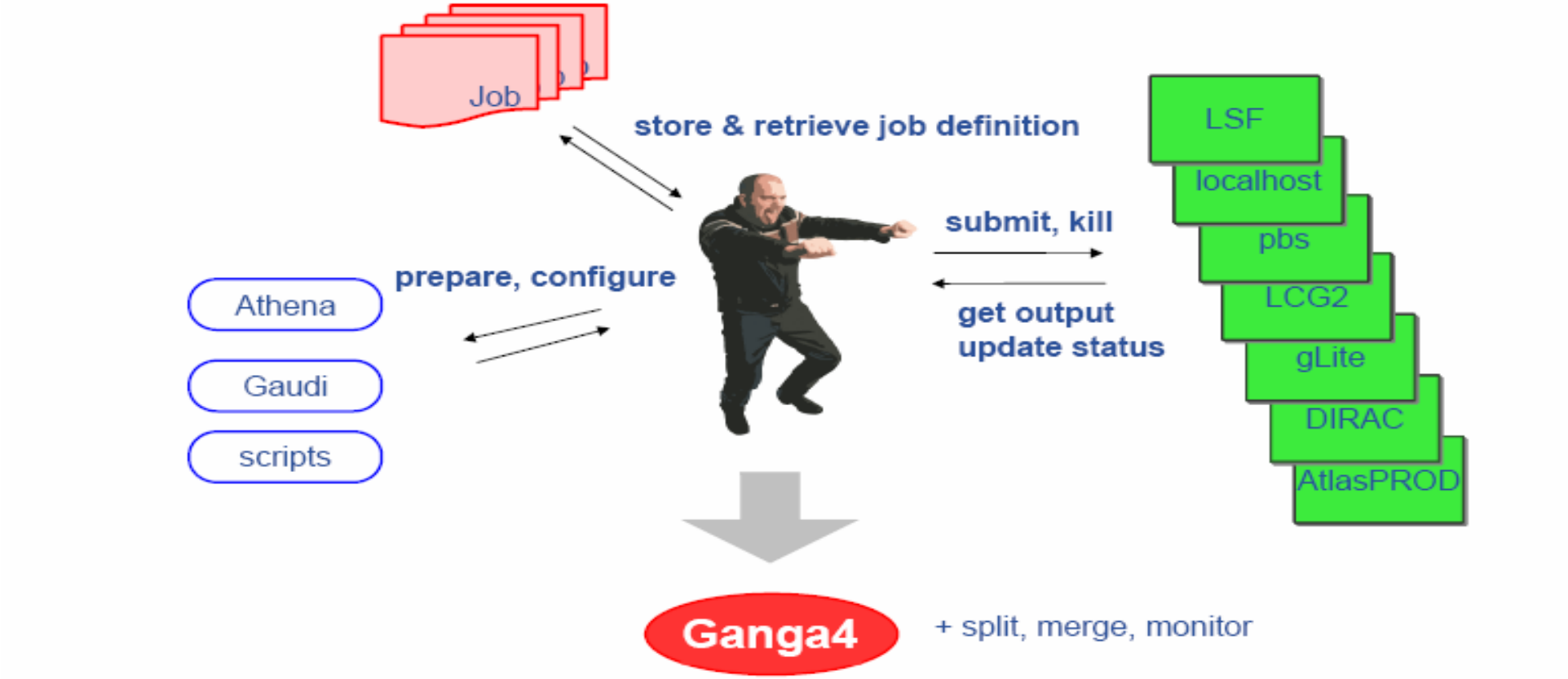
- **Classic APIs**
  - Static compilation
  - Shared libraries
  - Need access to static or dynamic libraries
- **Web Service Interfaces**
  - The programmer may generate Web Service stubs and develop new clients from scratch
  - Libraries are comprised of precompiled Service clients
  - Need access to Web Services WSDL



Invocation of applications

- **From the UI**
  - Command Line Interfaces / Scripts
  - APIs
  - Higher level tools
- **From desktop Windows applications**
  - Use Grids without awareness of them!
  - But gLite not (yet) supporting Windows
- **From portals**
  - For recurring tasks: “core grid services” as well as application layer
  - Accessible from any browser
  - Tailored to applications
  - In EGEE: P-GRADE and GENIUS

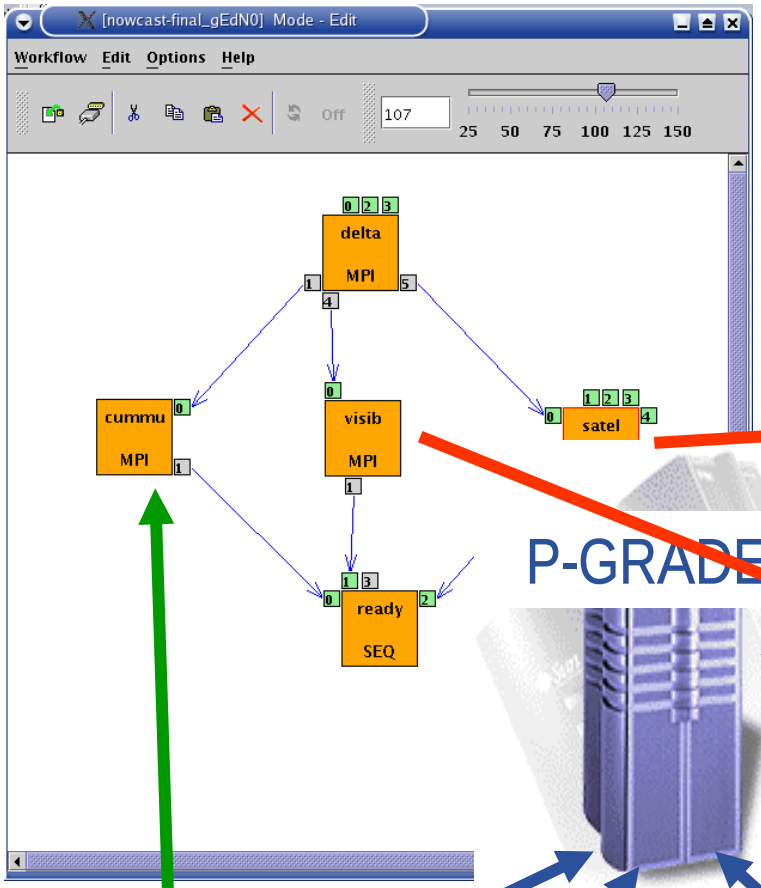
Example of higher-level tools: GANGA



- **Ganga is a lightweight user tool**
  ganga.web.cern.ch/
- **But also: Ganga is a developer framework**

GENIUS

- **For many application communities**
  - Interface can be tailored for specific requirements
- **For demonstration purposes**
  - https://glite-demo.ct.infn.it/
    - Available for anyone to use
  - https://glite-tutor.ct.infn.it/
    - Fuller functionality for users who have stored long-lived proxy in MyProxy server



Different jobs of a workflow can be executed in different grids

EGEE Grid e.g. VOCE

P-GRADE-Portal

The portal can be connected to multiple grids

UK NGS

London

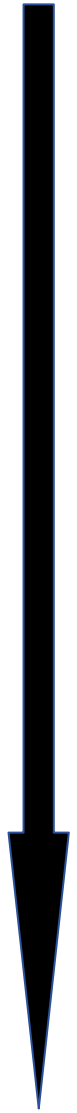
Rome

Athens

- Basic Concepts
- Types of Grid Applications
- Challenges
- Crash introduction to gLite services

Different Goals for App. Development

- **I need resources for my research**
  - I need richer functionality
    - MPI, parametric sweeps,…
    - Data and compute services together…
- **I provide an application for (y)our research**
  - How!?
    - Pre-install executables ?
    - Hosting environment?
    - Share data
    - Use it via portal?
- ***We* provide applications for (y)our research**
  - Also need:
    - Coordination of development
    - Standards
    - …



Engineering challenges increasing

Challenges

- **Research software is often**
  - Created for one user: the developer
  - Familiarity makes it useable
  - Short-term goals: Used until papers are written and then discarded
- **Grid applications are often used**
  - by a VO
  - Without support from developer
  - In new contexts and workflows
- **Grid application developers are**
  - In a research environment
  - Yet their s/w must have:
    - Stability
    - Documentation
    - Useability
    - Extendability
  - i.e. Production quality

Need expertise in:

- **software engineering**
- **application domain**
- **grid computing**

Consequences

- **Team work!**
- **Engaged in world-wide initiatives – reuse, don't make your own! Cross disciplines for solutions.**
- **From research to production software: ~5 times the effort.**
  - "80% of the time for last 10% of the functionality & reliability"
- **Standardisation is key**
  - For re-use, for dynamic configuration of services,..
  - Both for middleware and domain specific (e.g. GEON)
- **Need to follow a deliberate development process**
  - Waterfall? Rapid prototyping?
  - Requirements engineering, design, implementation, validation, deployment
  - Engaged with the user community

- Basic Concepts
- Types of Grid Applications
- Challenges
- Crash introduction to gLite services

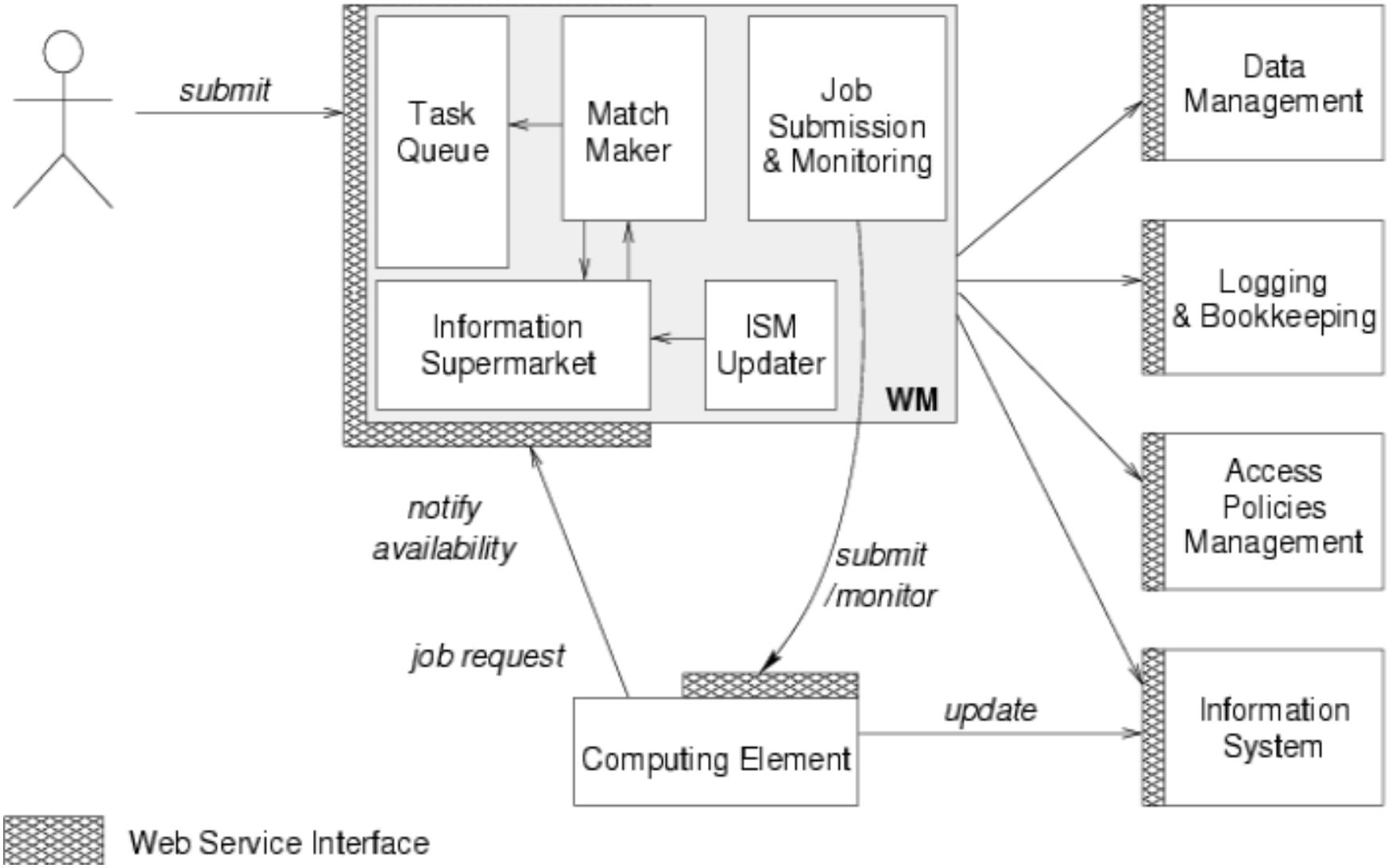
More about gLite services

- **gLite 3.0 Workload Management**
- **Accessing data on SEs**
  - Can have massive files, too big to copy
  - How to access these?
- **Management of metadata**
  - May have many thousands of files
  - Need to access and re-use based on characteristics… more than by their logical file names.
- **Monitoring of applications**
  - May be running many long jobs
  - What's happening?!

Workload Management System

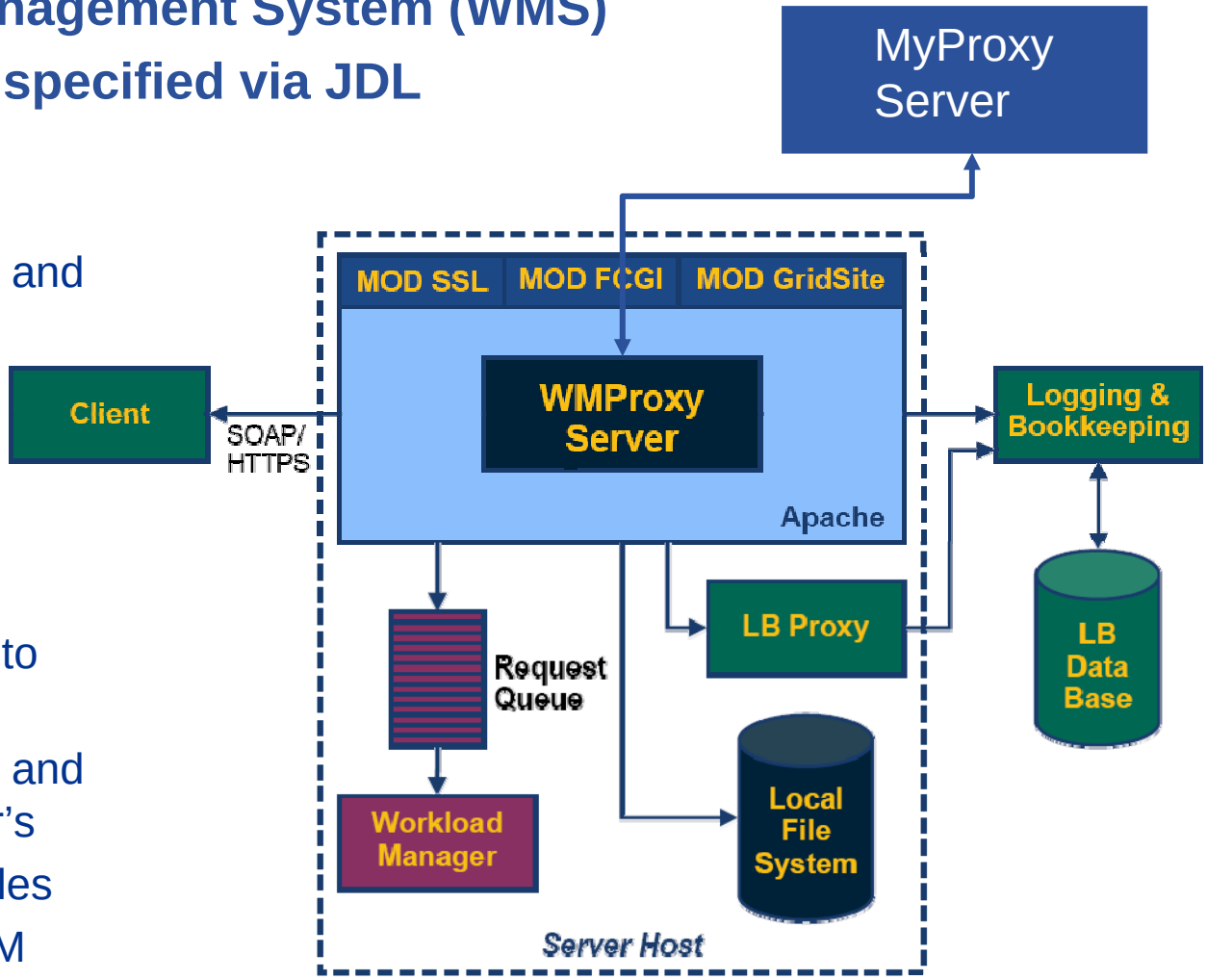
- **Helps the user accessing computing resources**
  - resource brokering
  - management of input and output
  - management of complex workflows
- **Support for MPI job even if the file system is not shared between CE and Worker Nodes (WN) – easy JDL extensions**
- **Web Service interface via WMProxy**

Workload Management System



WMProxy

- **WMProxy is a SOAP Web service providing access to the Workload Management System (WMS)**
- **Job characteristics specified via JDL**
  - jobRegister
    - create id
    - map to local user and create job dir
    - register to L&B
    - return id to user
  - input files transfer
  - jobStart
    - register sub-jobs to L&B
    - map to local user and create sub-job dir's
    - unpack sub-job files
    - deliver jobs to WM

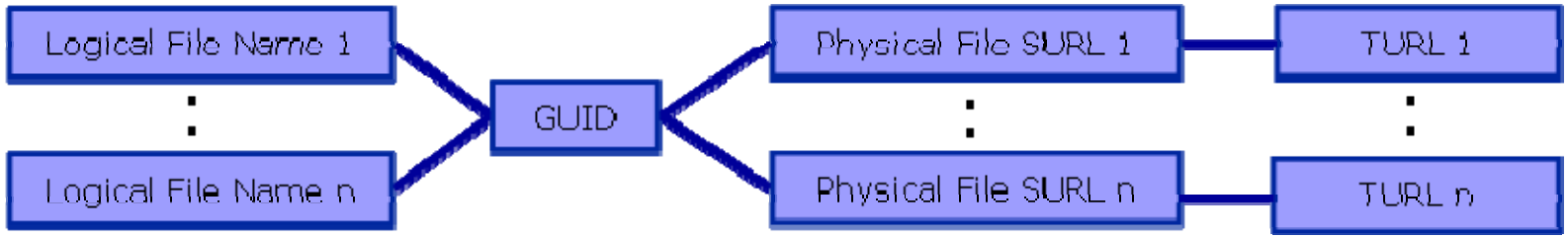


Data Management

- **User and programs produce and require data**
- **Data may be stored in Grid datasets (files)**
  - Located in Storage Elements (SEs)
  - Accessed/Transferred using GSIFTP
  - Several replicas of one file in different sites
  - Accessible by Grid users and applications from “anywhere”
  - Locatable by the WMS (data requirements in JDL)
- **Also…**
  - WMS can send (small amounts of) data to/from jobs: Input and Output Sandbox
  - Data may be copied from/to local filesystems (WNs, UIs) to the Grid

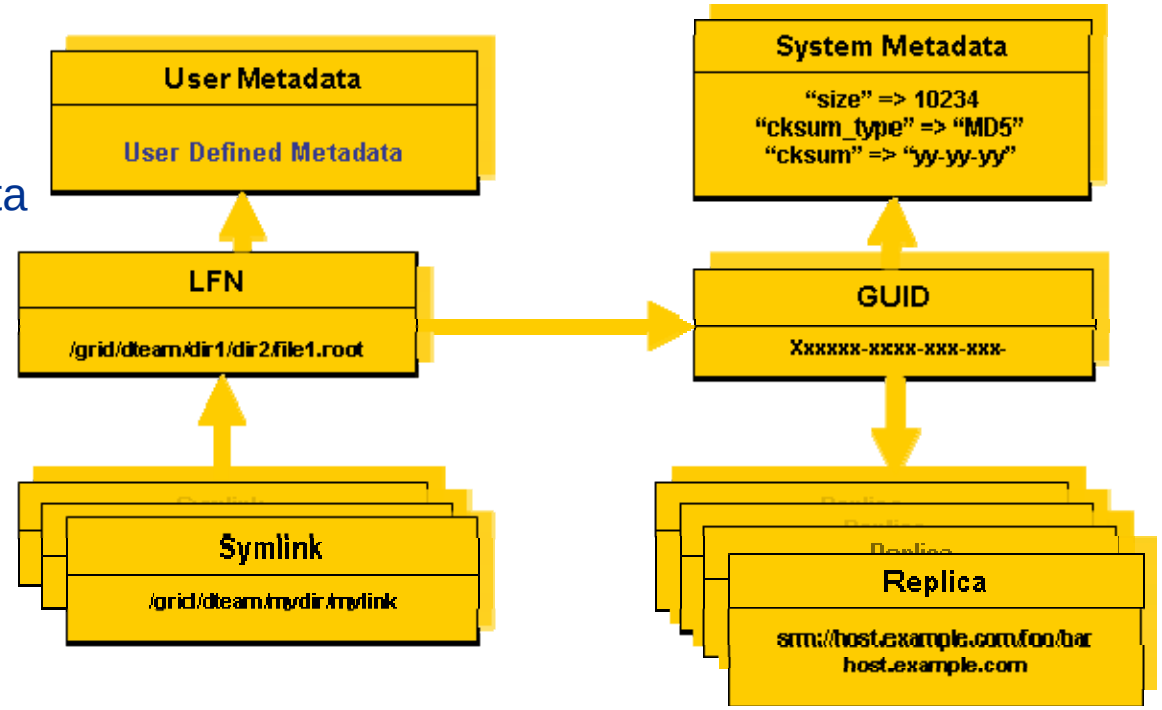
Name conventions

- **Logical File Name (LFN)**
  - An alias created by a user to refer to some item of data, e.g.
    "lfn:cms/20030203/run2/track1"
- **Globally Unique Identifier (GUID)**
  - A non-human-readable unique identifier for an item of data, e.g.
    "guid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
- **Site URL (SURL) (or Physical File Name (PFN) or Site FN)**
  - The location of an actual piece of data on a storage system, e.g.
    "srm://pcrd24.cern.ch/flatfiles/cms/output10_1" (SRM)
    "sfn://lxshare0209.cern.ch/data/alice/ntuples.dat" (Classic SE)
- **Transport URL (TURL)**
  - Temporary locator of a replica + access protocol: understood by a SE, e.g.
    "gsiftp://lxshare0209.cern.ch//data/alice/ntuples.dat"



The Logical File Catalog

- **Manages the identification, sharing and replication of data in the gLite Grid.**
- **LFN acts as main key in the database. It has:**
  - Symbolic links to it (additional LFNs)
  - Unique Identifier (GUID)
  - System metadata
  - Information on replicas
  - One field of user metadata



More to come

- **Security (VOMS)**
- **Metadata (AMGA)**
- **Monitoring (R-GMA)**
- **…**

Questions?