Architecture of gLite WMS

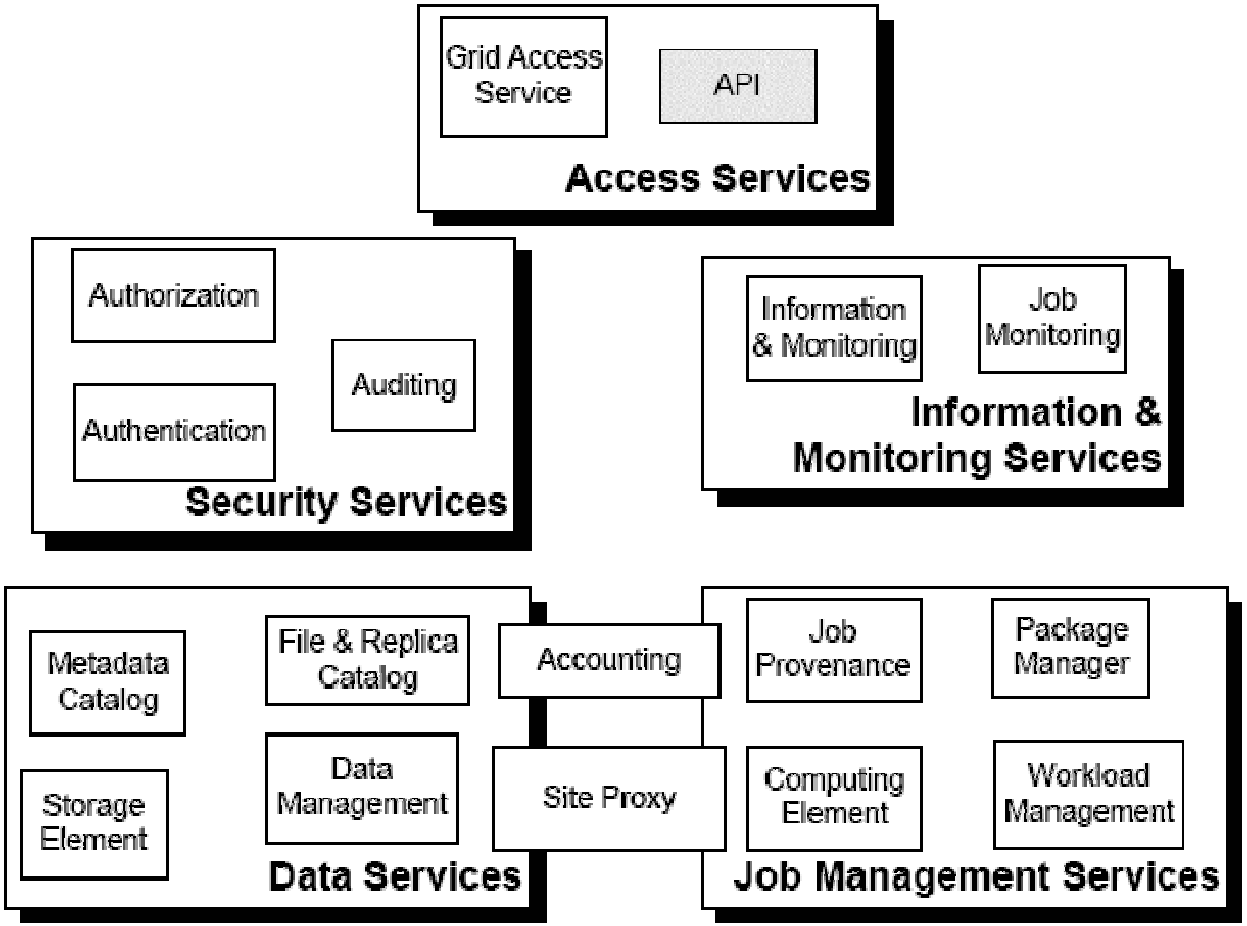
Salvatore Monforte
Marco Pappalardo
INFN

First gLite tutorial on GILDA, Catania, 13-15.06.2005

Overview of EGEE Middleware

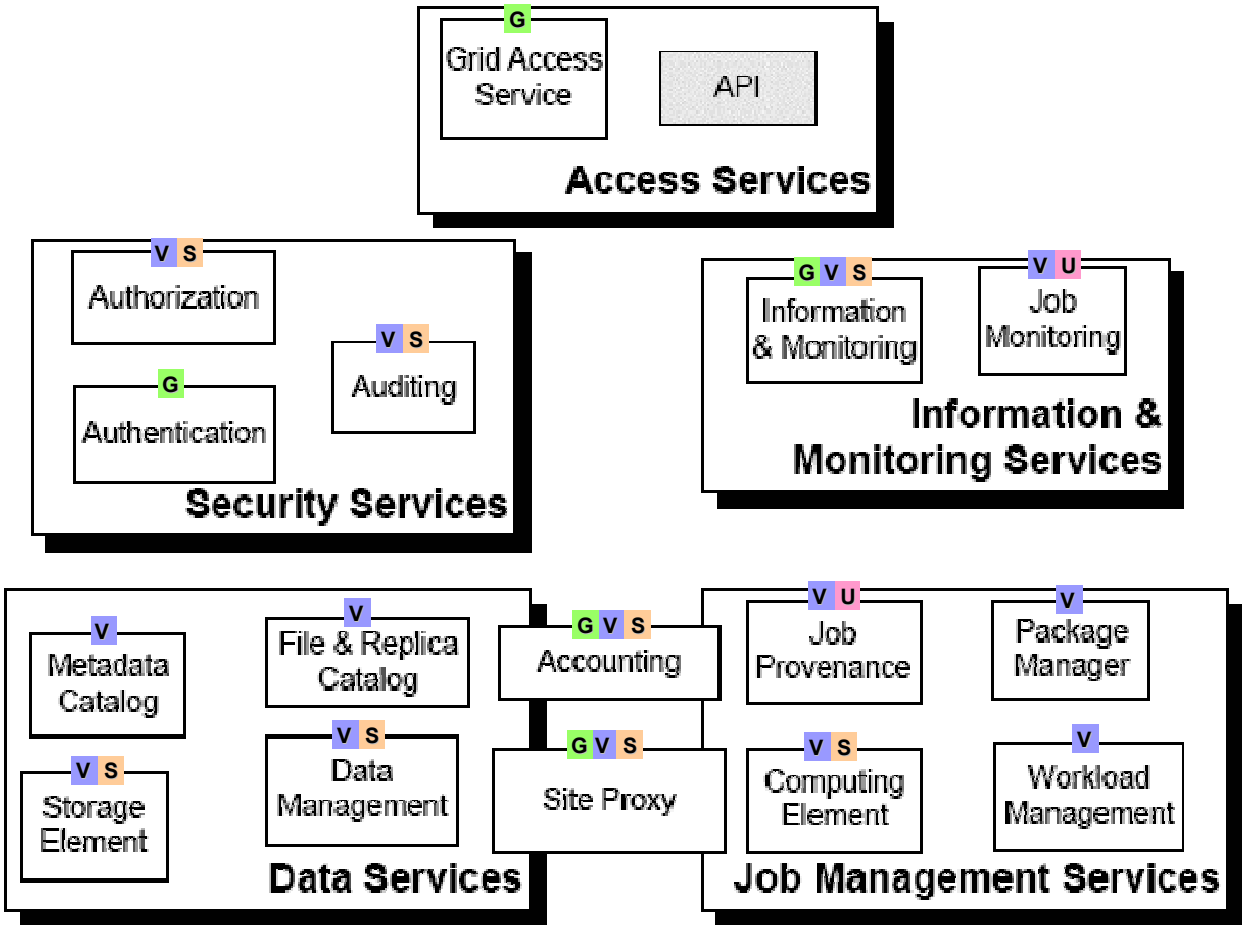
- The gLite Grid services follow a *Service Oriented Architecture*
  - **facilitate interoperability among Grid services**
  - **allow easier compliance with upcoming standards**
- Architecture is not bound to specific implementations
  - **services are expected to work together**
  - **services can be deployed and used independently**
- The gLite service decomposition has been largely influenced by the work performed in the LCG project

Overview of EGEE Middleware



- The gLite services are characterised by the scopes and enforcement of their policies: ***user***, ***site***, ***VO*** *and* ***global*** *(i.e. multi-vo)*

Overview of EGEE Middleware



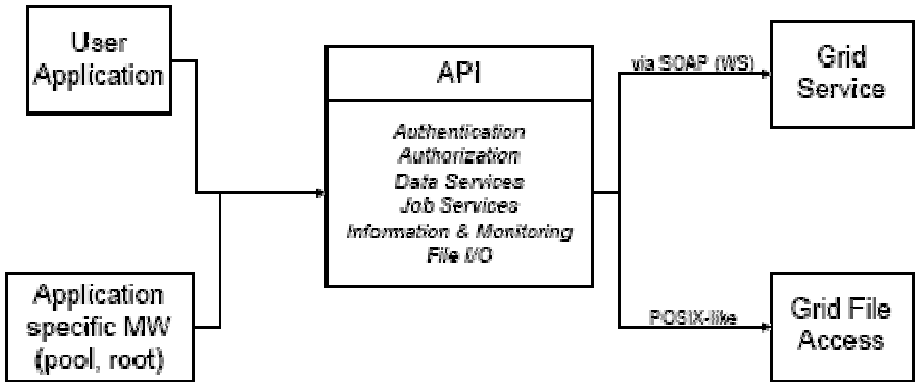
- Most services are managed by a VO
  - **independent service instances per VO**
  - **service instances will in most cases serve multiple VOs**
    - **performance**
    - **scalability**

Overview of EGEE Middleware

- **Security services**
  - **Authentication, Authorization, and Auditing**
    - **identification of entities (users, systems, and services)**
    - **allow or deny access to services and resources**
    - **provide information for post-mortem analysis of security related events.**
  - **Data confidentiality and a Site Proxy**
    - **control network access patterns of applications and Grid services utilising its resources.**

Overview of EGEE Middleware

- **Grid Access Service**
  - **common framework for gaining access to the Grid services.**
  - **manage the life-cycle of the Grid services available to a user, according to his/her privileges.**
- API library
  - **client applications, graphical user interfaces or even Grid Web portals**
  - **authenticate users, submit jobs, inquire job status and manage jobs, access the files available, put files**



Overview of EGEE Middleware

- **Information and Monitoring Services**
  - **provide a mechanism to publish and consume information concernig grid resources**
- The information system relies upon registering:
  - **the location of publishers**
  - **the subset of the total information**
- Consumers may issue queries to the information system while not having to know where the information was published
- Finally there is a fine-grained, rule-based, authorization scheme to ensure that people can only read or write within their authority.

Overview of EGEE Middleware

- **Data Services**
  - **Storage Element**
  - **Catalog Services**
  - **Management**

- Closely related to the data services are
  - **the security-relatedservices**
  - **the Package Manager**
  -
- The granularity of the data is on the file level.
  - **generic enough to be extended to other levels of granularity.**
    - **Data sets or collections are a very common extension**
      - ***information about which files belong to a dataset maybe kept in an application metadata catalog.***

- EGEE data services users are supplied with the abstraction of a global file system.
  - **A client user application may look like a Unix shell (as in AliEn) which can seamlessly navigate this virtual file system, listing files, changing directories, etc.**

- The data in the files can be accessed through the Storage Element (SE).
- The access to the files is controlled by Access Control Lists (ACL)..

Overview of EGEE Middleware

- **Job Management Services**
  - **main services related to job management/execution are**
    - **computing element**
      - ***job management (job submission, job control, etc.), but it must also provide***
      - ***provision of information about its characteristics and status***
    - **workload management**
      - ***core component discussed in details***
    - **accounting**
      - ***special case as it will eventually take into account***
        - **computing, storage and network resources**
    - **job provenance**
      - ***keep track of the definition of submitted jobs, execution conditions and environment, and important points of the job life cycle for a long period***
        - **debugging, post-mortem analysis, comparison of job execution**
    - **package manager**
      - ***automates the process of installing, upgrading, configuring, and removing software packages from a shared area on a grid site.***
        - **extension of a traditional package management system to a Grid**
- Services communicate with each other as the job request progresses through the system
  - **a consistent view of the status of the job is maintained**

Workload Management System

- Workload Management System (WMS) comprises a set of Grid middleware components responsible for distribution and management of tasks across Grid resources
  - **applications are conveniently, efficiently and effectively executed.**

- Comparable services from other grid projects are, among others, the EDG WMS, Condor and the Eurogrid-Unicore resource broker.

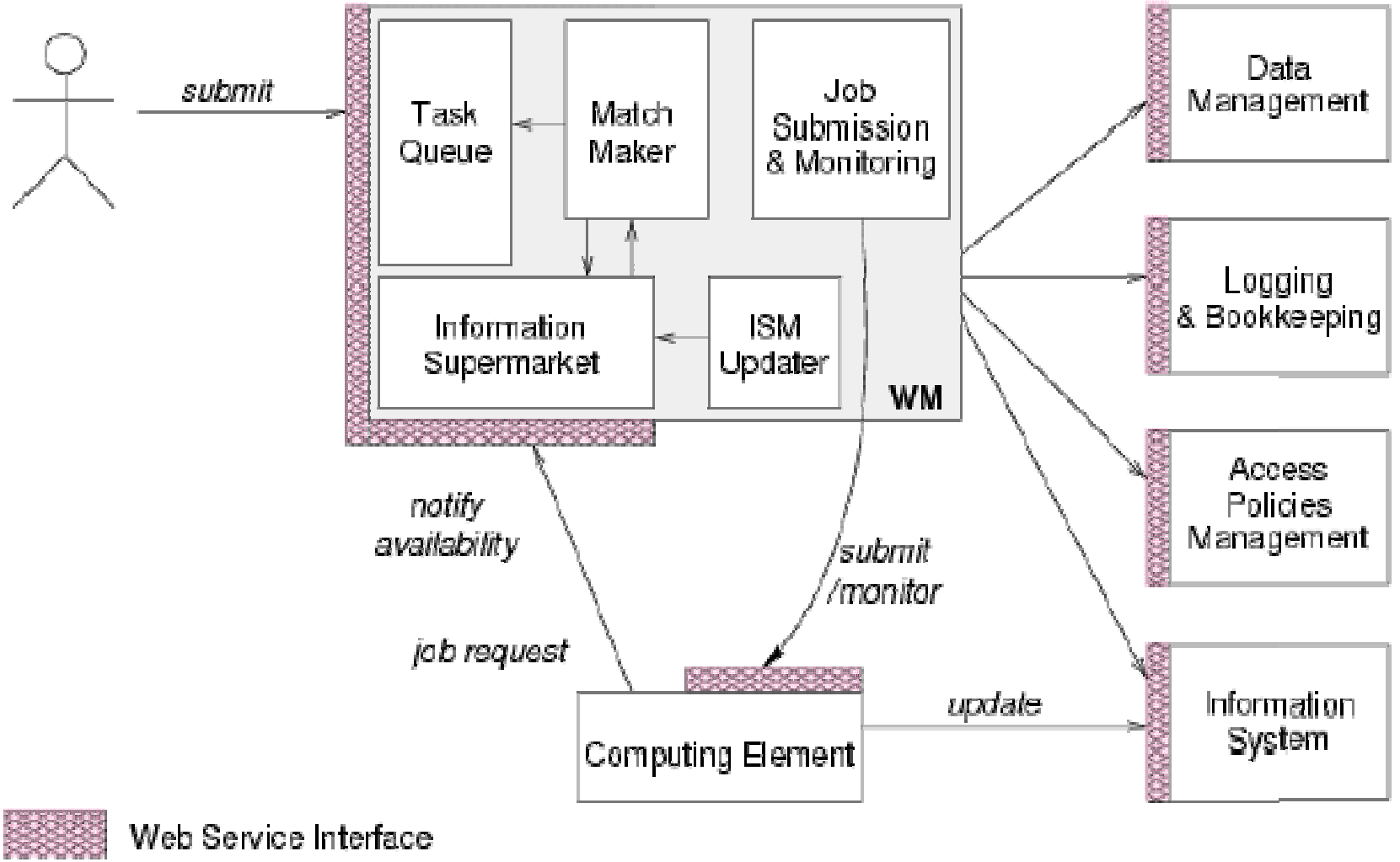
Workload Management System

- Purpose of Workload Manager (WM) is accept and satisfy requests for job management coming from its clients
  - **meaning of the submission request is to pass the responsibility of the job to the WM.**
    - **WM will pass the job to an appropriate CE for execution**
      - ***taking into account requirements and the preferences expressed in the job description***
- The decision of which resource should be used is the outcome of a *matchmaking* process between submission requests and available resources
  - **availability of resources for a particular task depends**
    - **on the state of the resources**
    - **on the utilisation policies**
      - ***assigned for the VO the user belogs***

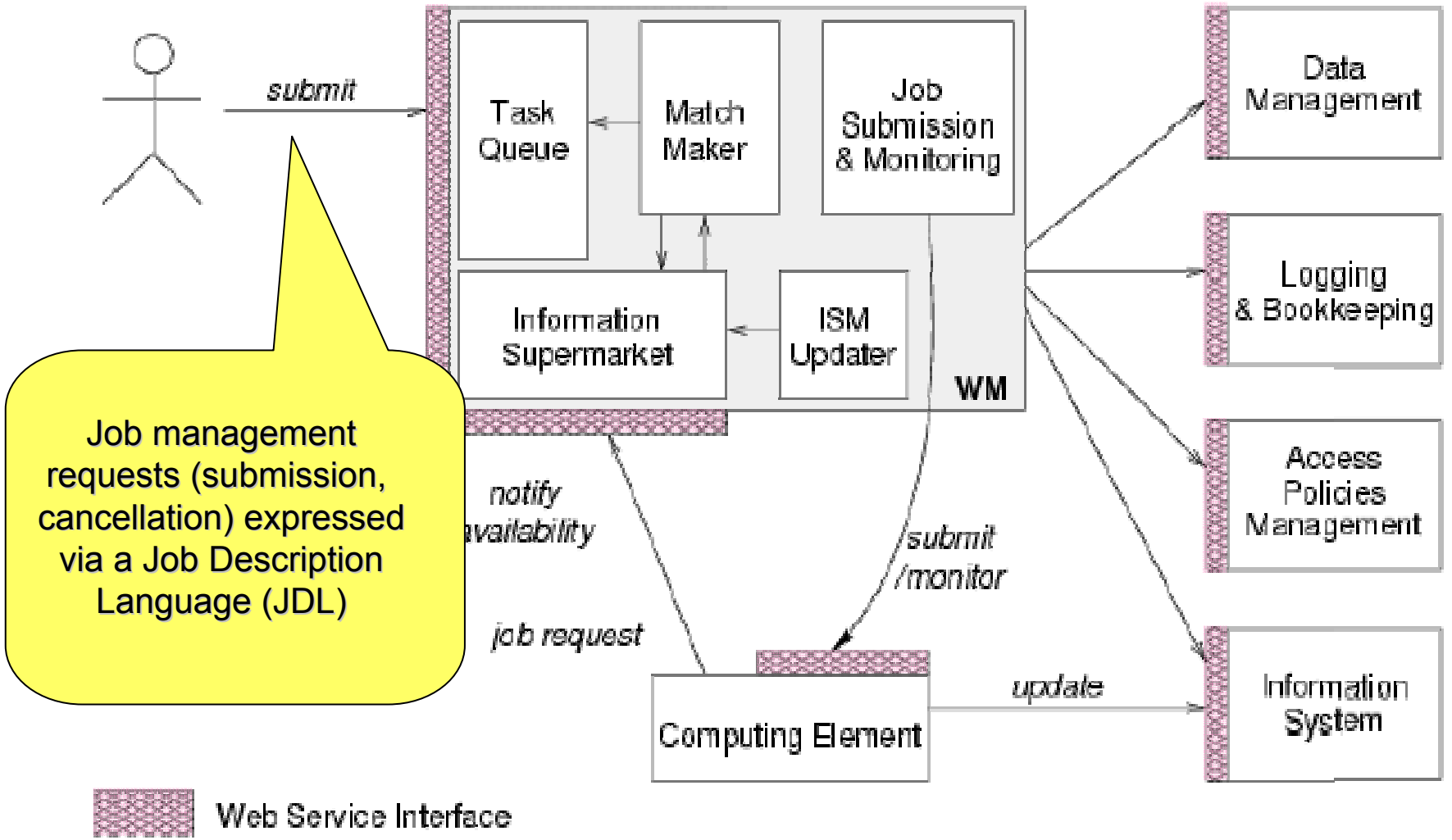
WMS's Scheduling Policies

- WM can adopt
  - **eager scheduling**
    - **a job is bound to a resource as soon as possible and, once the decision has been taken, the job is passed to the selected resource for execution**
  - **lazy scheduling**
    - **foresees that the job is held by the WM until a resource becomes available, at which point that resource is matched against the submitted jobs**
      - ***the job that fits best is passed to the resource for immediate execution.***
- Varying degrees of eagerness (or laziness) are applicable
  - **match-making level**
    - **eager scheduling**
      - ***implies matching a job against multiple resources***
    - **lazy scheduling**
      - ***implies matching a resource against multiple jobs***

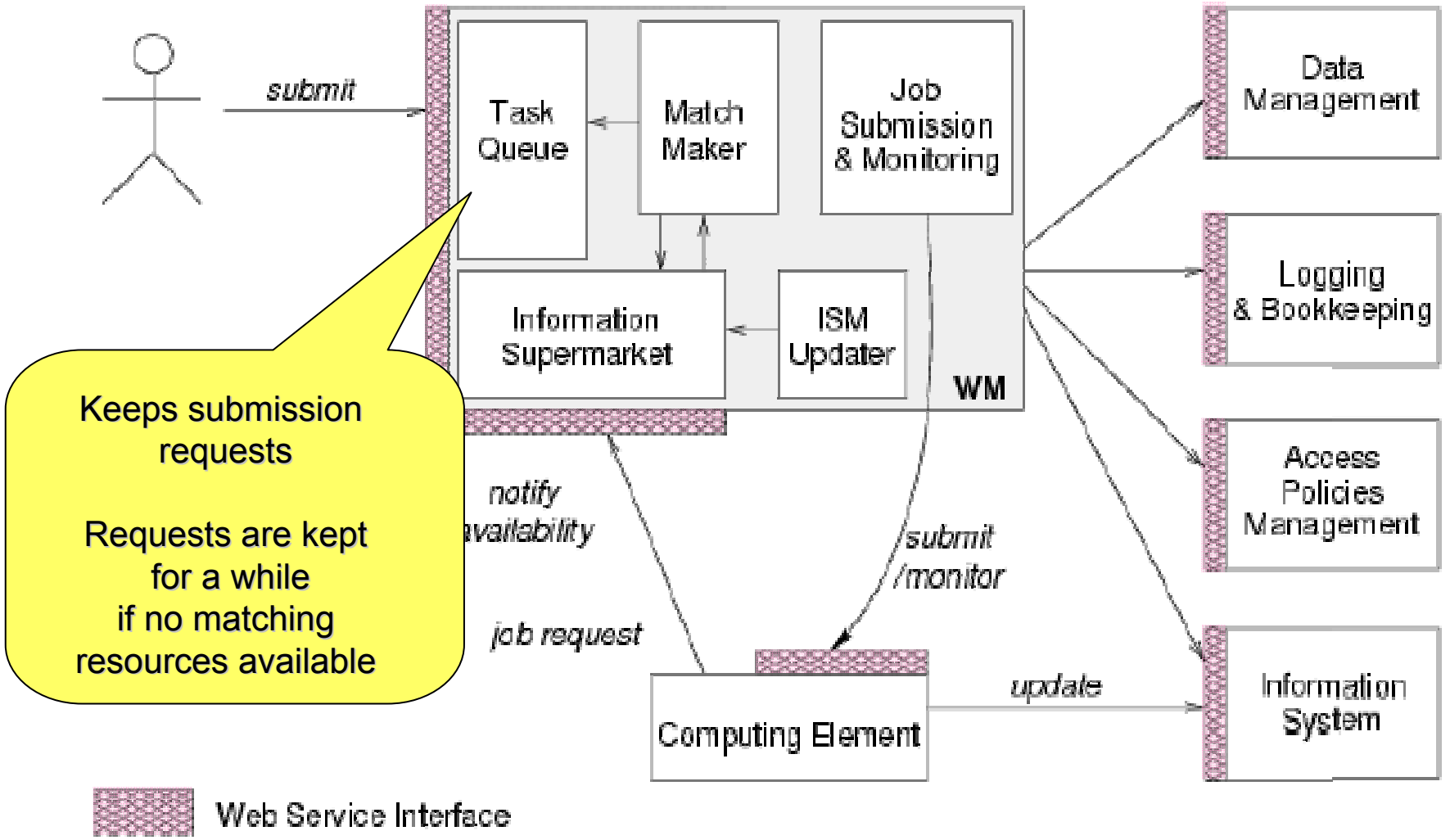
WMS's Architecture



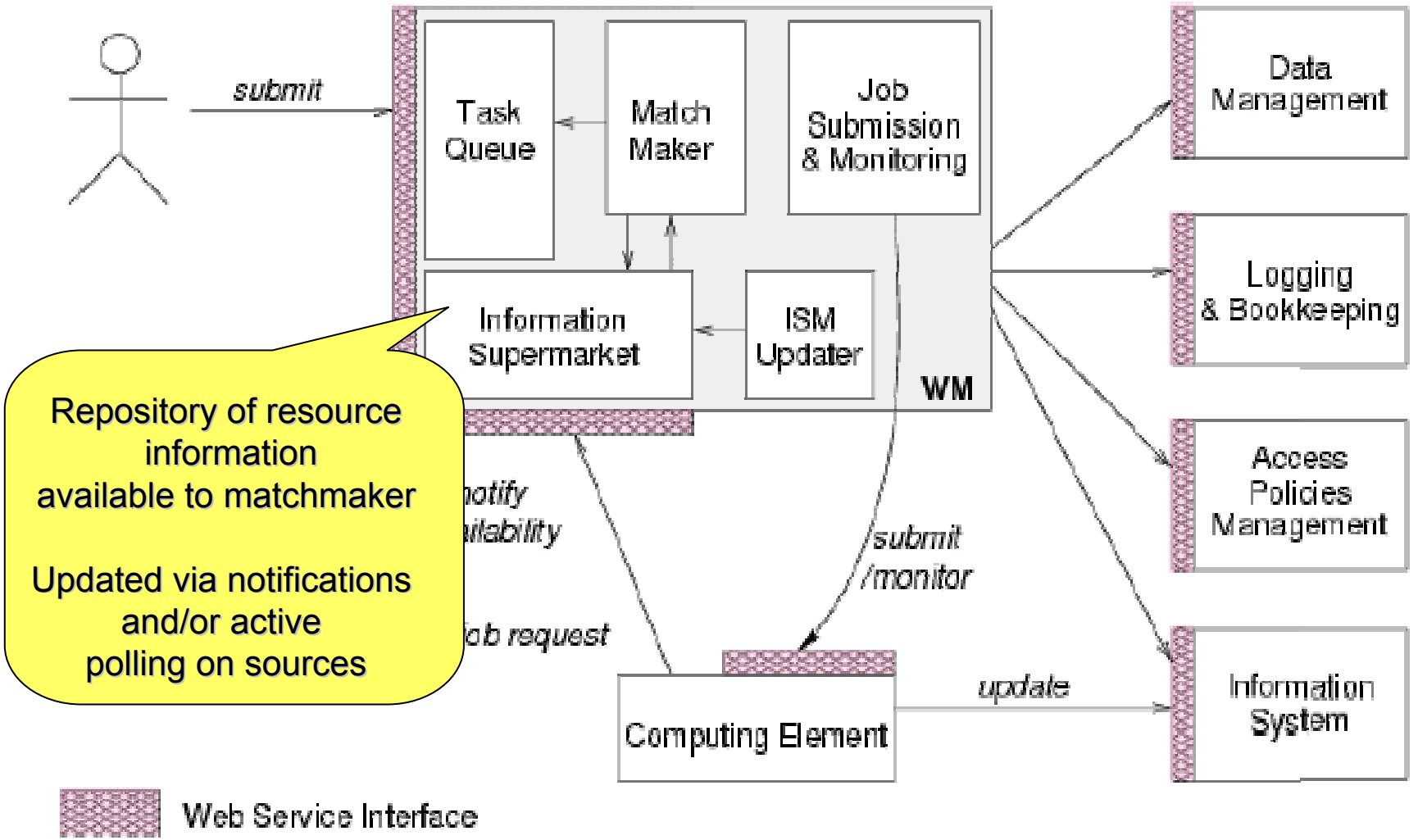
WMS's Architecture



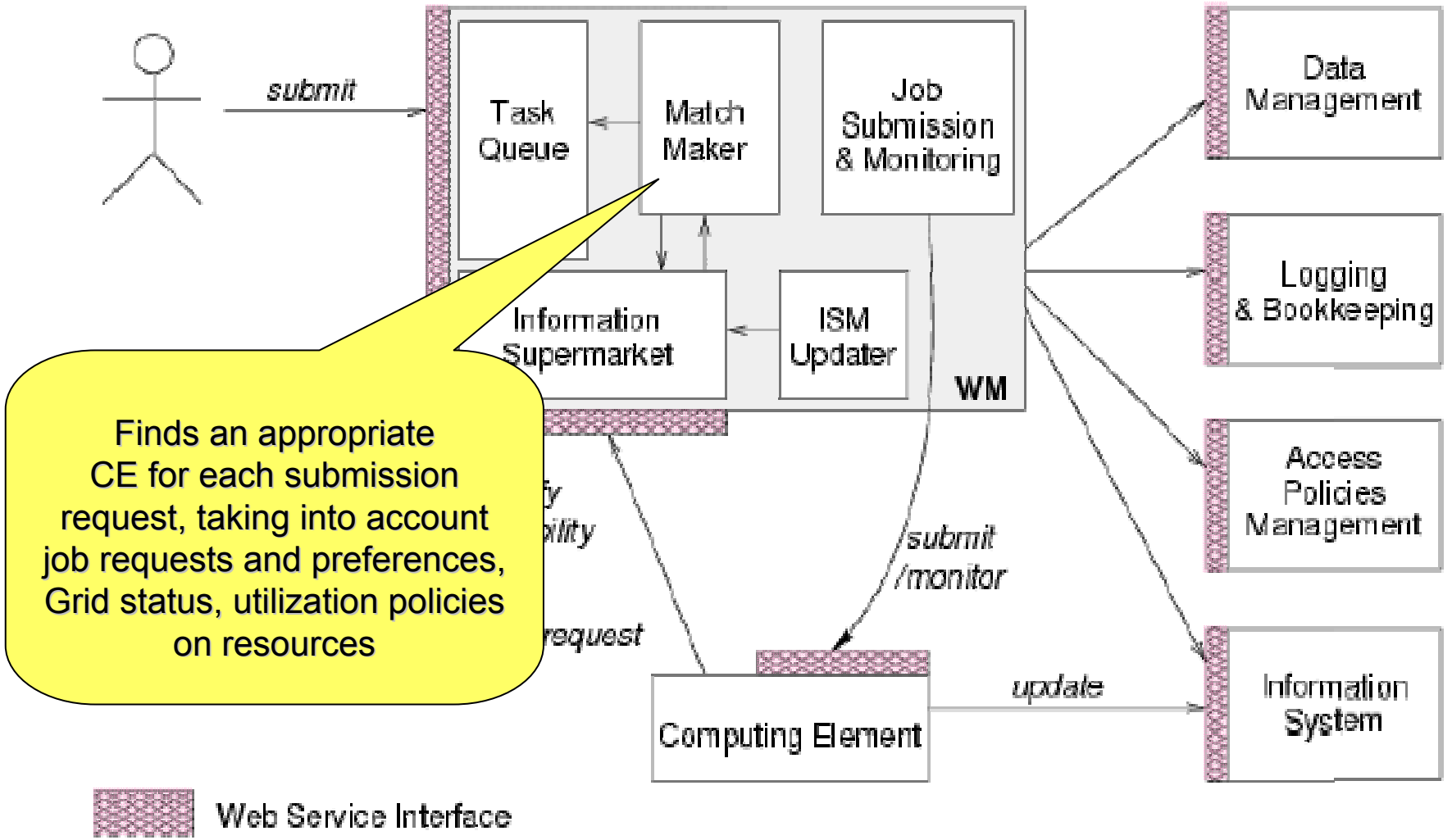
WMS's Architecture



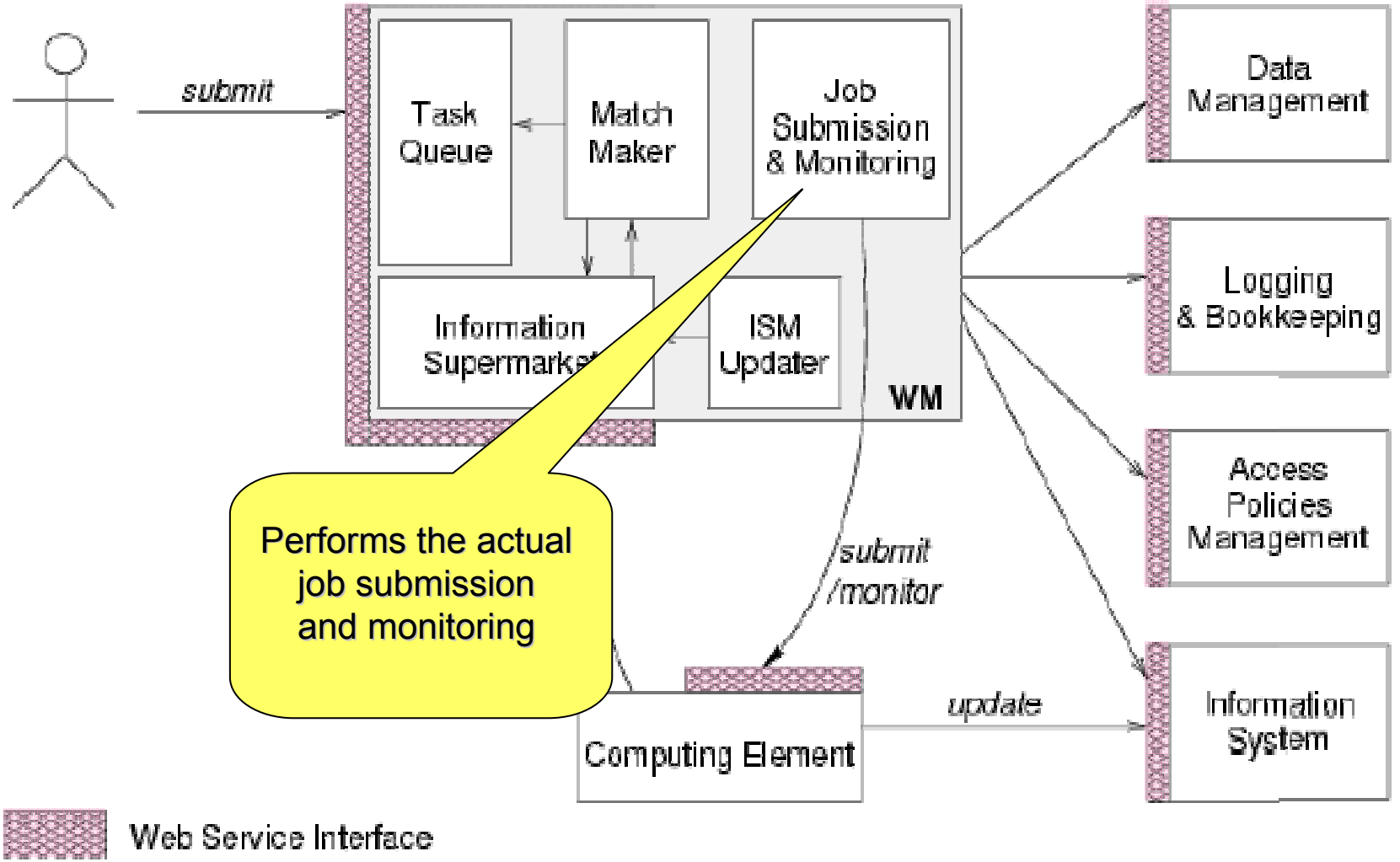
WMS's Architecture



WMS's Architecture



WMS's Architecture



THE INFORMATION SUPERMARKET

- ISM represents one of the most notable improvements in the WM as inherited from the EU DataGrid (EDG) project
  - **decoupling between the collection of information concerning resources and its use**
    - **allows flexible application of different policies**
- The ISM basically consists of a repository of resource information that is available in read only mode to the matchmaking engine
  - **the update is the result of**
    - **the arrival of notifications**
    - **active polling of resources**
    - **some arbitrary combination of both**
  - **can be configured so that certain notifications can trigger the matchmaking engine**
    - **improve themodularity of the software**
    - **support the implementation of lazy scheduling policies**

THE TASK QUEUE

- The Task Queue represents the second most notable improvement in the WM internal design
  - **possibility to keep a submission request for a while if no resources are immediately available that match the job requirements**
    - **technique used by the AliEn and Condor systems**
- Non-matching requests
  - **will be retried either periodically**
    - **eager scheduling approach**
  - **or as soon as notifications of available resources appear in the ISM**
    - **lazy scheduling approach**

JOB LOGGING AND BOOKKEEPING

- L&B tracks jobs in terms of *events*
  - **important points of job life**
    - **submission, finding a matching CE, starting execution etc**
      - ***gathered from various WMS components***
- The events are passed to a physically close component of the L&B infrastructure
  - ***locallogger***
    - **avoid network problems**
      - ***stores them in a local disk file and takes over the responsibility to deliver them further***
- The destination of an event is one of *bookkeeping servers*
  - **assigned statically to a job upon its submission**
    - **processes the incoming events to give a higher level view on the job states**
      - **Submitted, Running, Done**
    - **various recorded attributes**
      - ***JDL, destination CE name, job exit code***
- Retrieval of both job states and raw events is available via legacy (EDG) and WS querying interfaces
  - **user may also register for receiving notifications on particular job state changes**

JOB SUBMISSION Services

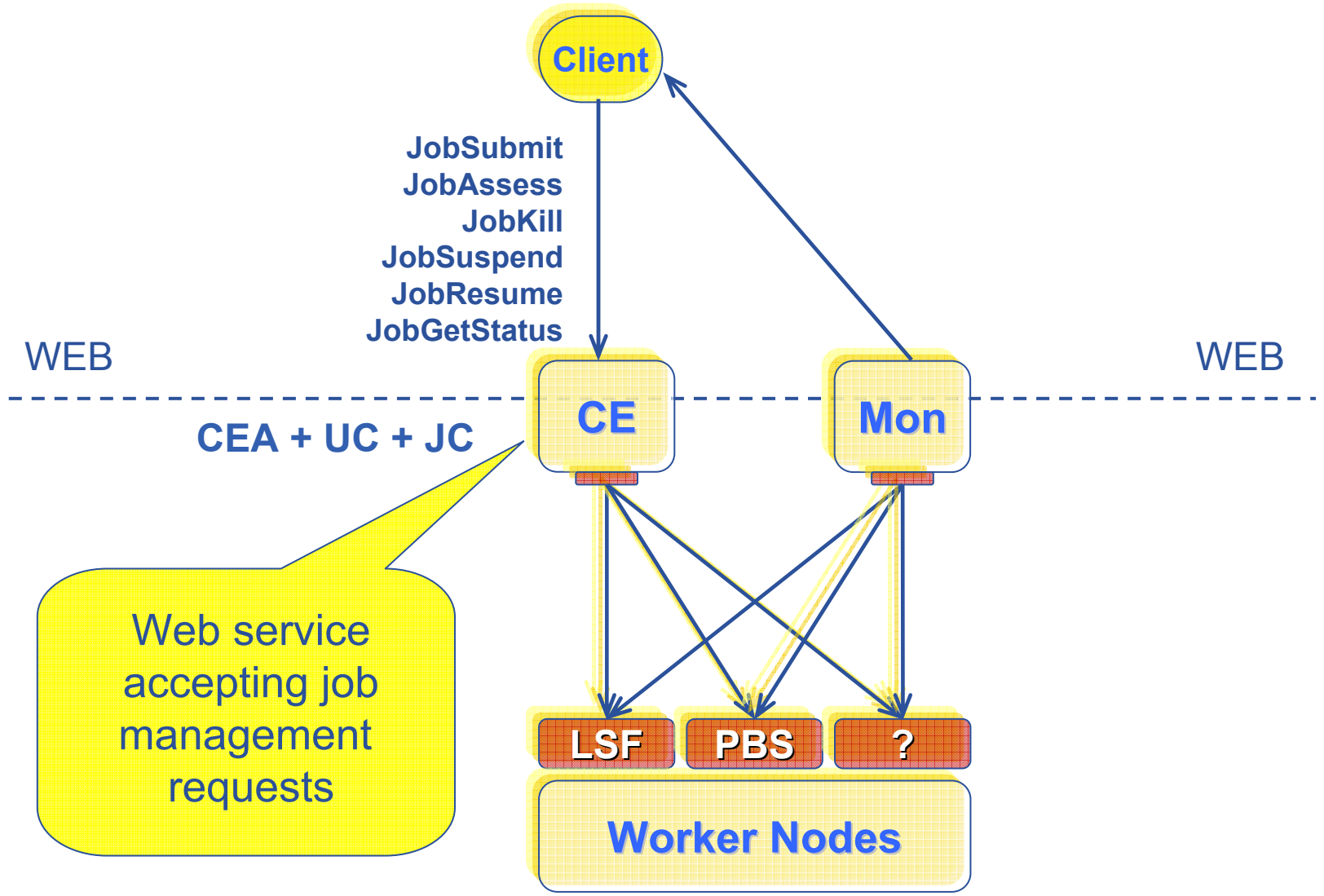
WMS components handling the job during its lifetime and performing the submission

- **Job Adapter**
  - **is responsible for**
    - **making the final touches to the JDL expression for a job, before it is passed to CondorC for the actual submission**
    - **creating the job wrapper script that creates the appropriate execution environment in the CE worker node**
      - ***transfer of the input and of the output sandboxes***
- **CondorC**
  - **responsible for**
    - **performing the actual job management operations**
      - ***job submission, job removal***
- **DAGMan**
  - **meta-scheduler**
    - **purpose is to navigate the graph**
    - **determine which nodes are free of dependencies**
    - **follow the execution of the corresponding jobs.**
  - **instance is spawned by CondorC for each handled DAG**
- **Log Monitor**
  - **is responsible for**
    - **watching the CondorC log file**
    - **intercepting interesting events concerning active jobs**
      - ***events affecting the job state machine***
    - **triggering appropriate actions.**

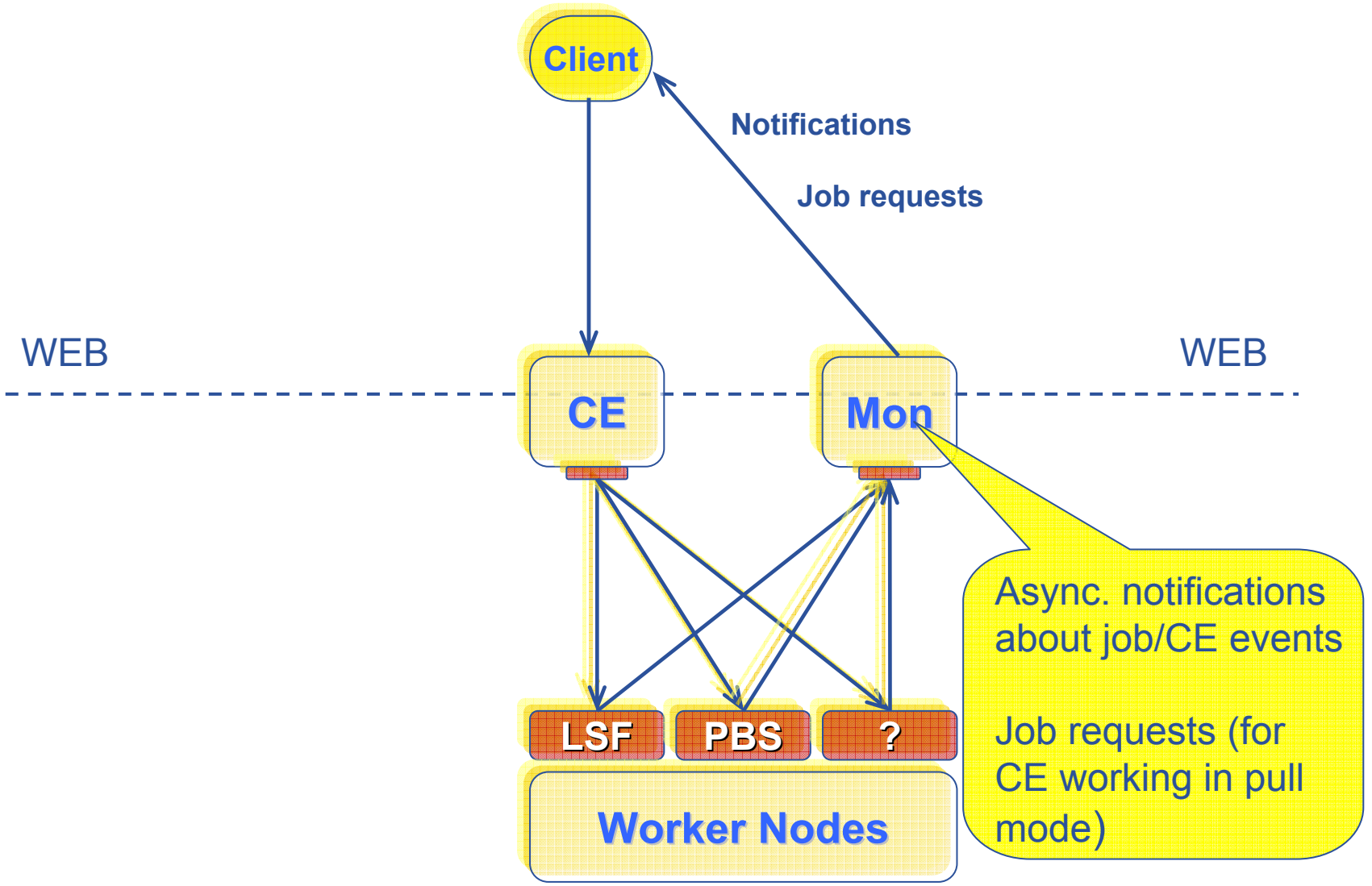
Computing Element

- **Service representing a computing resource**
- **Refers to a Cluster of Computational Resources, also heterogeneous.**
- **Main functionality: job management**
  - Run jobs
  - Cancel jobs
  - Suspend and resume jobs, send signals to them, get status or notification.
  - Provide info on “quality of service”
    - How many resources match the job requirements ?
    - What is the estimated time to have the job starting its execution ? (ETT)
- **Used by the WM or by any other client (e.g. end-user)**
- **CE architecture accommodated to support both push and pull model**
  - Push model: the job is pushed to the CE by the WM
  - Pull model: the CE asks the WM for jobs
- **These two models are somewhat mirrored in the resource information flow**
  - In order to 'pull' a job a resource must choose where to 'push' information about itself (CE Availability message)

CE Architecture



CE Architecture



Job Provenance

- **Keeps track of definition of submitted jobs, execution conditions and job life cycle for a long time**
  - Job life logs (JDL, timestamps, jobids, …)
  - Executable and input/output files
  - Execution environment (OS, installed software version, …)
  - Custom data provided by user
- **Used for**
  - Debugging
  - Post-mortem analysis
  - Comparison of job executions in an evolving environment
- **Service components**
  - Primary Storage Server (permanent)
    - Keeps data in the most compact and economic form – Job Records
  - Index Servers (volatile)
    - Configured to support a set of queryable attributes (LB and non-LB attributes)

Grid Accounting

A generic Grid accounting process accumulates info on Grid Usage by users/groups (VOs) and involves many subsequent phases as:

- **Metering:** Collection of usage metrics on computational resources.
- **Accounting:** Storage of such metrics for further analysis.
- **Usage Analysis:** Production of reports from the available records.
- **Pricing:** Assign and manage prices for computational resources.
- **Billing:** Assign a cost to user operations and charge them.
- **To be used:** To track resource usage | To discover abuses (and help avoiding them).

- **Allows implementation of submission policies based on resource usage**
  - Exchange market among Grid users and Grid resource owners, which should result in market equilibrium → Load balancing on the Grid

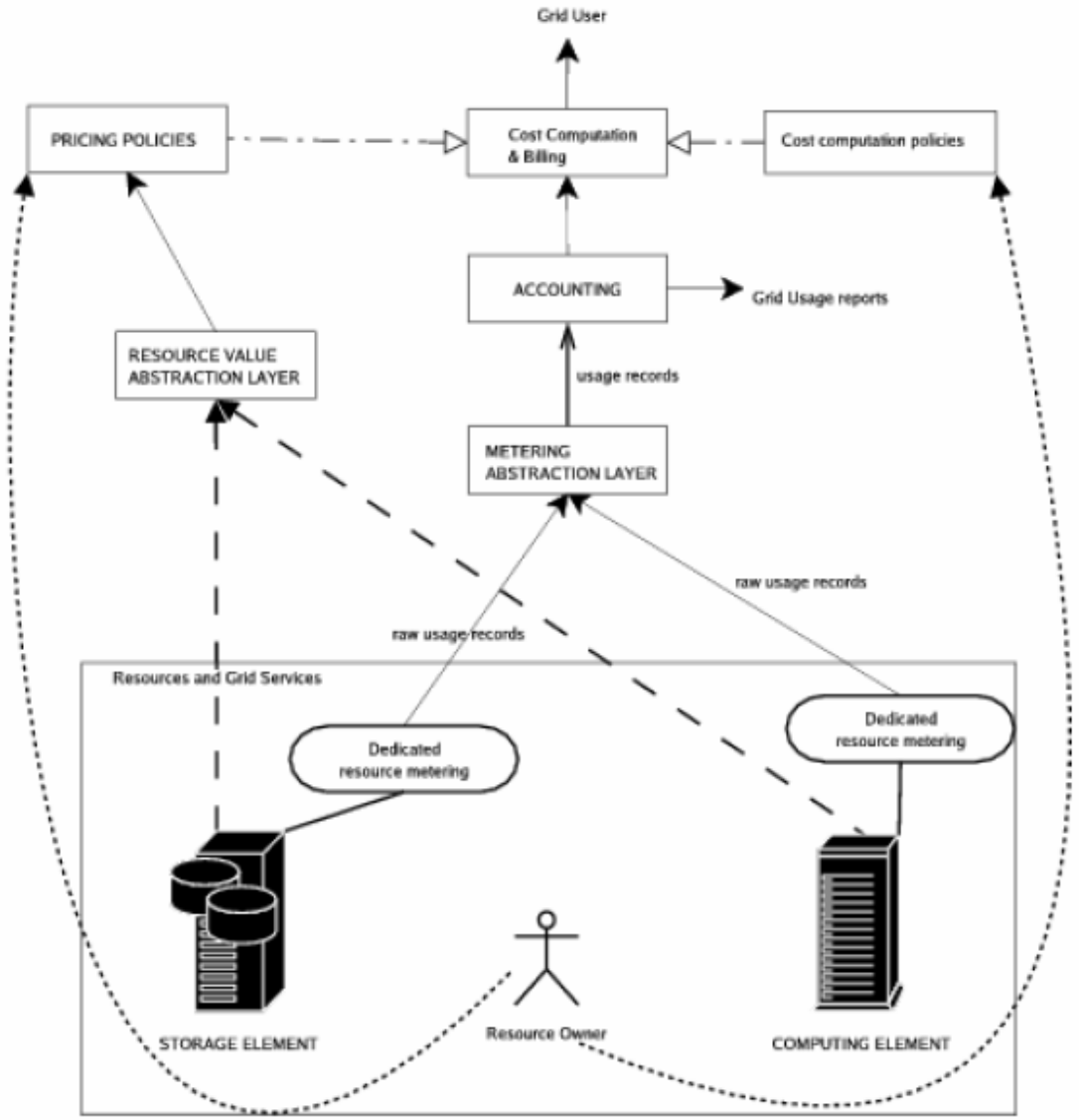
During the metering phase the user payload on a resource needs to be correctly measured, and unambiguously assigned to the Grid User that directly or indirectly requested it to the Grid → Load Dedicated Sensors for Grid Resources

These pieces of information, when organized, form the Usage Record **for the user process → *Grid Unique Identifier* (for User, Resource, Job) plus the metrics of the resource consumption.**

A distributed architecture **is essential, as well as reliable and** fault tolerant communication **mechanisms.**

Different types of users are interested in different views of the usage records.

Accounting System Architecture



Pricing & Billing

Resource owners may want to charge the users, thus it is necessary to establish a cost for the service furnished to the user.

A cost is usually computed according to a price assigned to the unit of usage of a computing resource and to the usage measured for the same resource.

Thus a service responsible for managing the resource prices and communicating them to all the partners is needed.

The way prices are set contributes to the creation of an economic market that deeply influences the behaviour of the Grid as a dynamic system.

Once the resource consumption is known and a price is assigned to the computational resources, it is possible to define a cost that can be charged to the Grid User.

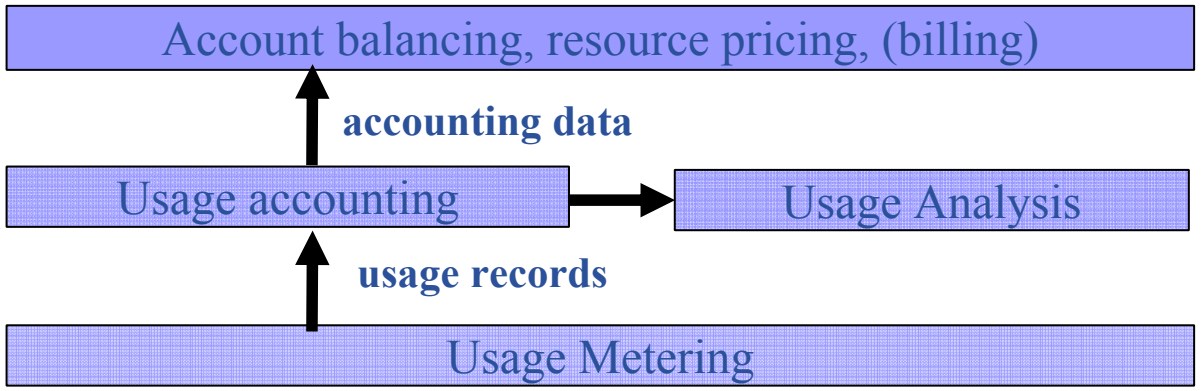
The final cost applied to the User is influenced also by policy issues like discounts or offers.

DGAS

The *Data Grid Accounting System* was originally developed within the EU Datagrid Project and is now being maintained and re-engineered within the EU EGEE Project.

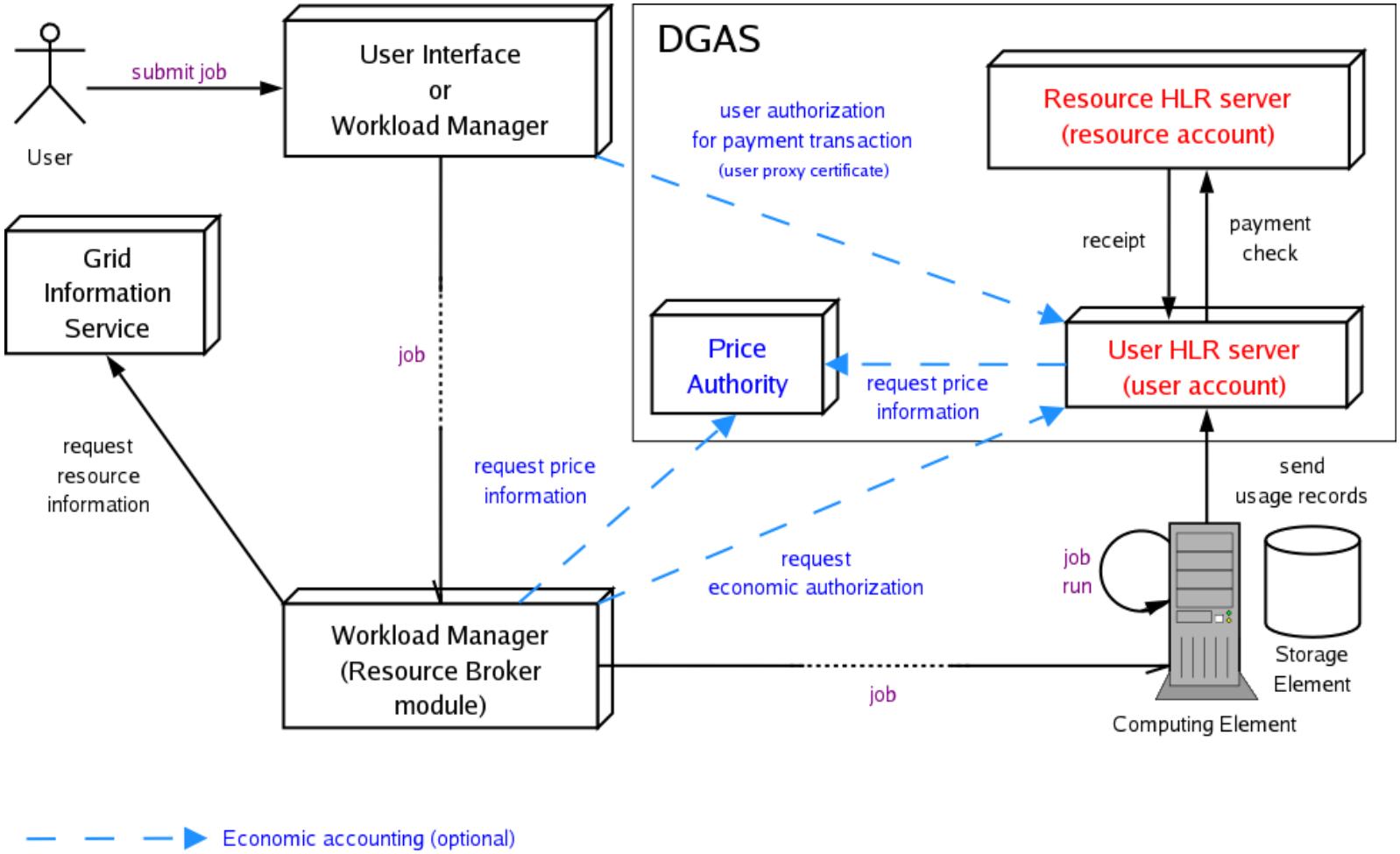
The Purpose of *DGAS* is to implement *Resource Usage Metering*, *Accounting* and *Account Balancing* (through *resource pricing*) in a fully distributed Grid environment. It is conceived to be distributed, secure and extensible**.**

The system is designed in order for Usage Metering, Accounting and Account Balancing (through resource pricing) to be indipendent layers.



DGAS Accounting Architecture

A simplified view of DGAS within the WMS context.



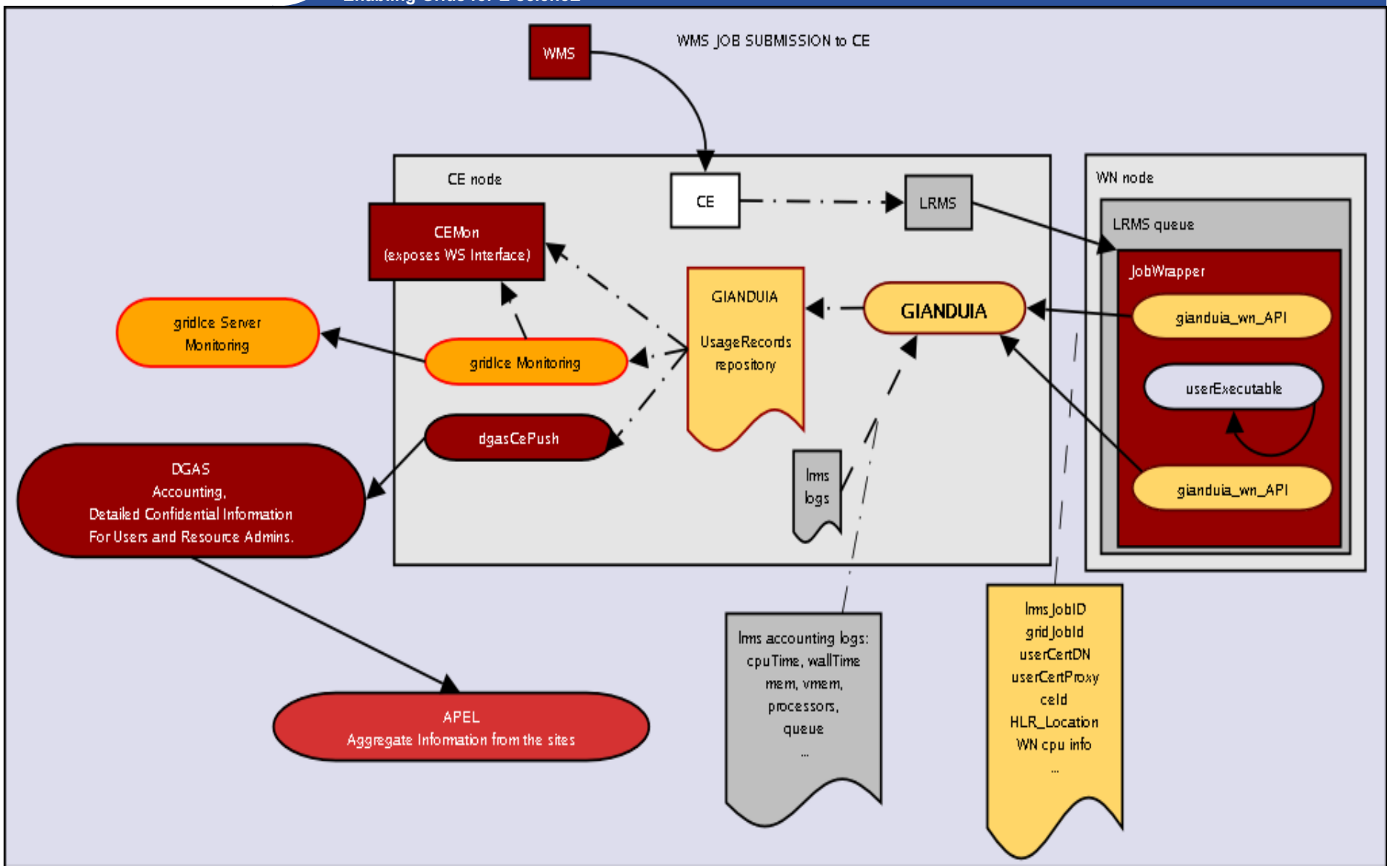
DGAS: Metering

Usage Metering **on Computing Elements is done by lightweight sensors installed on the Comuting Elements. These sensors parse PBS/LSF/Torque event logs to built** *Usage Records* **that can be passed to the accounting layer.**

For a reliable accounting of resource usage (essential for billing) it is important that the collected data is *unequivocally* associated to the unique grid ID of the user (certificate subject/DN), the resource (CE ID) as well as the job (global job ID).

A process, completely transparent to the Grid User collects the necessary information needed by the Accounting. These, and the corresponding metrics are sent via an *encrypted* channel to the Accounting System *signed with the user credentials*.

Metering: Gianduia



X.509: security inside the Grid

- For the Grid to be an effective framework for largely distributed computation, users, user processes and grid services must work in a secure environment.

- **All interactions between WMS components (expecially if network separated) will be mutually authenticated: any entity authenticates itself to the other peer using either its own credential or a delegated user credential or both**.
  - User Interface passing a job to the Network Server.
  - WMS-UI interacting with the Logging and Bookkeeping service.

- The user or service identity and their public key are included in a **X.509 certificate** signed by a trusted **Certification Authority (CA)**, to guarantee the association between that public key and its owner.

- **The user has to possess a valid X.509 certificate on the submitting machine, consisting of two files: the *certificate file* and the *private key file*.**
  - ***"$X509_USER_CERT"* and *"$X509_USER_KEY"***
  - ***"$HOME/.globus/usercert.pem" and "$HOME/.globus/userkey.pem"***

X.509: extracting user{cert | key} files

Usually X.509 Certificates are downloaded using a browser and managed by the browser itself.

- Anyway it is possible to export your certificate in a file PKCS12 (which will probably have the extension .p12 or .pfx).
- Unfortunately PKCS12 format is not accepted by Globus security infrastructure, but you can easily convert it into the supported standard (PEM). This operation will split your *.p12 file in two files: the certificate (usercert.pm) and the private key (userkey.pm).
- *With openssl tool:*
- *$ openssl pkcs12 -nocerts -in mycert.p12 -out userkey.pem*
- *$ openssl pkcs12 -clcerts -nokeys -in mycert.p12 -out usercert.pem*
- *$ chmod 0400 userkey.pem*
- *$ chmod 0600 usercert.pem*
- Permission must be set as shown not only for security reasons: *voms-proxy-init* and *grid-proxy-init* commands will fail if your private key is not protected as listed above.

X.509: Creating a Proxy Certificate

- Actually the user certificate and private key files are not mandatory on the WMS-UI machine:
  - needed for the creation of the proxy user credentials through *grid-proxy-init* or *voms-proxy-init*
  - downloadability of proxy credentials from a trusted site.

- **All WMS-UI commands, when started, check for the existence and expiration date of user proxy credentials in the location pointed to by *"$X509_USER_PROXY"* or in *"/tmp/x509up_u<UID>"* (where <UID> is the user identifier in the submitting machine OS) if the X509 environment variable is not set.**

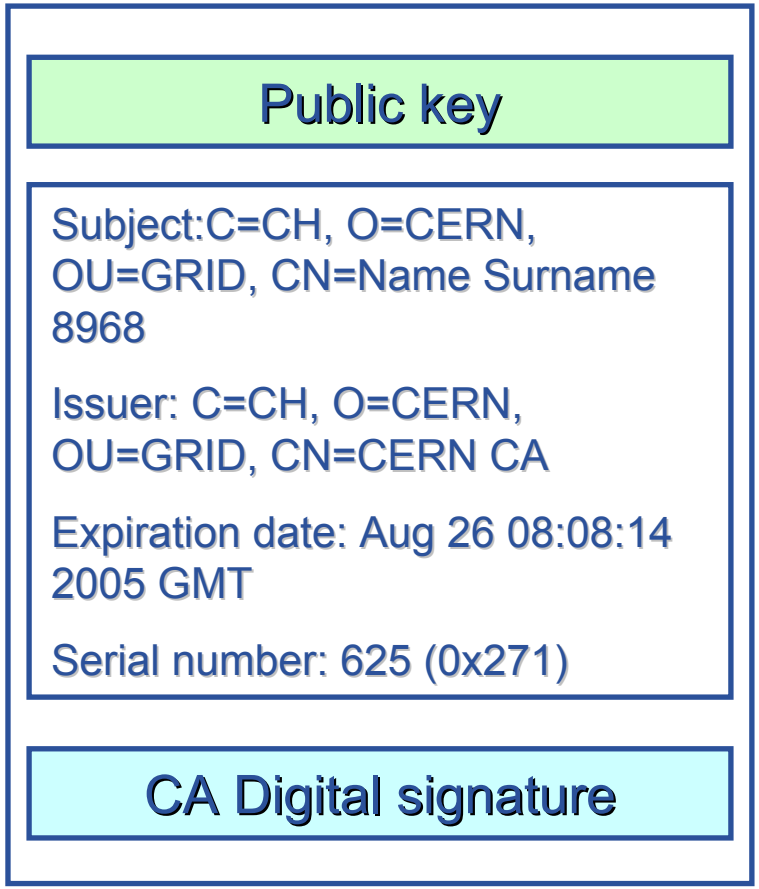
- **If the proxy certificate does not exist or has expired the WMS-UI returns an error message to the user and exits.**

- **Notes: Existence of multiple VOs.**

- **A job gets associated a valid proxy certificate (the submitting user's one) when it is submitted by the WMS-UI to NS. Proxy validity default set to 12 hours unless differently specified.**
  - --valid *voms-proxy-init*
  - --hours *grid-proxy-init*
  - features of MyProxy package. Registering a valid long-term certificate proxy that will be used by the WMS to perform a periodic credential renewal for the submitted job.

X.509: containt of the Certificate

- **An X.509 Certificate contains:**
  - owner's public key;
  - identity of the owner;
  - info on the CA;
  - time of validity;
  - Serial number;
  - digital signature of the CA



MyProxy

Consists of a server and a set of client tools that can be used to delegate and retrieve credentials to and from a server.

MyProxy Client commands:

- *myproxy-init*
- *myproxy-info*
- *myproxy-destroy*
- *myproxy-get-delegation*
- *myproxy-change-pass-phrase*

```
// myproxy-info -s <host name> -d
```

```
// myproxy-get-delegation -s <host name> -d
   -t <hours> -o <output file> -a <user proxy>
```

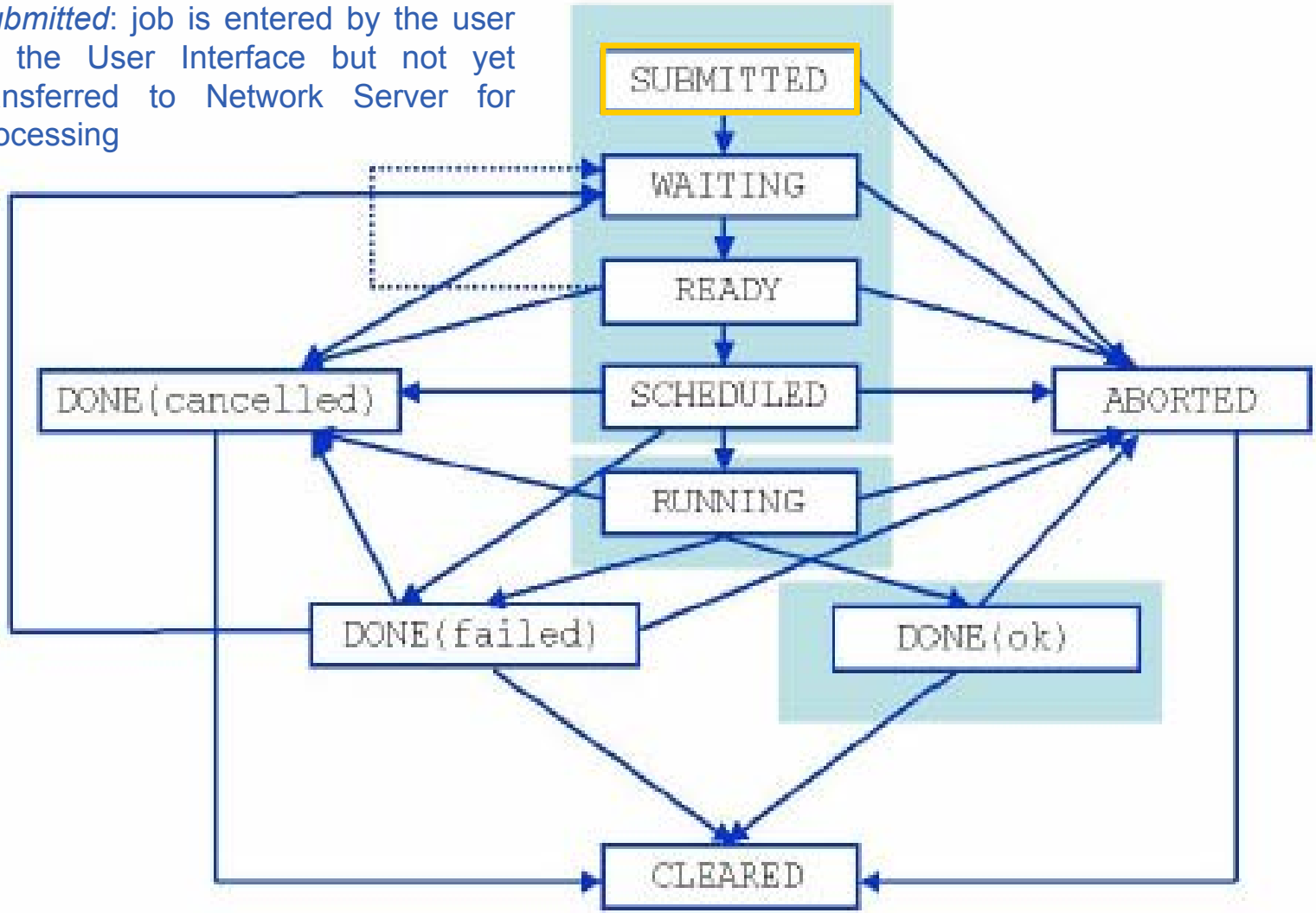
The ***myproxy-init*** command allows you to create and send a delegated proxy to a MyProxy server for later retrieval; in order to launch it you have to assure you're able to execute the grid-proxy-init or vomsproxy-init command.

```
myproxy-init -s <host name> -t <hours> -d –n
```

The myproxy-init command stores a user proxy in the repository specified by <host name> (the –s option). Default lifetime of proxies retrieved from the repository will be set to <hours> (see -t) and no password authorization is permitted when fetching the proxy from the repository (the -n option). The proxy is stored under the same user-name as is your subject in your certificate (-d).

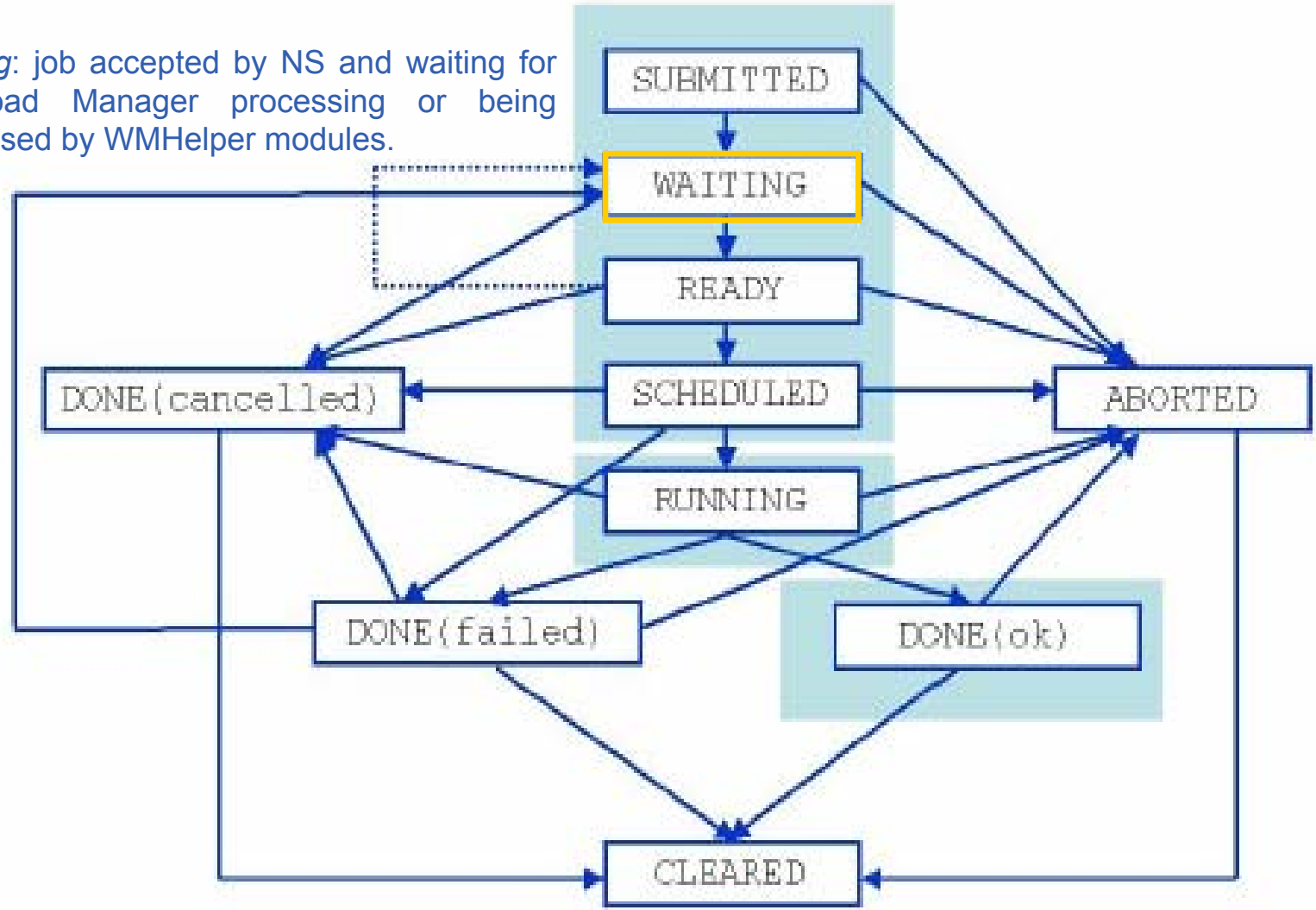
Jobs State Machine (1/9)

Submitted: job is entered by the user to the User Interface but not yet transferred to Network Server for processing

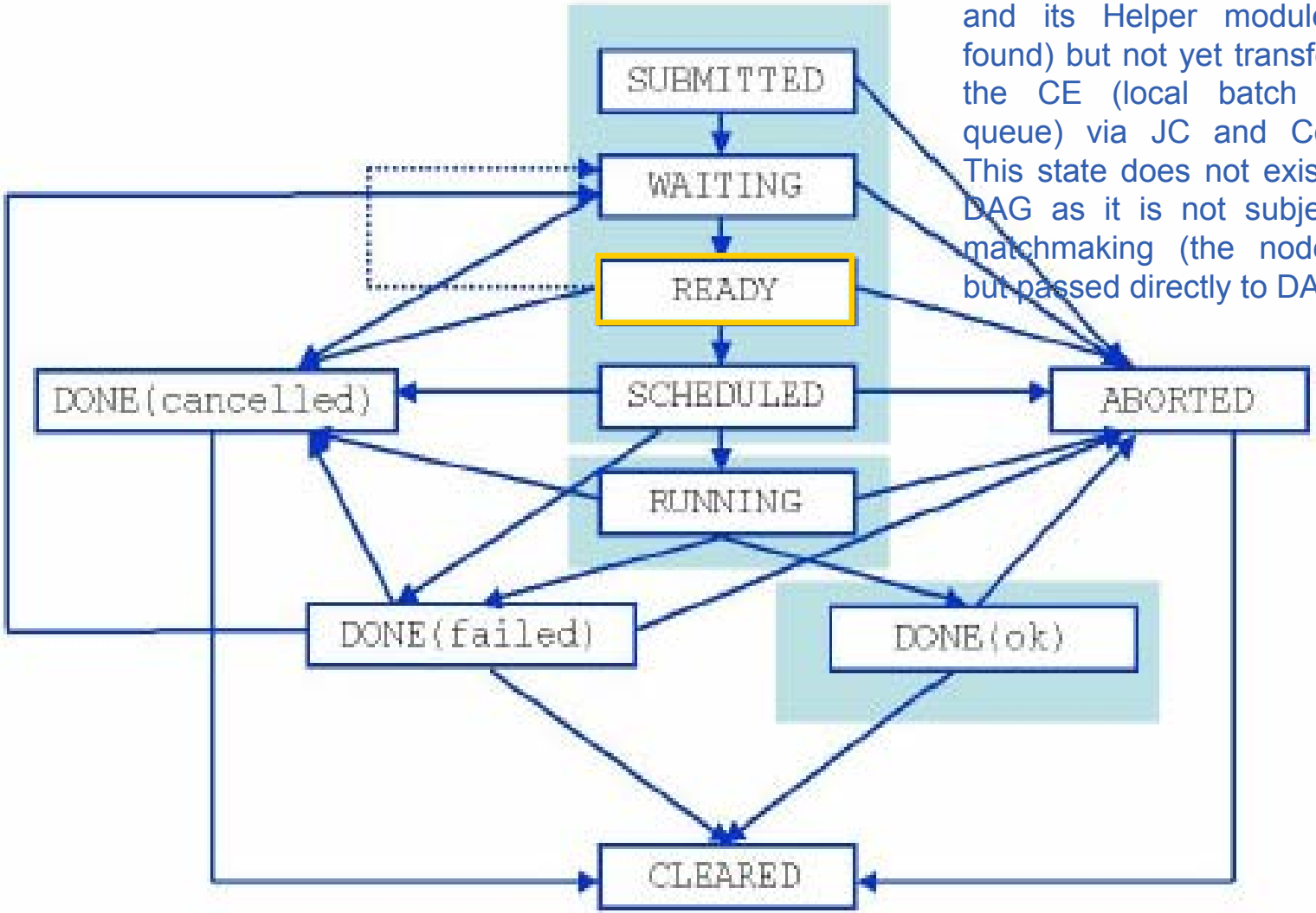


Jobs State Machine (2/9)

Waiting: job accepted by NS and waiting for Workload Manager processing or being processed by WMHelper modules.

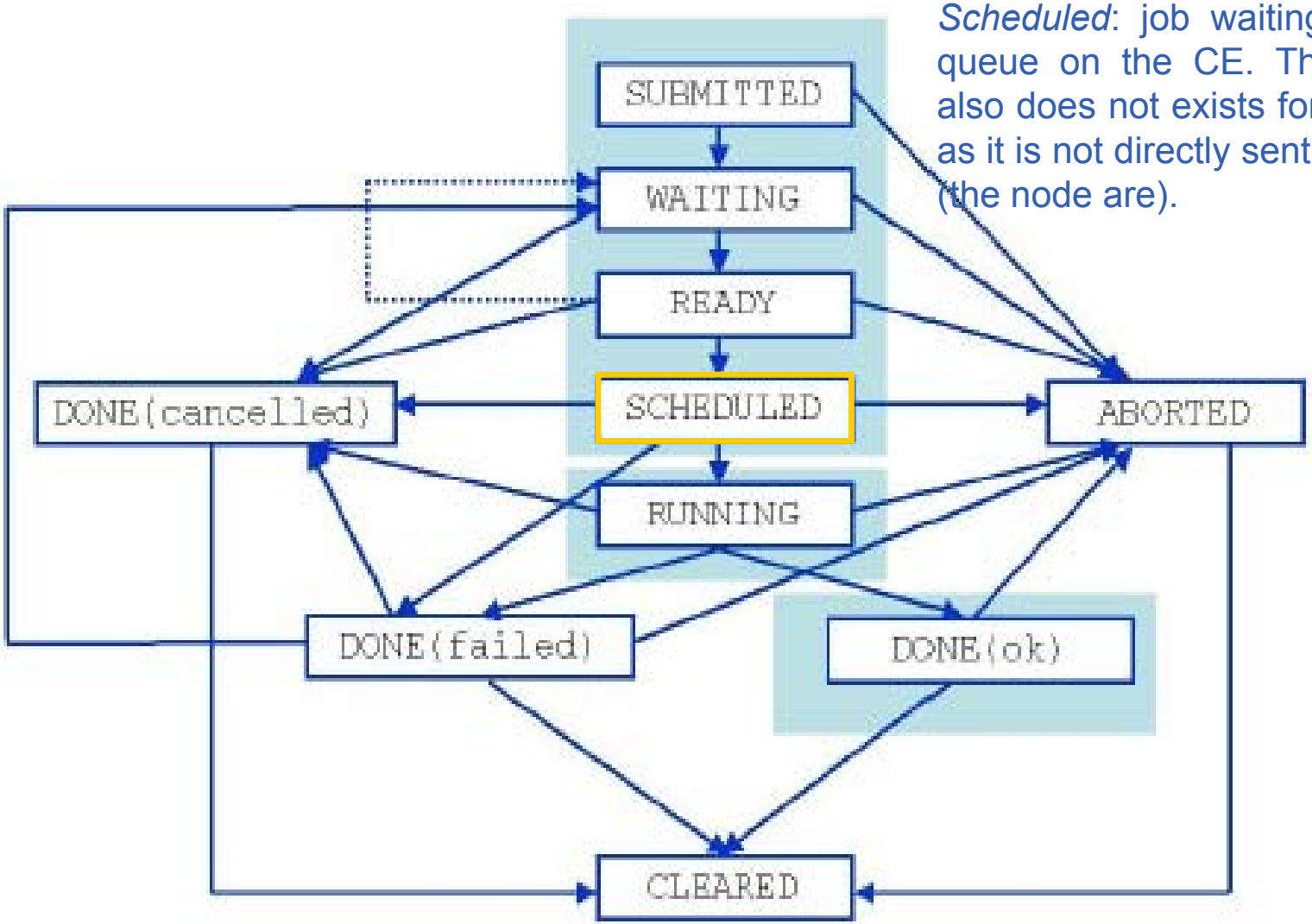


Jobs State Machine (3/9)



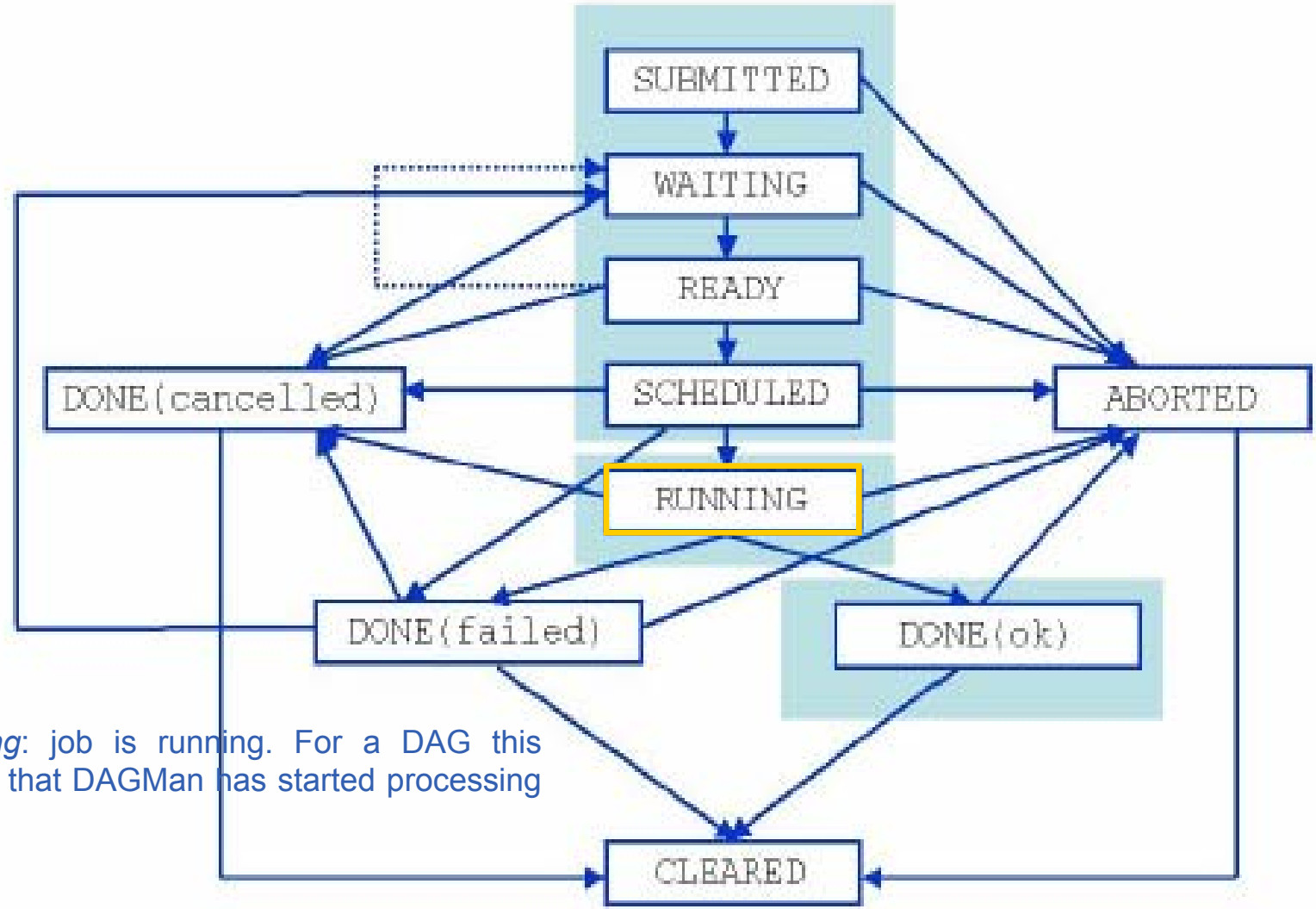
Ready: job processed by WM and its Helper modules (CE found) but not yet transferred to the CE (local batch system queue) via JC and CondorC. This state does not exists for a DAG as it is not subjected to matchmaking (the nodes are) but passed directly to DAGMan.

Jobs State Machine (4/9)



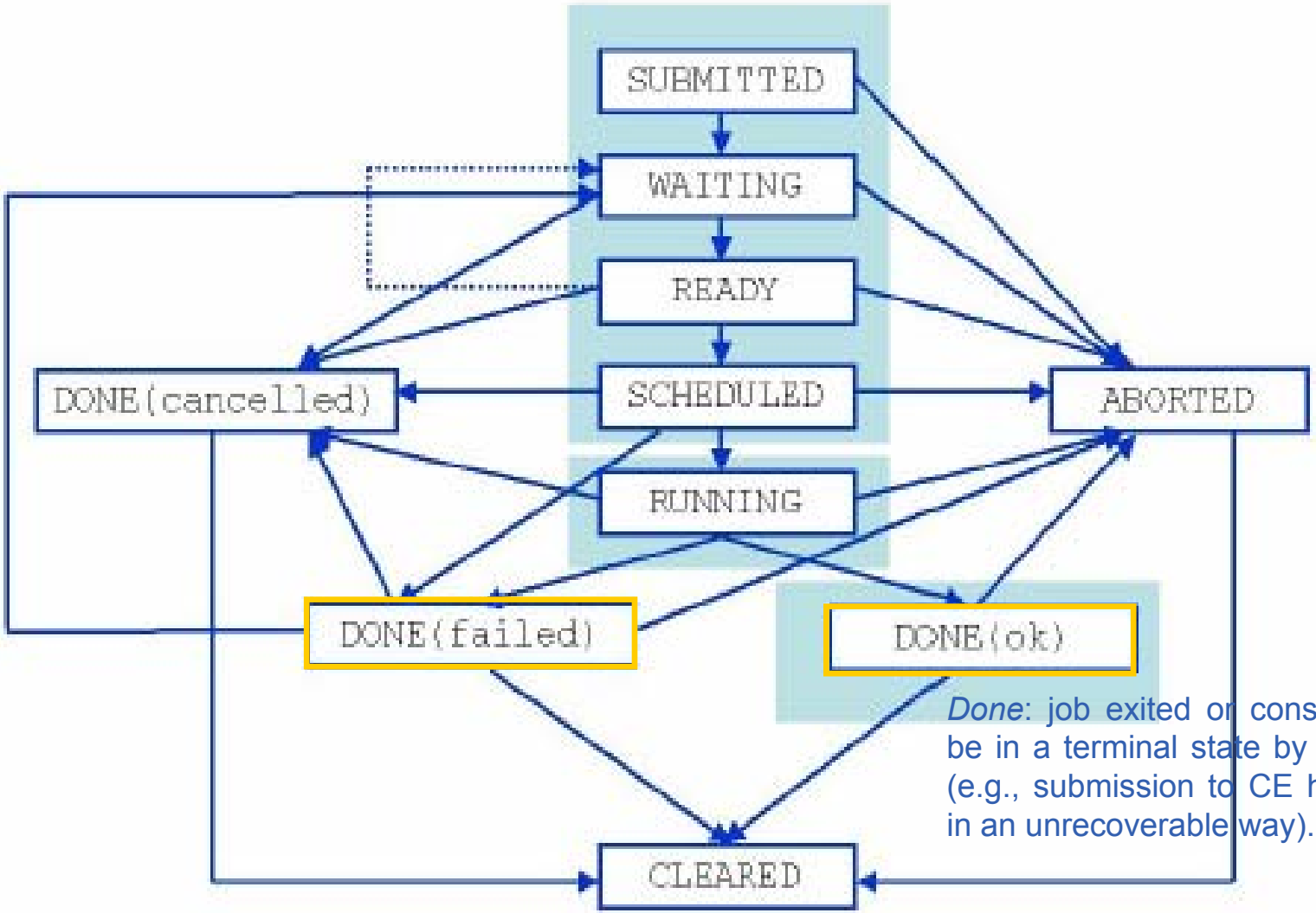
Scheduled: job waiting in the queue on the CE. This state also does not exists for a DAG as it is not directly sent to a CE (the node are).

Jobs State Machine (5/9)



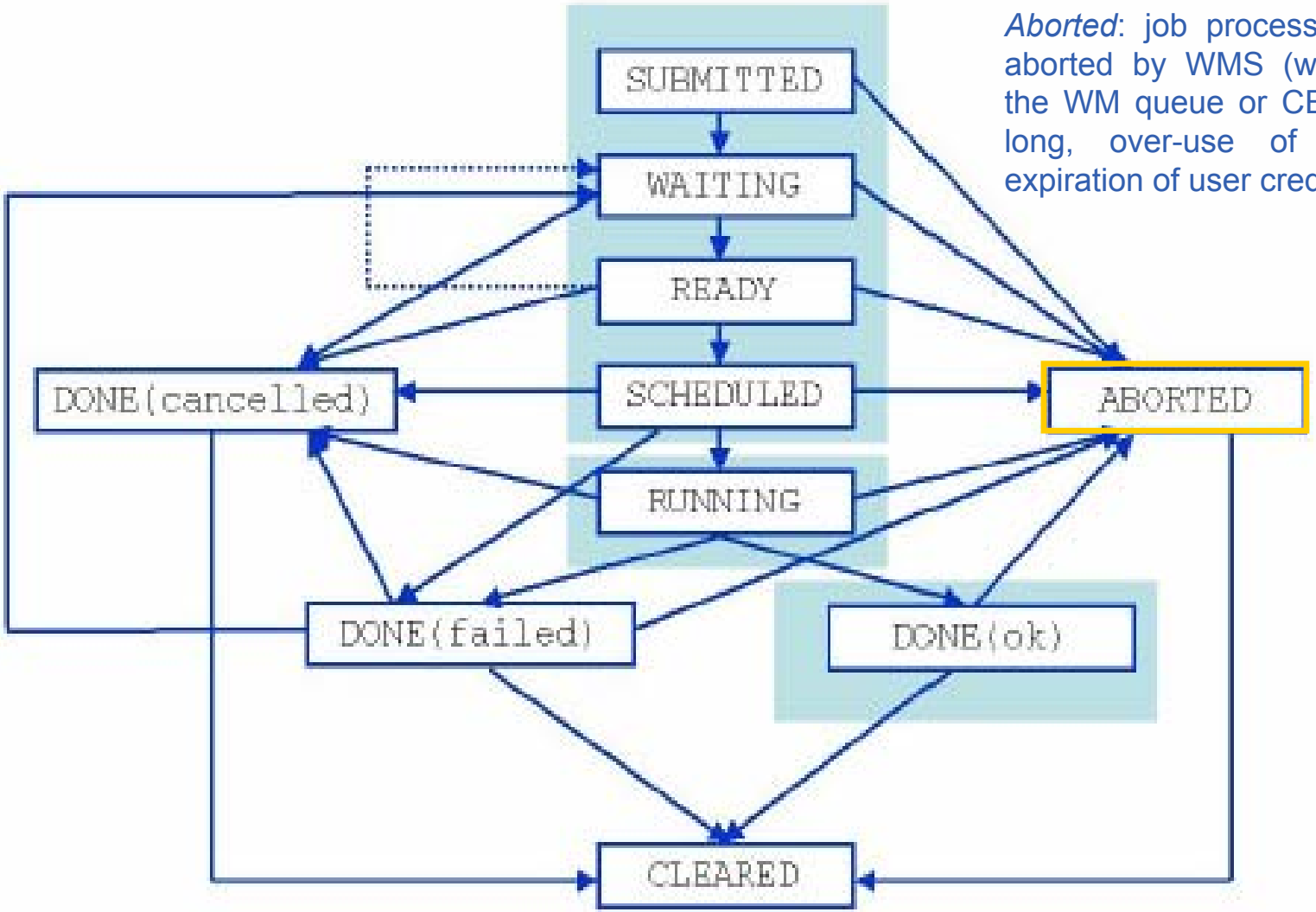
Running: job is running. For a DAG this means that DAGMan has started processing it.

Jobs State Machine (6/9)



Done: job exited or considered to be in a terminal state by CondorC (e.g., submission to CE has failed in an unrecoverable way).

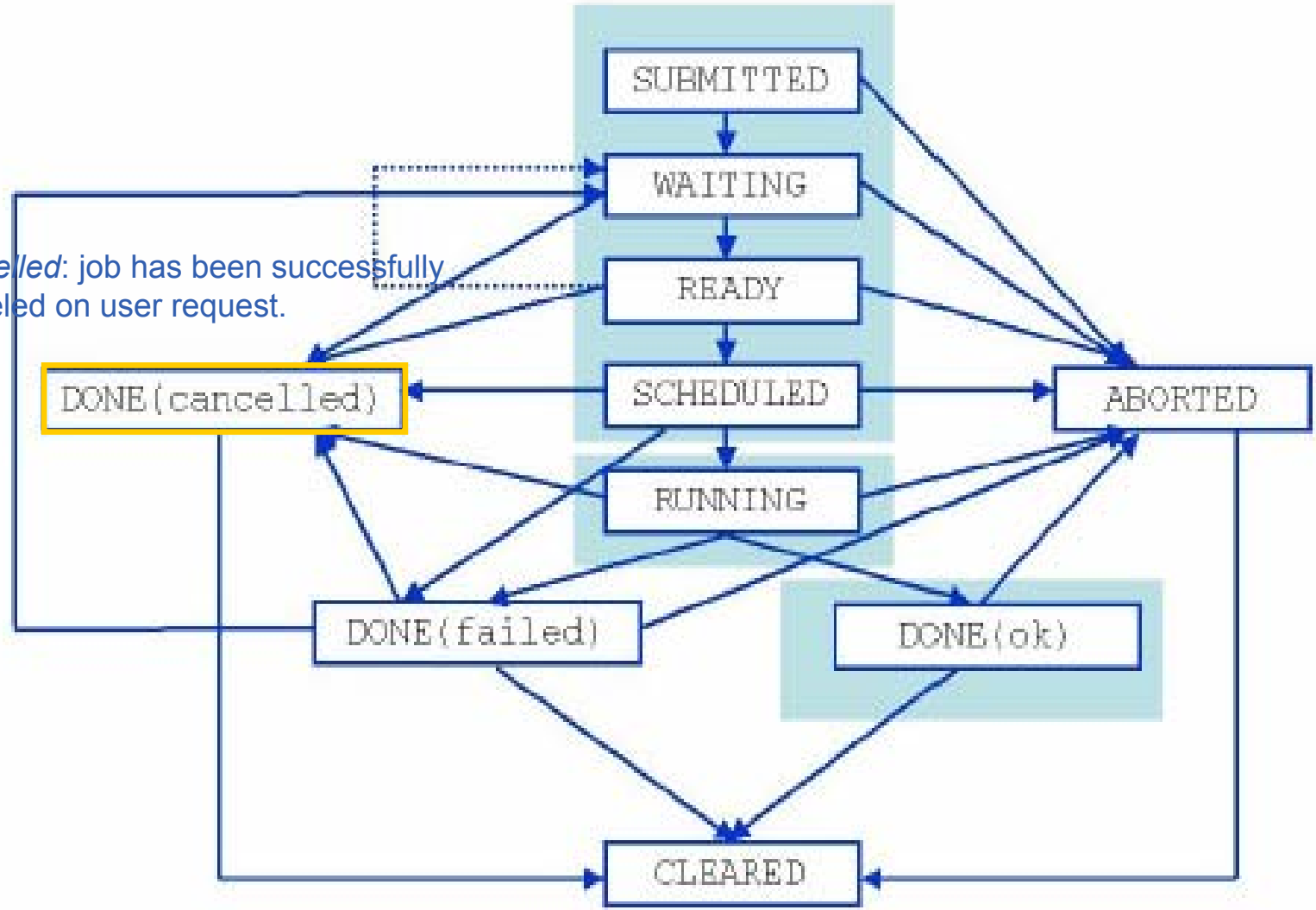
Jobs State Machine (7/9)



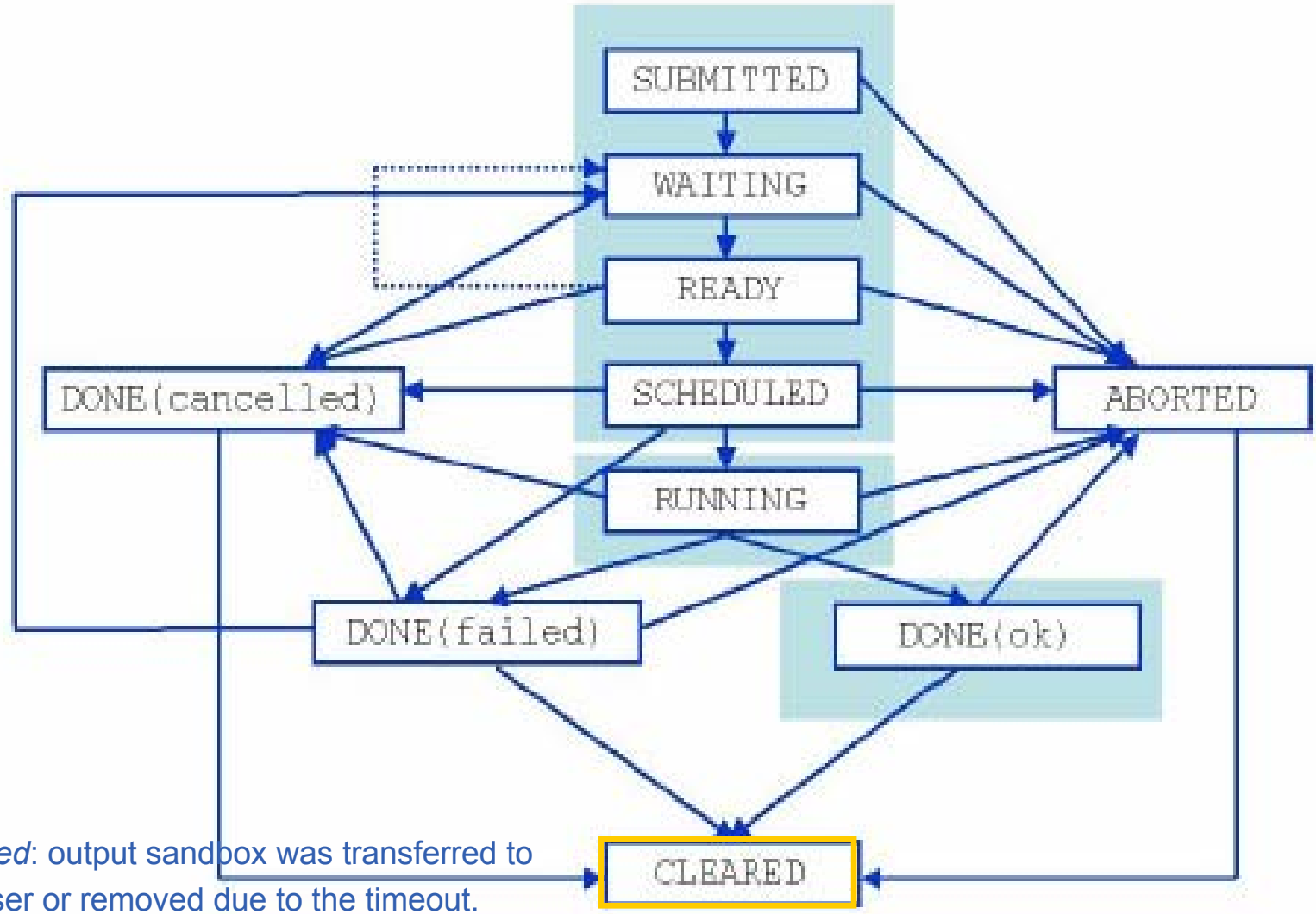
Aborted: job processing was aborted by WMS (waiting in the WM queue or CE for too long, over-use of quotas, expiration of user credentials).

Jobs State Machine (8/9)

Cancelled: job has been successfully canceled on user request.

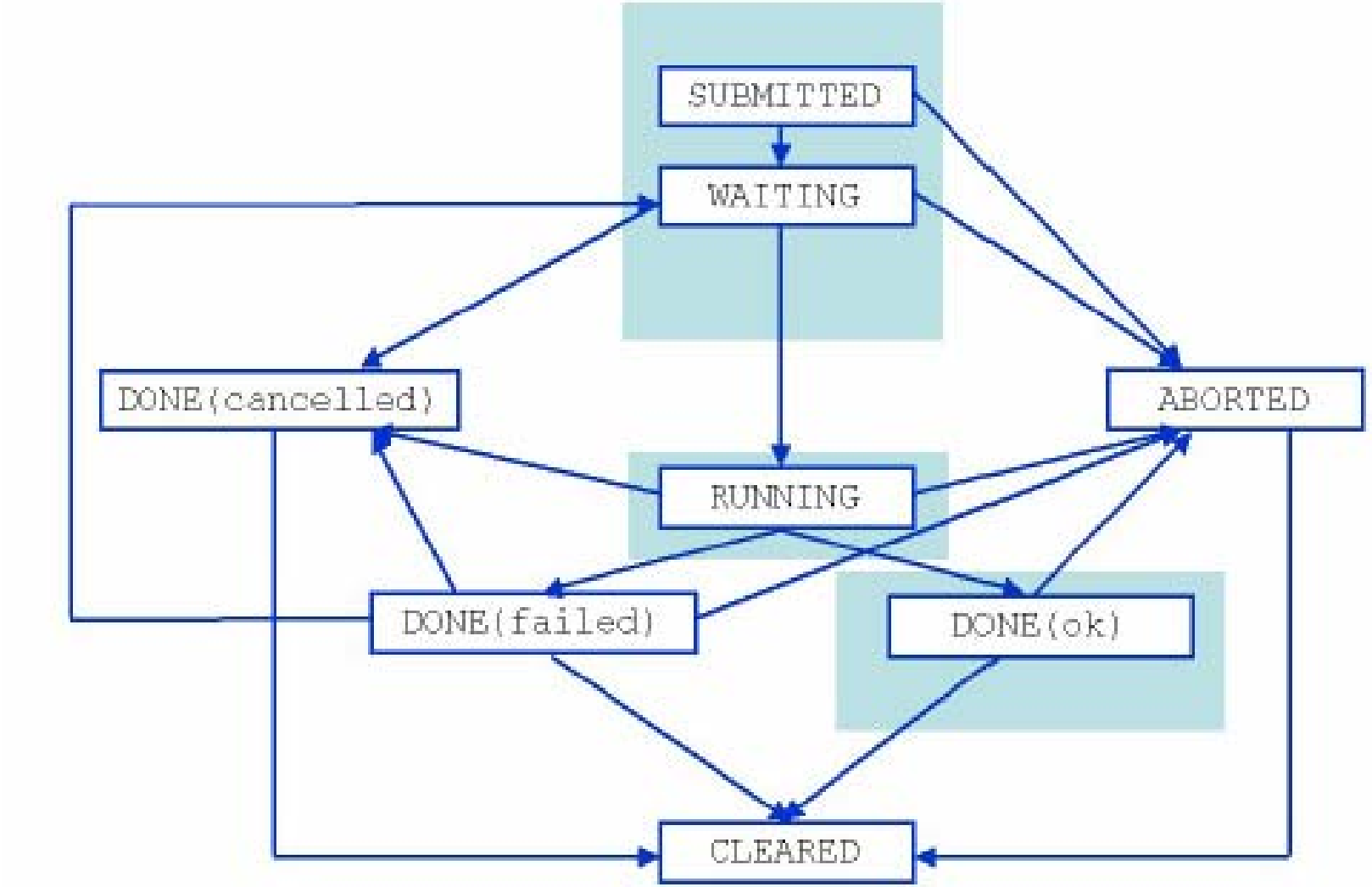


Jobs State Machine (9/9)



Cleared: output sandbox was transferred to the user or removed due to the timeout.

DAG State Machine



References

- **gLite WMS's User Guide**
  - https://edms.cern.ch/document/572489/1
- **EGEE Middleware Architecture DJRA1.1**
  - https://edms.cern.ch/document/476451/
- **Practical approaches to Grid workload management in the EGEE project – CHEP 2004**
  - https://edms.cern.ch/document/503558
- **Grid accounting in EGEE, current practices – Terena Network Conference 2005**
  - http://www.terena.nl/conferences/tnc2005/programme/presentations/show.php?pres_id=107