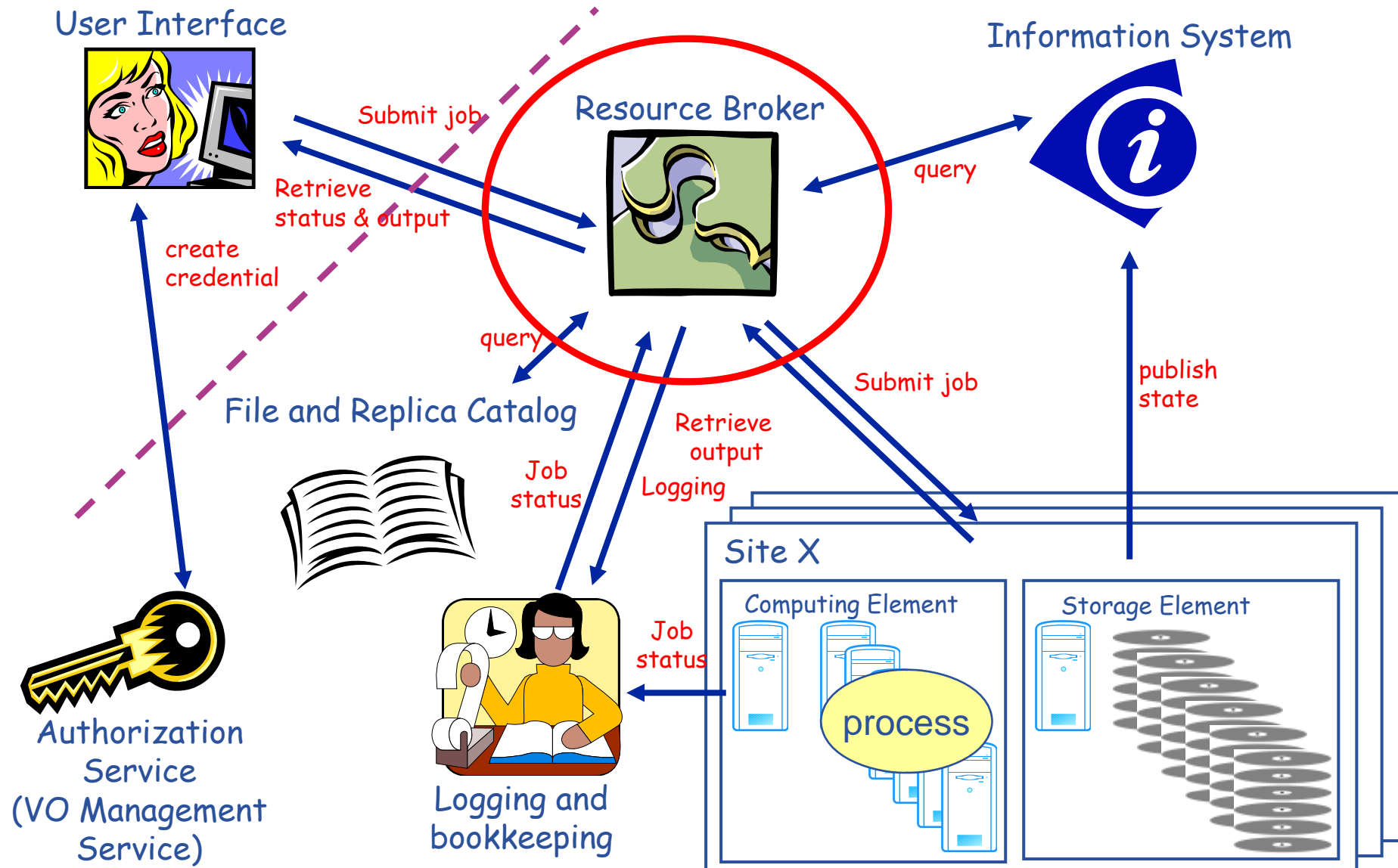


Workload Management System

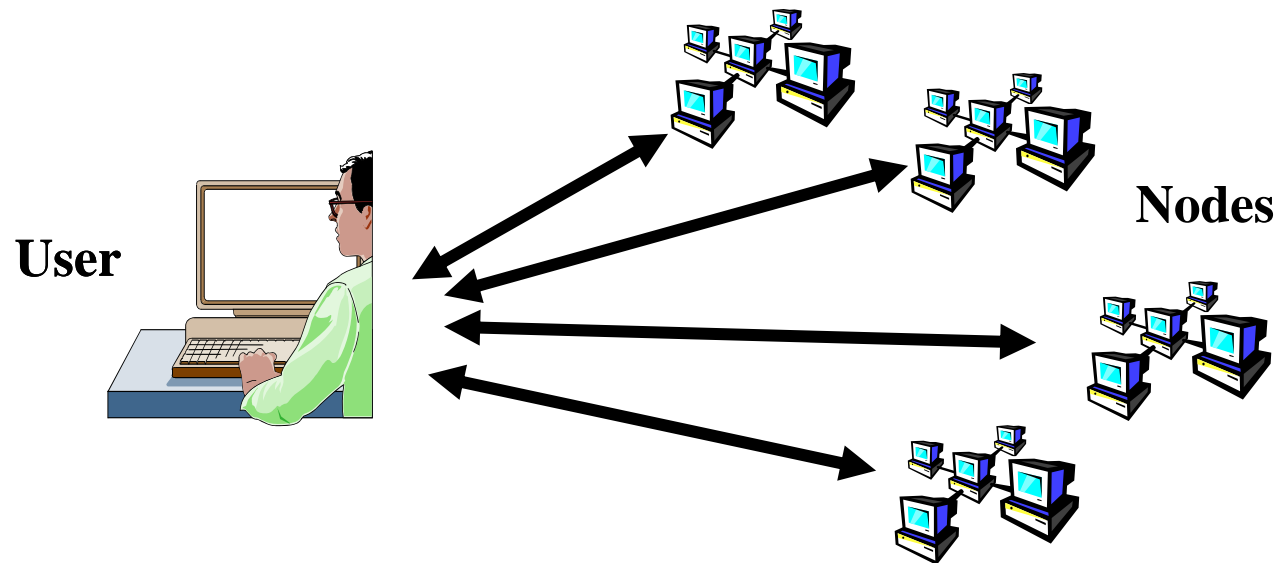
Gergely Sipos
sipos@sztaki.hu

With thanks to EGEE colleagues for many of these slides



- **What is the Workload Management System (WMS)?**
- **How are your jobs handled**
- **How do you manage jobs from command line?**
 - **Practicals**





- **Without the WMS, need direct interaction with nodes**
 - Need to know resource addresses, capabilities
- **Usually want a higher level abstraction – submit a job to a Grid not to a CE**

Why does the Workload Management System exist?

- **Grids have**
 - Many users
 - Running many jobs – a “job” = an executable you want to run
 - Where many compute nodes are available
 - Workload Management System is a software service that makes running jobs easier for the user
- **It builds on the basic grid services**
 - E.g. Authorisation, Authentication, Security, Information Systems, Job submission
- **Terminology: “Compute element”:** defined as a batch queue - One cluster can have many queues

Which CE do you want to use?

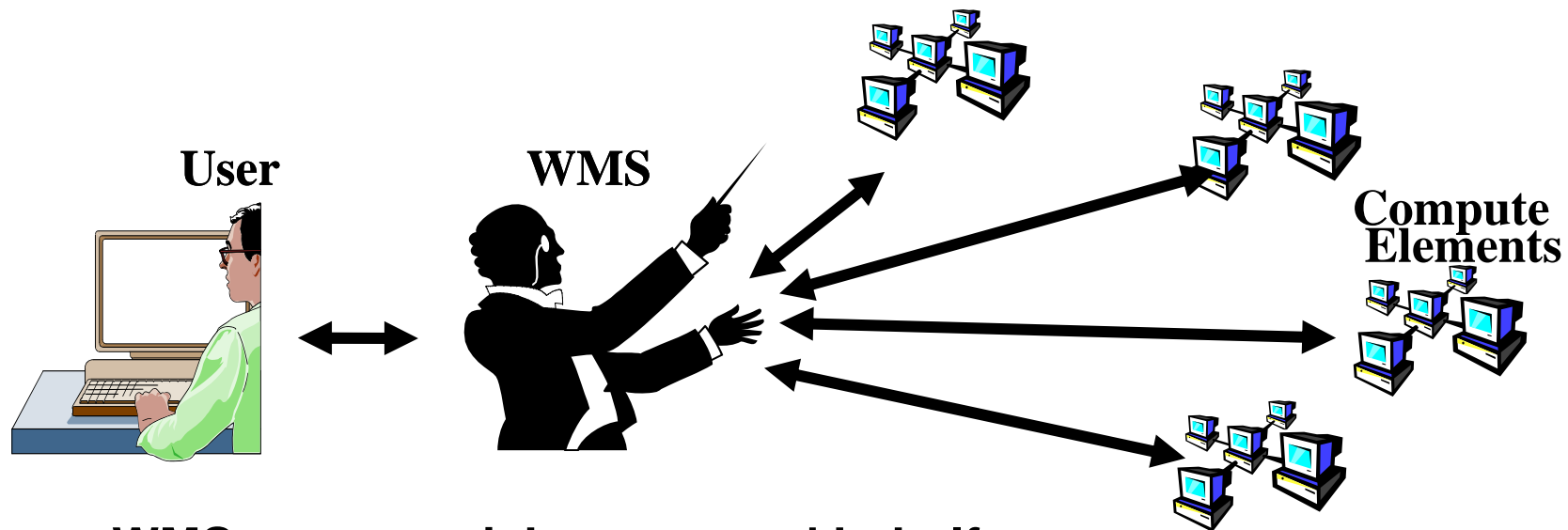
- Without the WMS, use the Information System to see what's available, then choose...

lcg-infosites --vo gilda ce

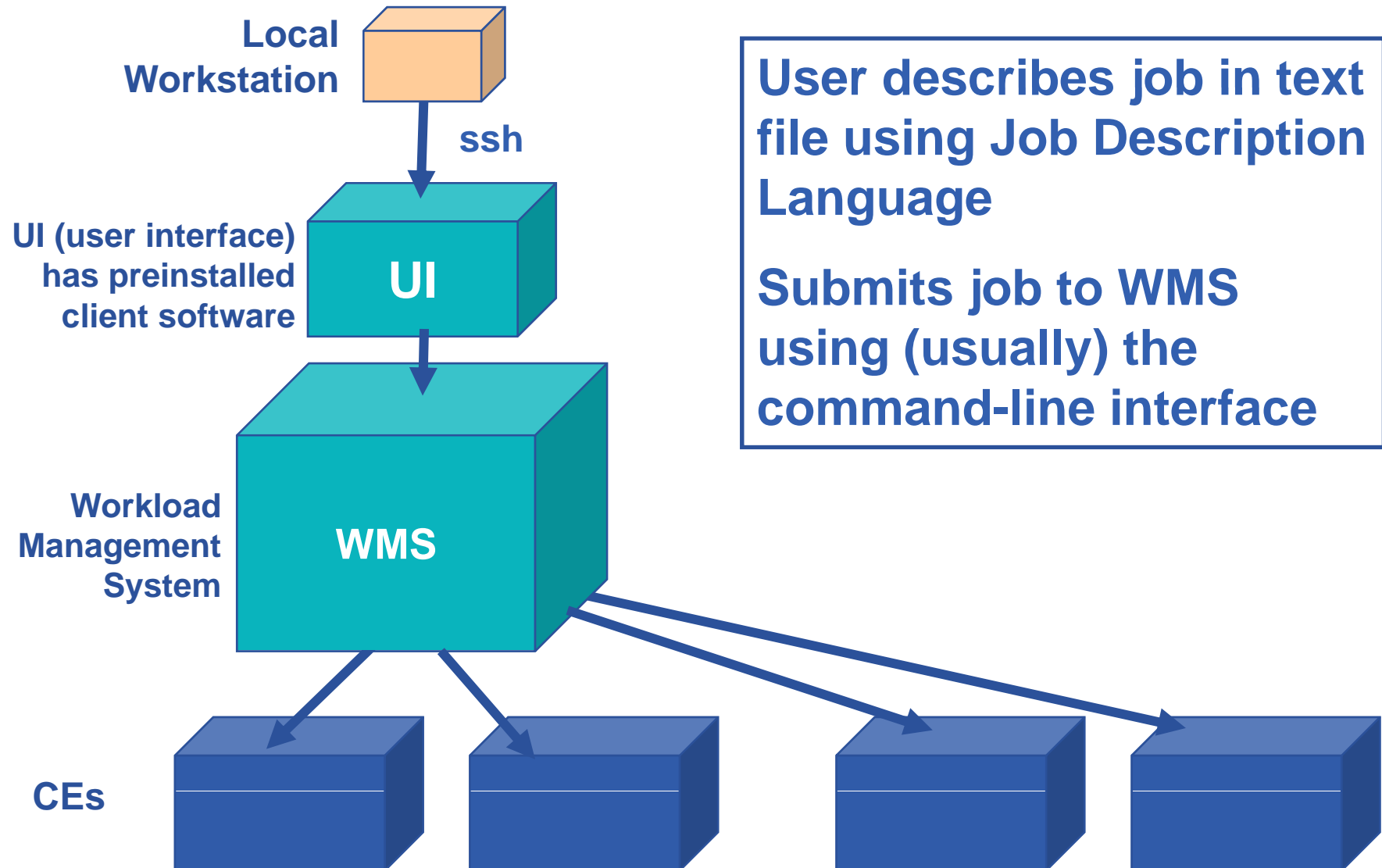
#CPU	Free	Total Jobs	Running	Waiting	ComputingElement
10	10	1	0	1	grid011f.cnaf.infn.it:2119/jobmanager-lcgpbs-short
10	10	0	0	0	grid011f.cnaf.infn.it:2119/jobmanager-lcgpbs-long
10	10	2	0	2	grid011f.cnaf.infn.it:2119/jobmanager-lcgpbs-infinite
48	48	0	0	0	grid010.ct.infn.it:2119/jobmanager-lcgpbs-short
48	48	0	0	0	grid010.ct.infn.it:2119/jobmanager-lcgpbs-long
48	48	0	0	0	grid010.ct.infn.it:2119/jobmanager-lcgpbs-infinite

.....[30% shown].

- **WMS does this for you!**
 - chooses CE for each job, balances workload, manages jobs and their files



- **WMS manages jobs on users' behalf**
 - User doesn't decide where jobs are run
 - User defines the job and its requirements, WMS matches this with available CEs
- **Effect:**
 - Easier submission
 - Users insulated from change in Compute elements
 - WMS – can optimise your jobs – e.g. which CE?



User describes job in text file using Job Description Language

Submits job to WMS using (usually) the command-line interface

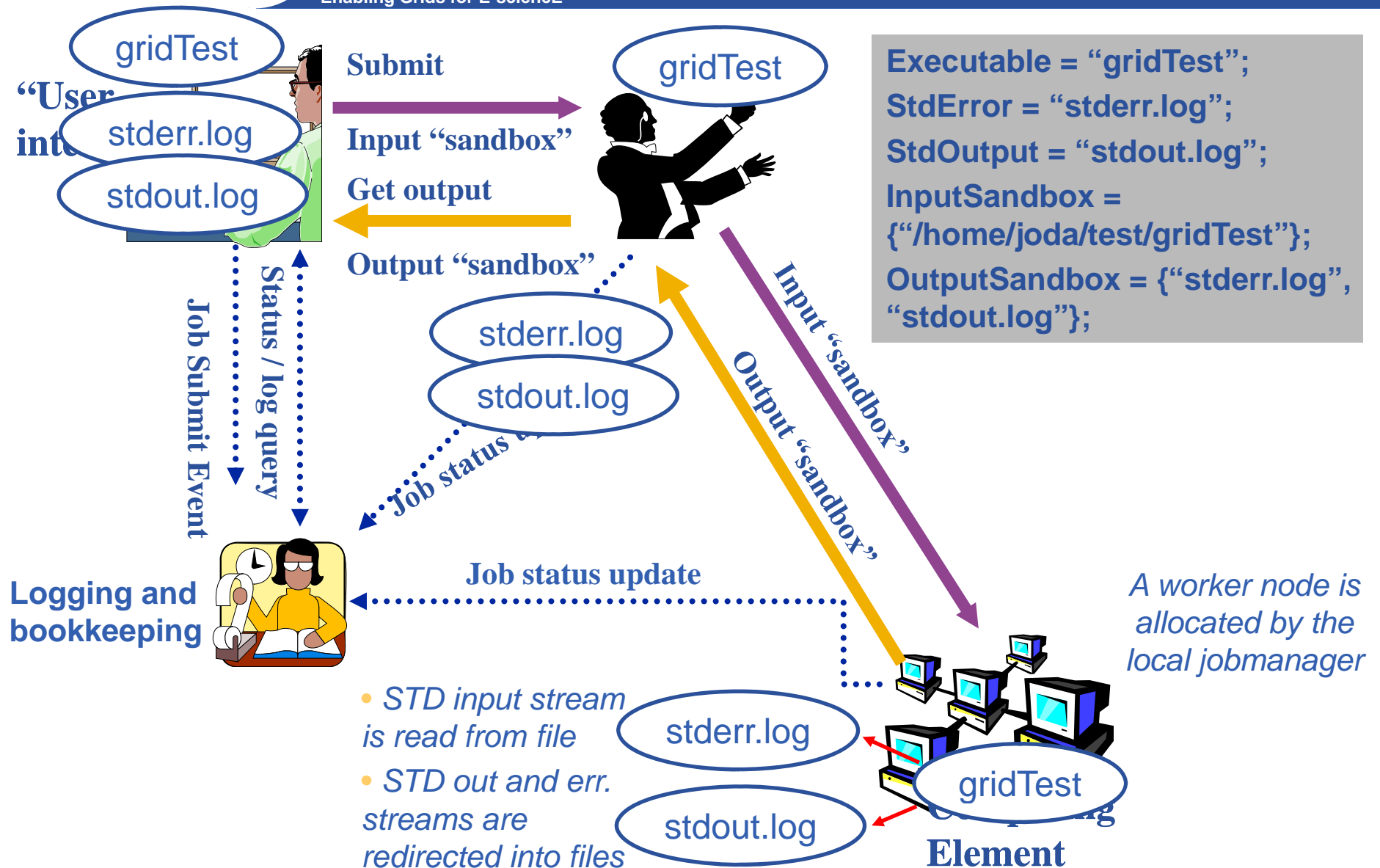
- **Jobs run in batch mode on grids.**
- **Steps in running a job on a gLite grid with WMS:**
 - 1. Create a text file in “Job Description Language”**
 - 2. Optional check: list the compute elements that match your requirements (“list match” command)**
 - 3. Submit the job ~ “glite-wms-job-submit myfile.jdl”
Non-blocking - Each job is given an id.**
 - 4. Occasionally check the status of your job**
 - 5. When “Done” retrieve output**

```
Executable = "gridTest";  
StdError = "stderr.log";  
StdOutput = "stdout.log";  
InputSandbox = {"/home/joda/test/gridTest"};  
OutputSandbox = {"stderr.log", "stdout.log"};
```

```
$ glite-wms-job-submit my.jdl
```

Returns a "job-id" used to monitor job, retrieve output...

- **Executable** – sets the name of the executable file;
- **Arguments** – command line arguments of the program;
- **StdOutput, StdError** - files for storing the standard output and error messages output;
- **InputSandbox** – set of input files needed by the program, including the executable;
- **OutputSandbox** – set of output files which will be written during the execution, including standard output and standard error output; these are sent from the CE to the WMS for you to retrieve
- **ShallowRetryCount** – in case of grid error, retry job this many times (“Shallow”: before job is running)



- **Script:**
 - No compilation is necessary
 - Can invoke binary that is statically installed on the CE
- **Binary:**
 - Must be compiled on the User Interface → binary compatibility with CEs is guaranteed
 - Statically linked → to avoid errors caused by library versions
- **Executable coming from client side**
 - Part of InputSandbox
- **Executable installed on the CE**
 - Standard software in Linux
 - VO specific software: advertised in information system
 - *Use JDL to navigate job to such a site*



WMS version	LCG-2 WMS	gLite WMS via NS gLite 3.0	gLite WMS via WMPProxy gLite 3.1+
Delegate proxy		D	glite-wms-job-delegate-proxy -d delegID
Submit	edg-job-submit [-o joblist]jdlfile	glite-job-submit [-o joblist] jdlfile	glite-wms-job-submit [-d delegID] [-a] [-o joblist] jdlfile
Status	edg-job-status [-v verbosity] [-i joblist] jobIDs	glite-job-status [-v verbosity] [-i joblist] jobIDs	glite-wms-job-status [-v verbosity] [-i joblist] jobIDs
Logging	edg-job-get-logging-info [-v verbosity] [-i joblist] jobIDs	glite-job-logging-info [-v verbosity] [-i joblist] jobIDs	glite-wms-job-logging-info [-v verbosity] [-i joblist] jobIDs
Output	edg-job-get-output [-dir outdir] [-i joblist] jobIDs	glite-job-output [-dir outdir] [-i joblist] jobIDs	glite-wms-job-output [-dir outdir] [-i joblist] jobIDs
Cancel	edg-job-cancel [-i joblist] jobID	glite-job-cancel [-i joblist] jobID	glite-wms-job-cancel [-i joblist] jobID
Compatible resources	edg-job-list-match jdlfile	glite-job-list-match jdlfile	glite-wms-job-list-match [-d delegID] [-a] jdlfile

```

Executable = "gridTest";
StdError = "std...";
StdOutput = "std...";
InputSandbox = "...Test"};
OutputSandbox = "...Test.log"};

```

WMS uses Information System to find CE

```

Requirements = other.Architecture=="INTEL" &&
  other.GlueCEInfoTotalCPUs > 480;

```

```

Rank = other.GlueCEStateTotalJobs;

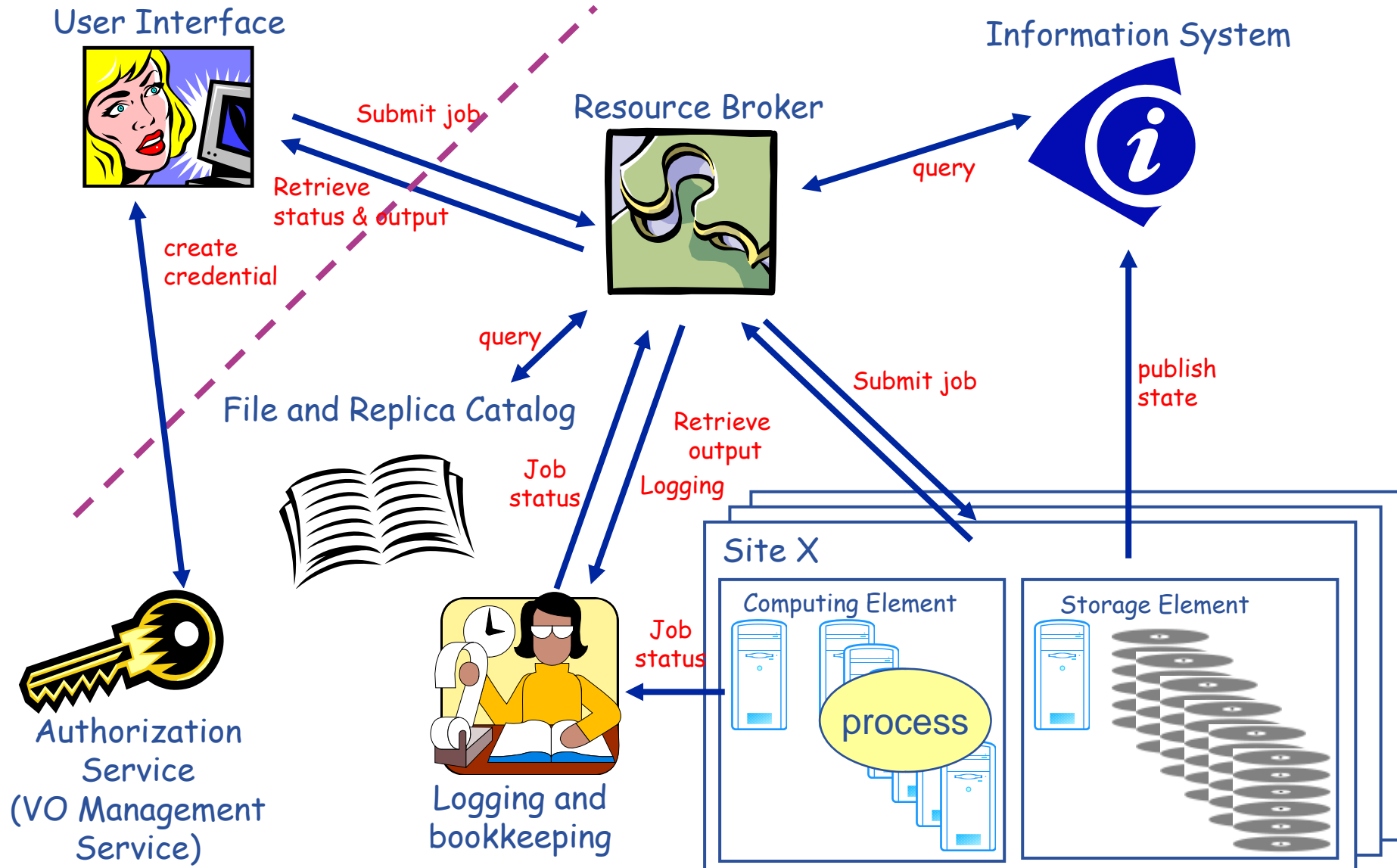
```

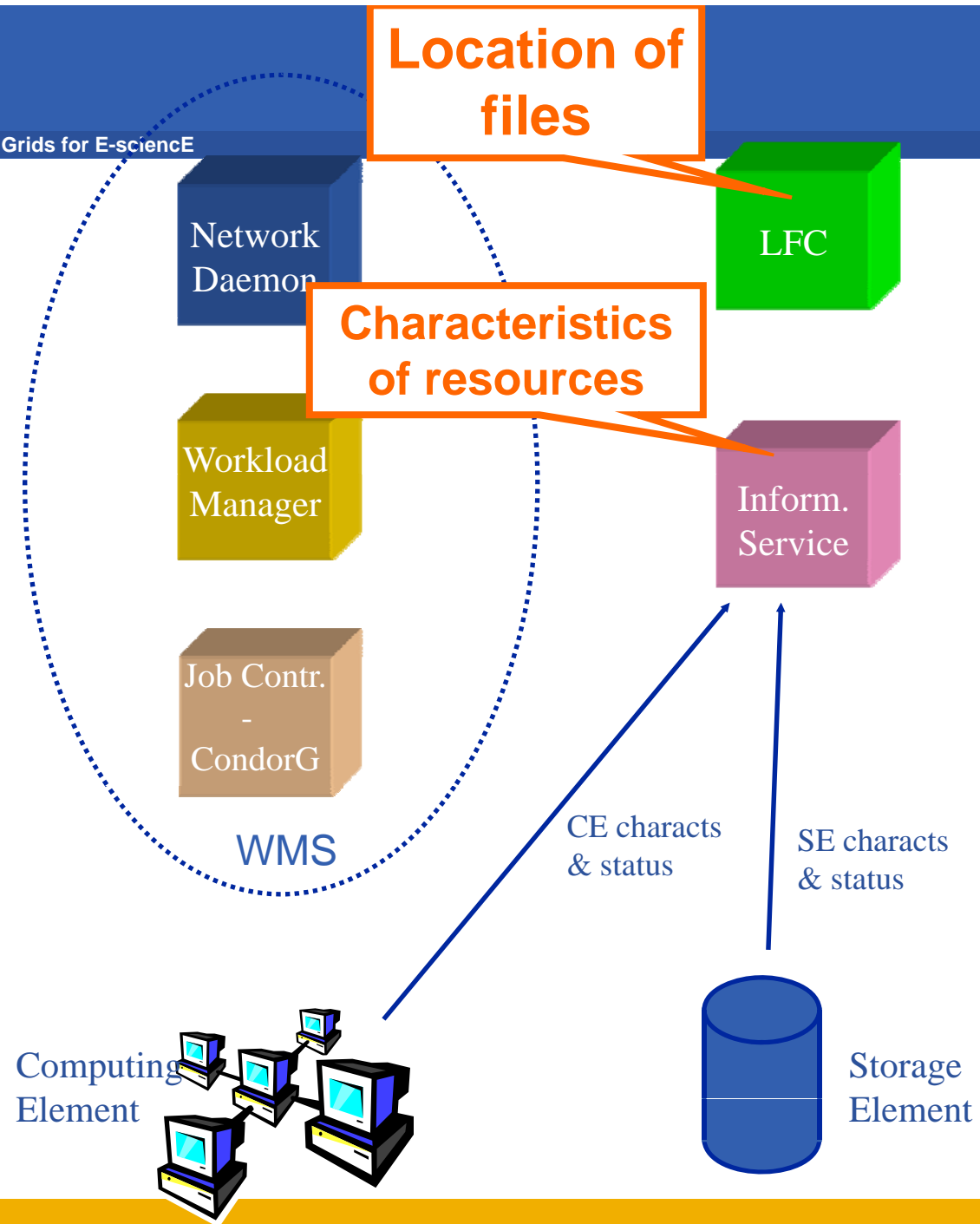
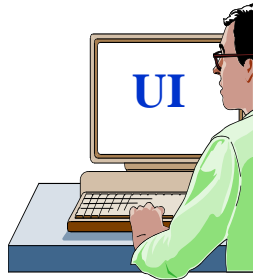
```

InputData = "lfn:/grid/gilda/sipos/test/00019.input";

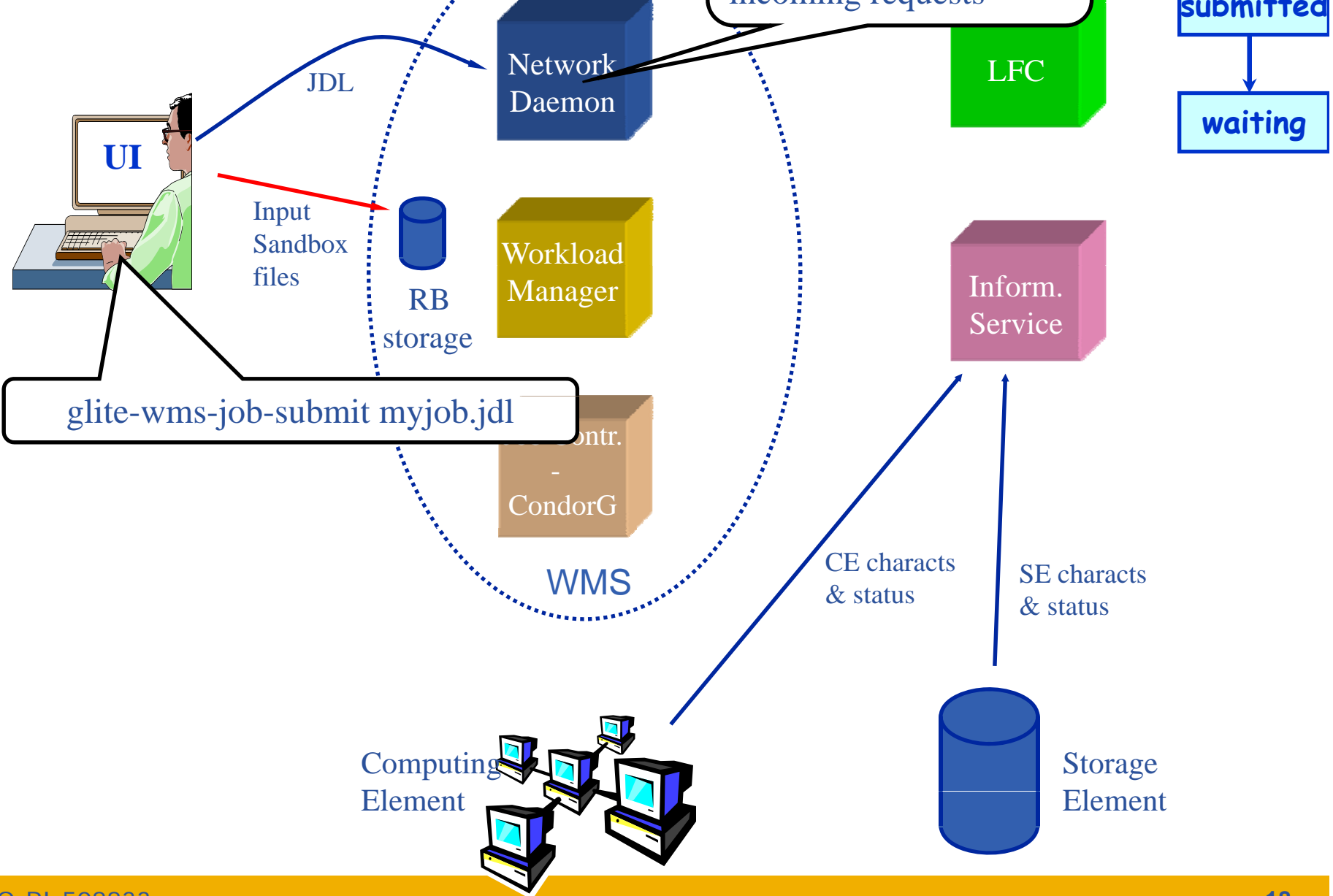
```

WMS uses File Catalog to find file location

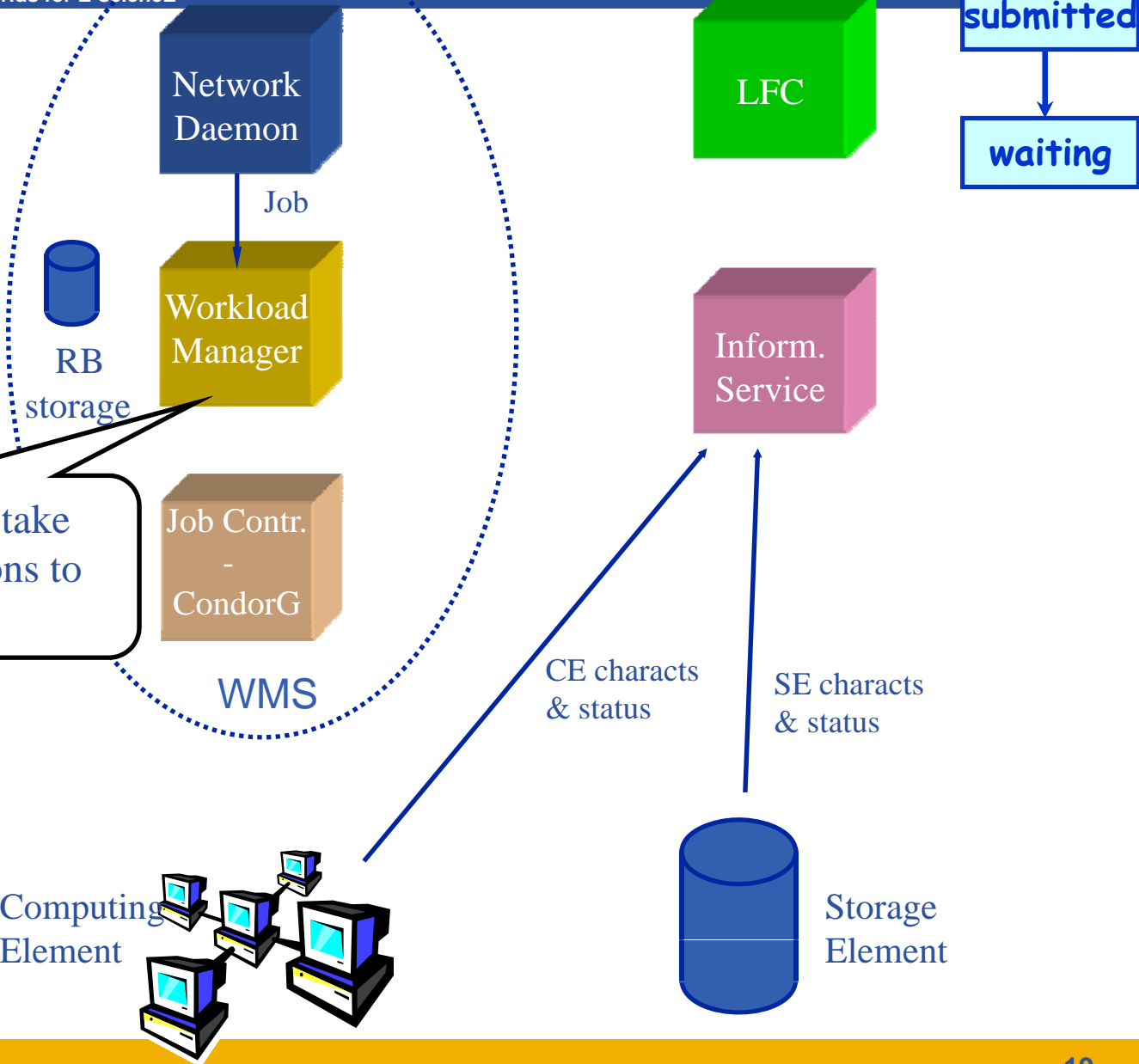
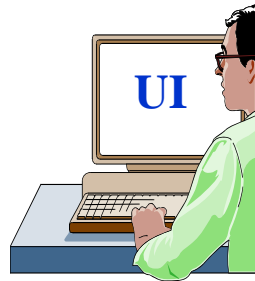


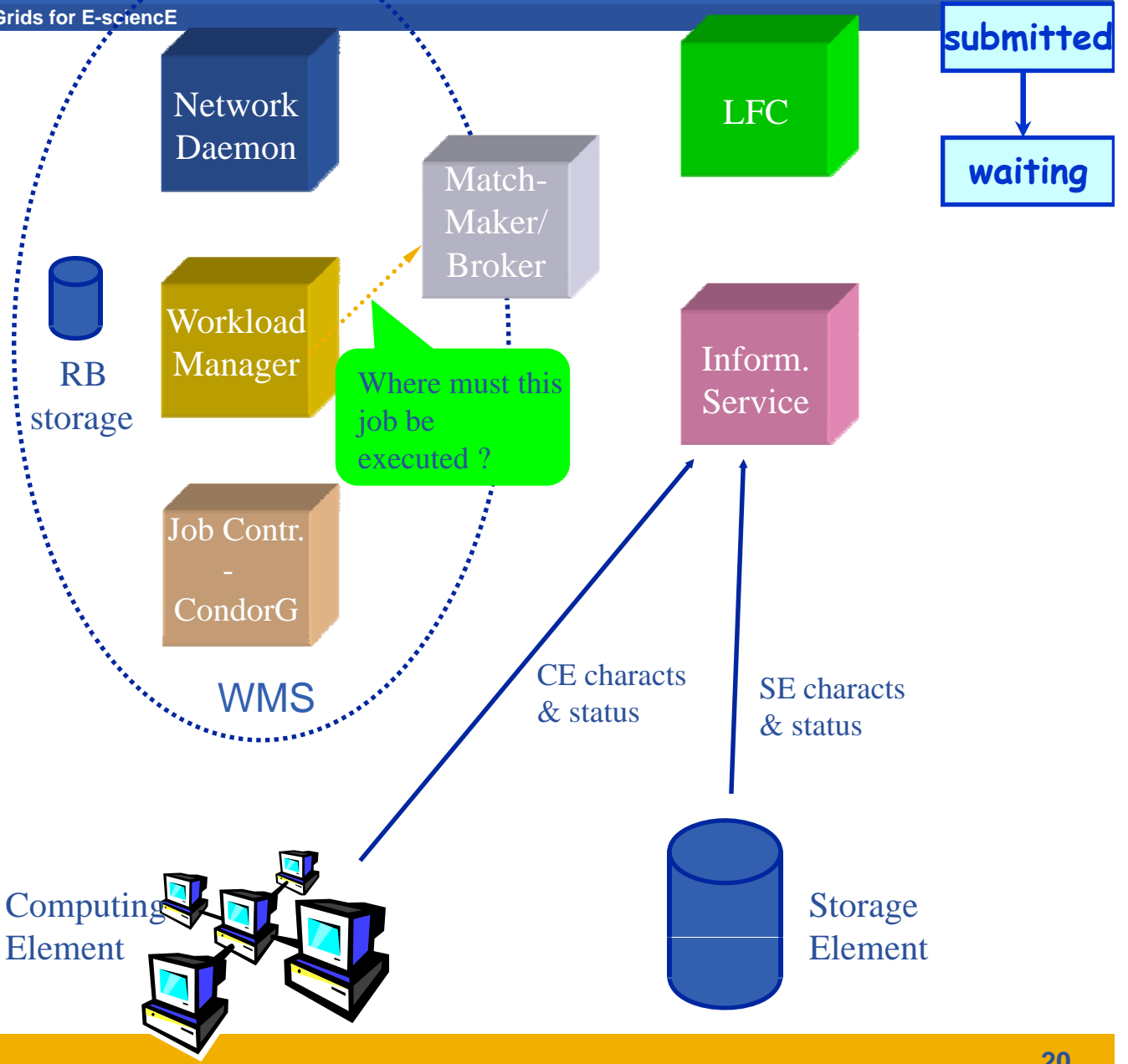
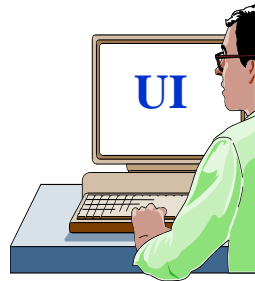


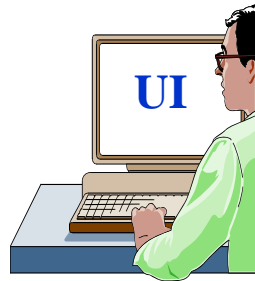
Daemon responsible for accepting incoming requests



```
glite-wms-job-submit myjob.jdl
```







Matchmaker: responsible to find the "best" CE where to submit a job

RB storage

Workload Manager

Job Contr. - CondorG

WMS

Match-Maker/ Broker

LFC

Inform. Service

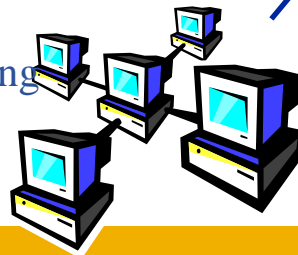
submitted

waiting

CE characts & status

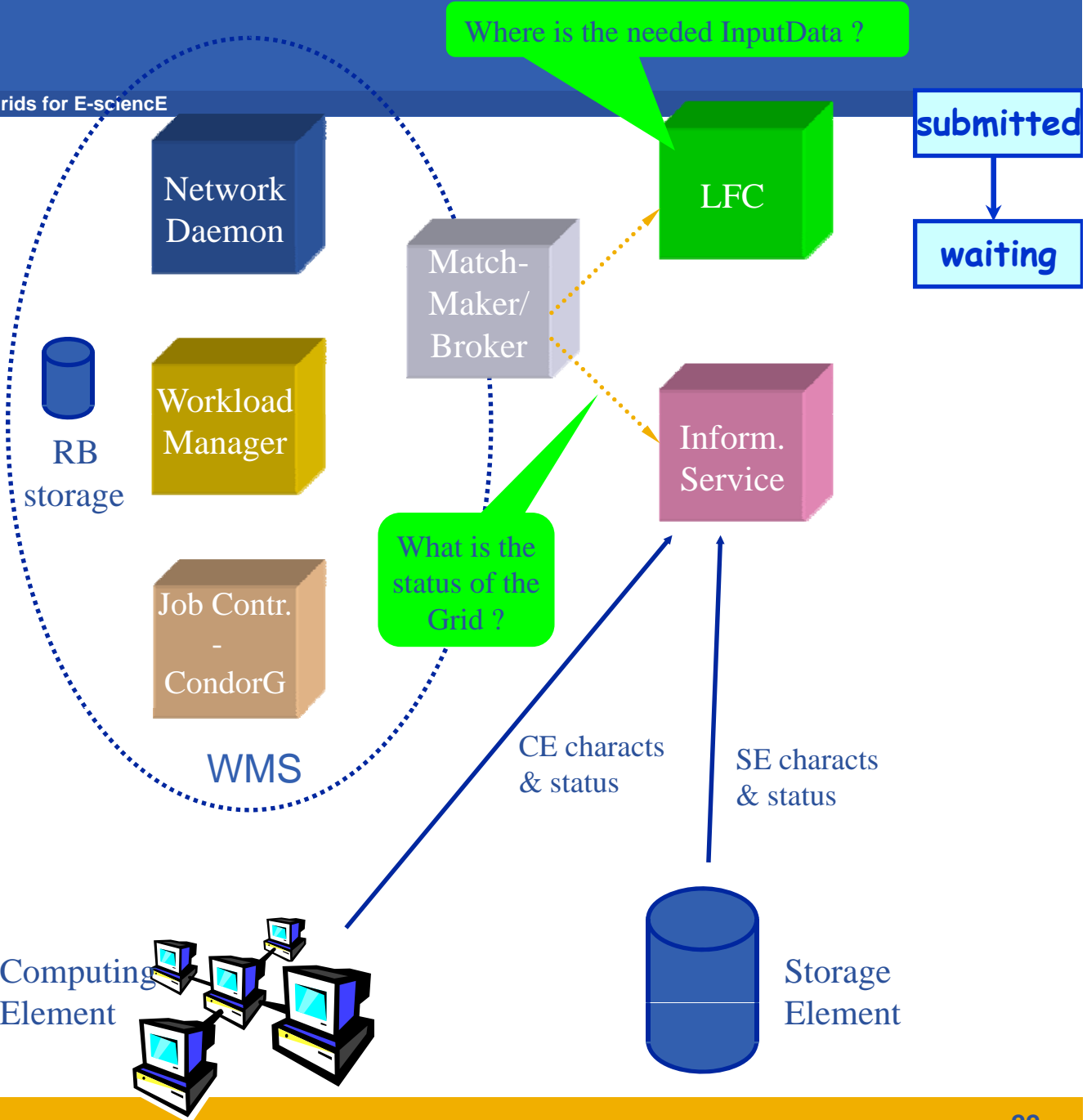
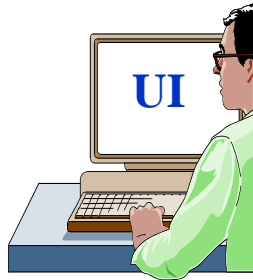
SE characts & status

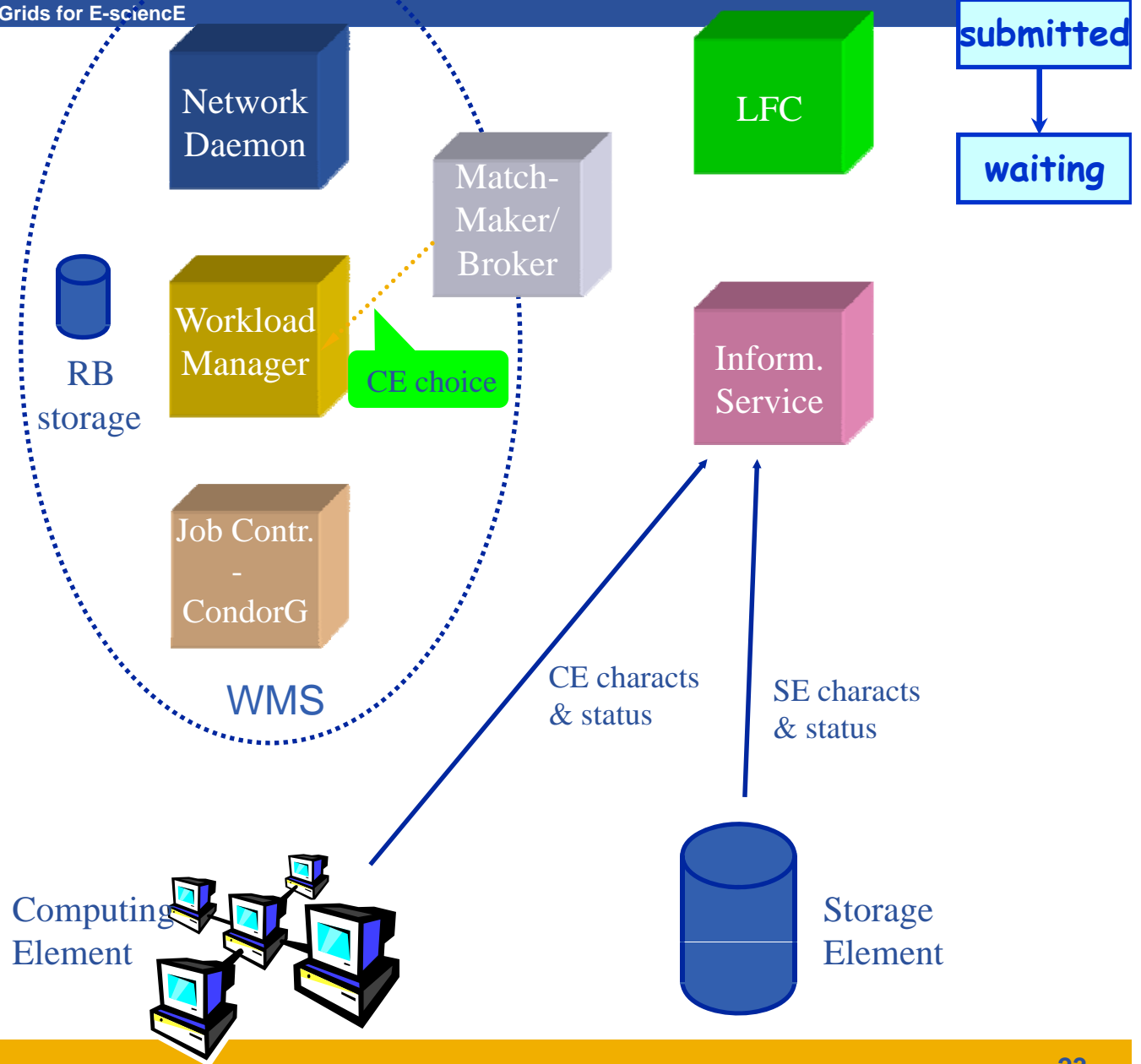
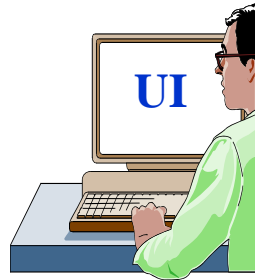
Computing Element

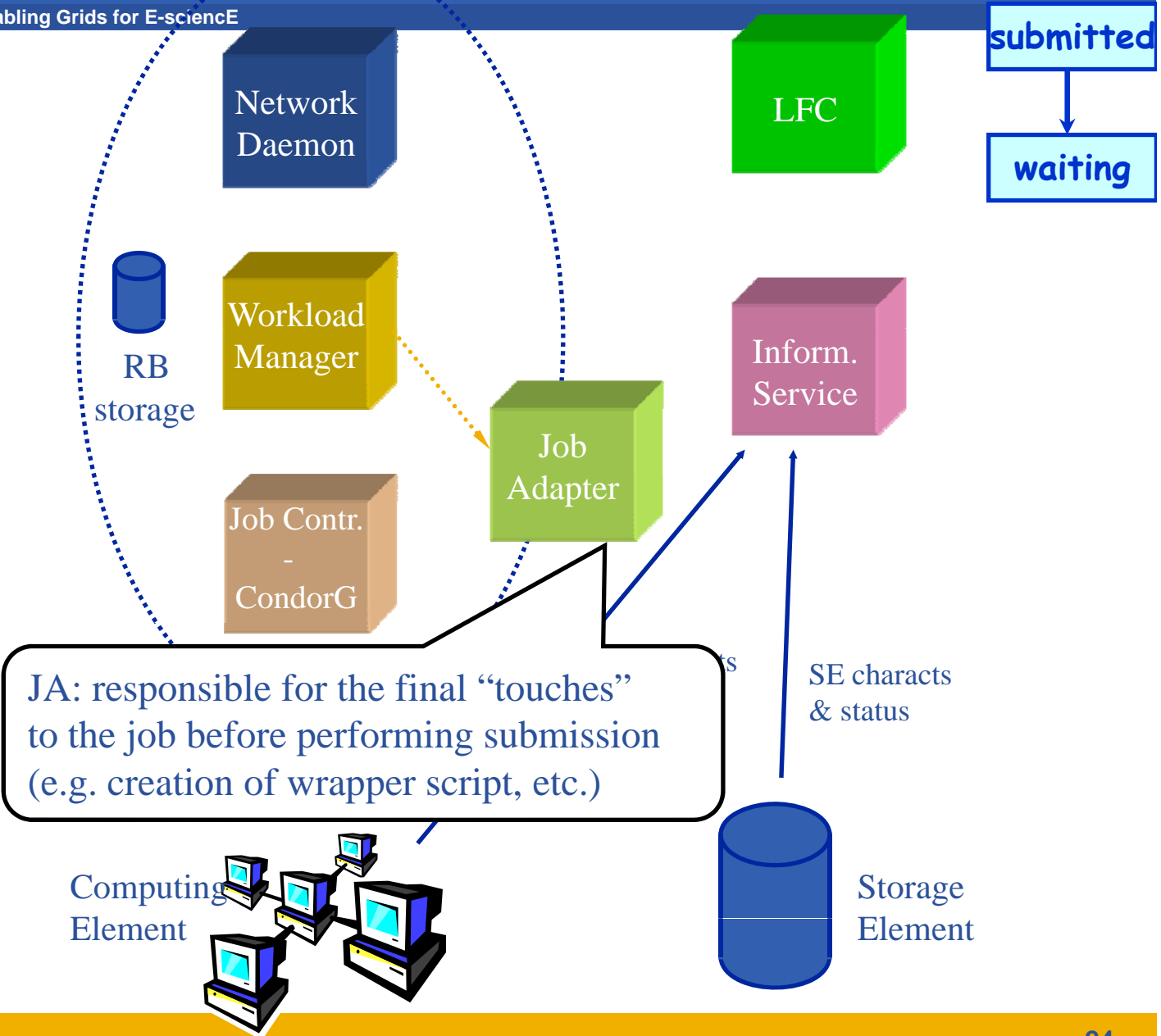


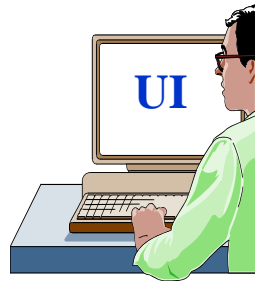
Storage Element



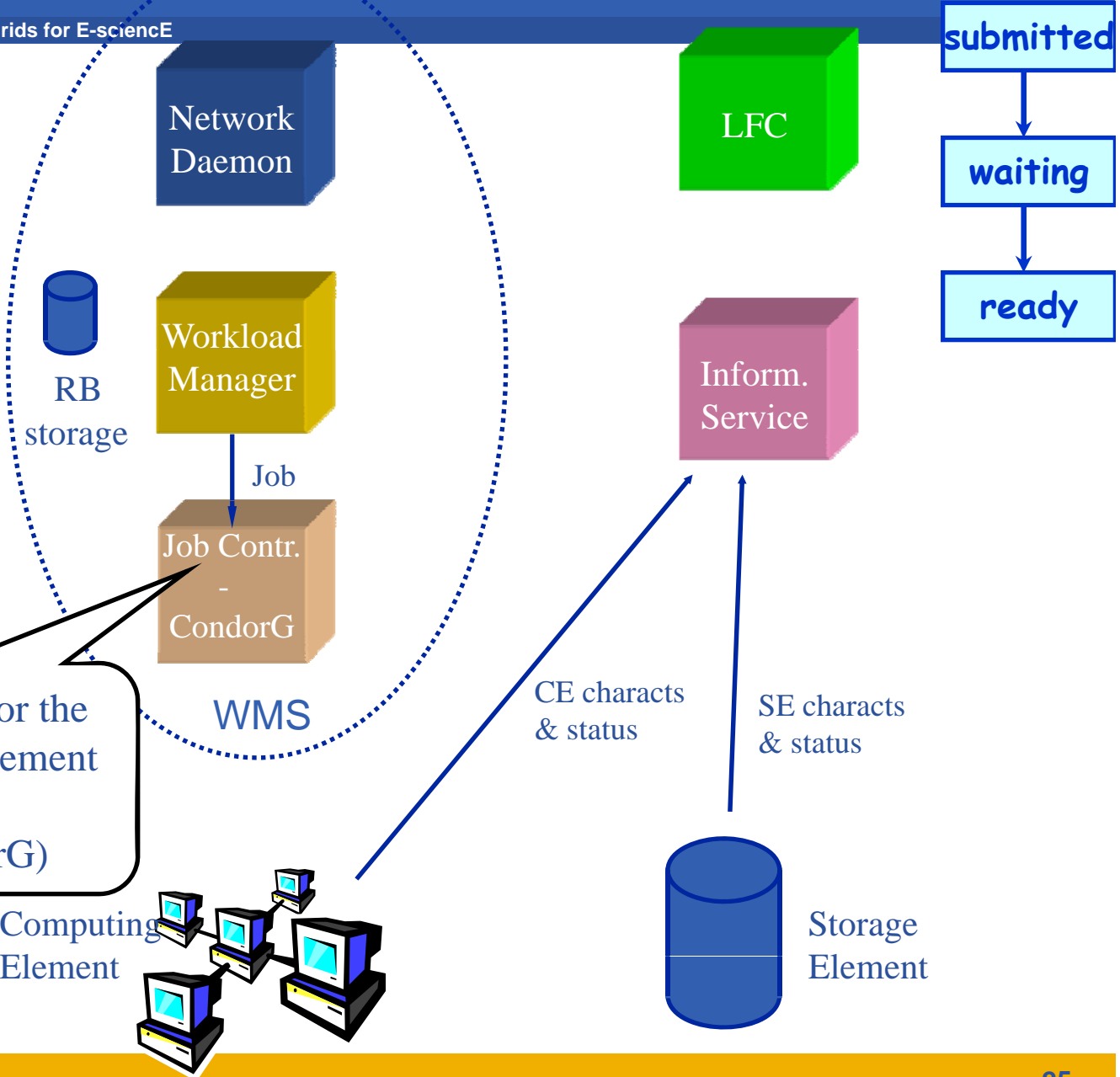


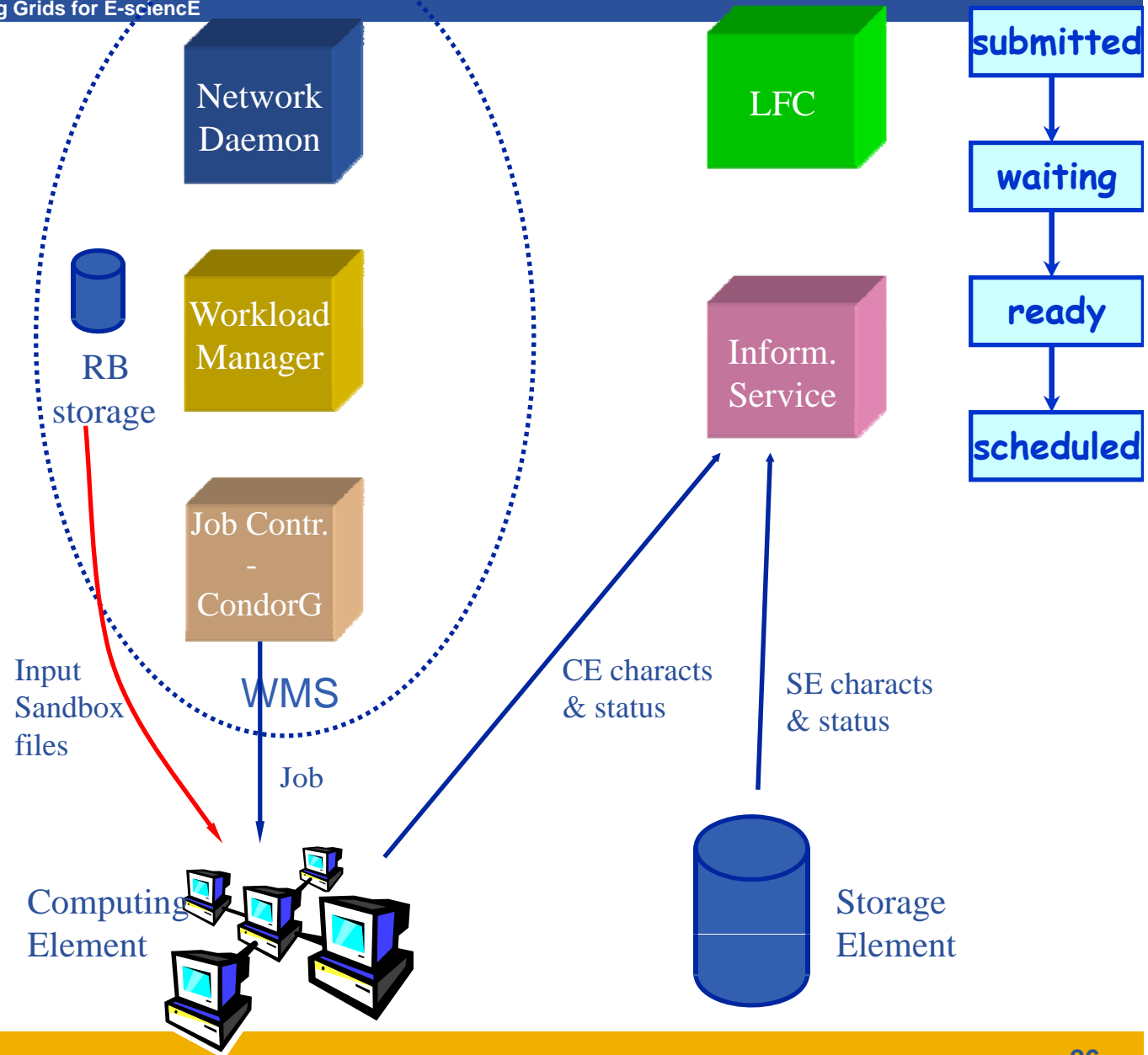
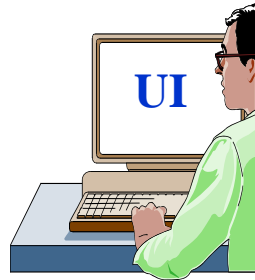


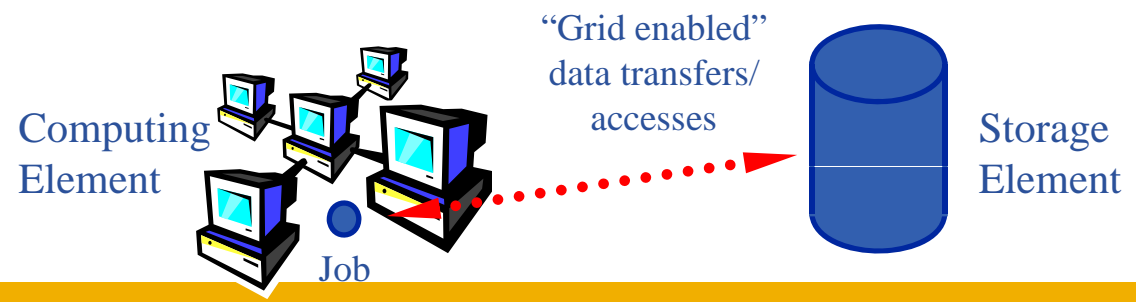
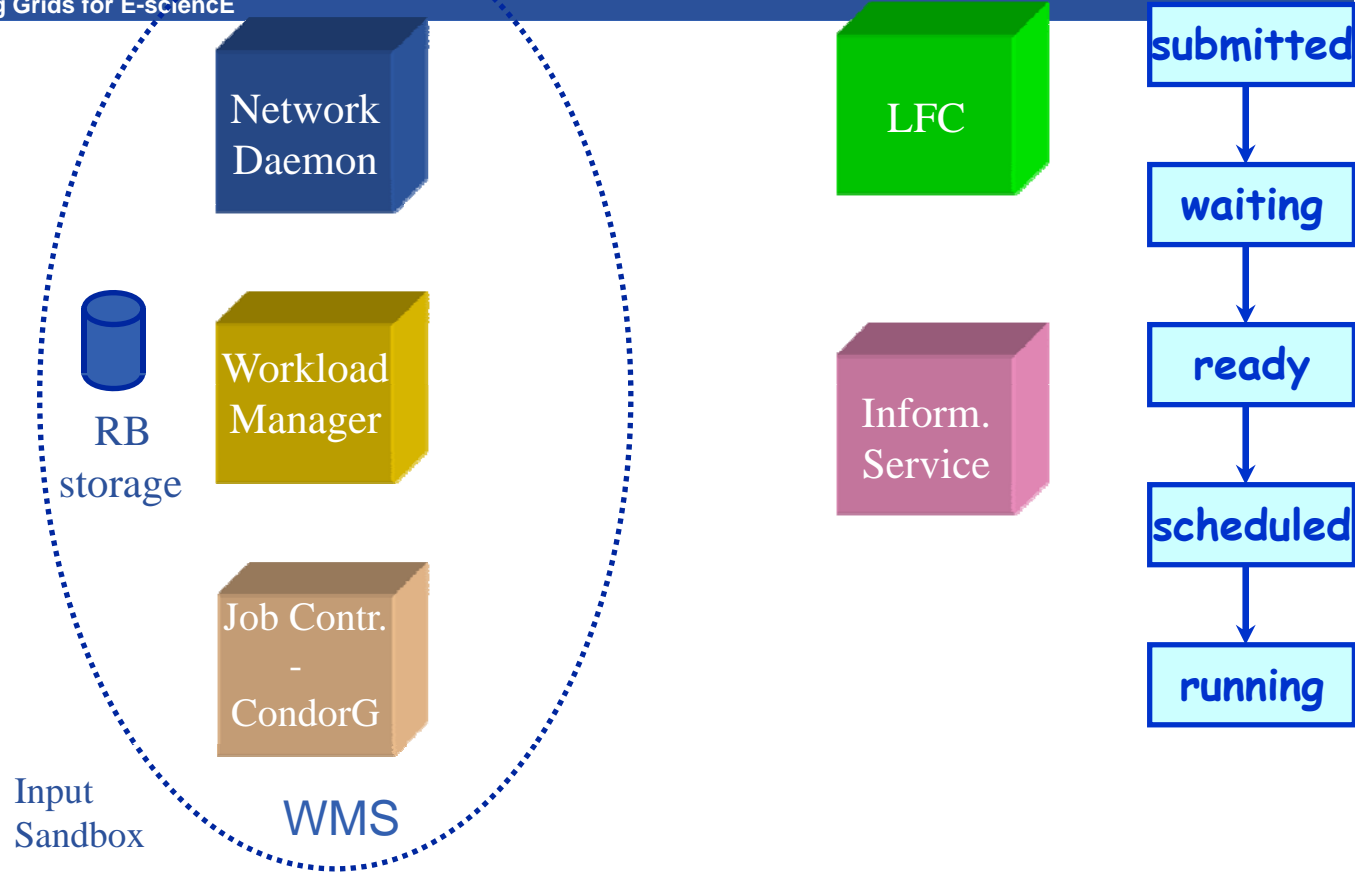
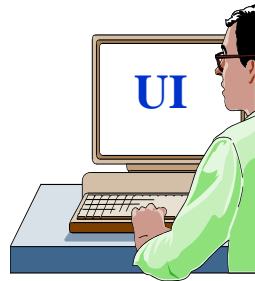


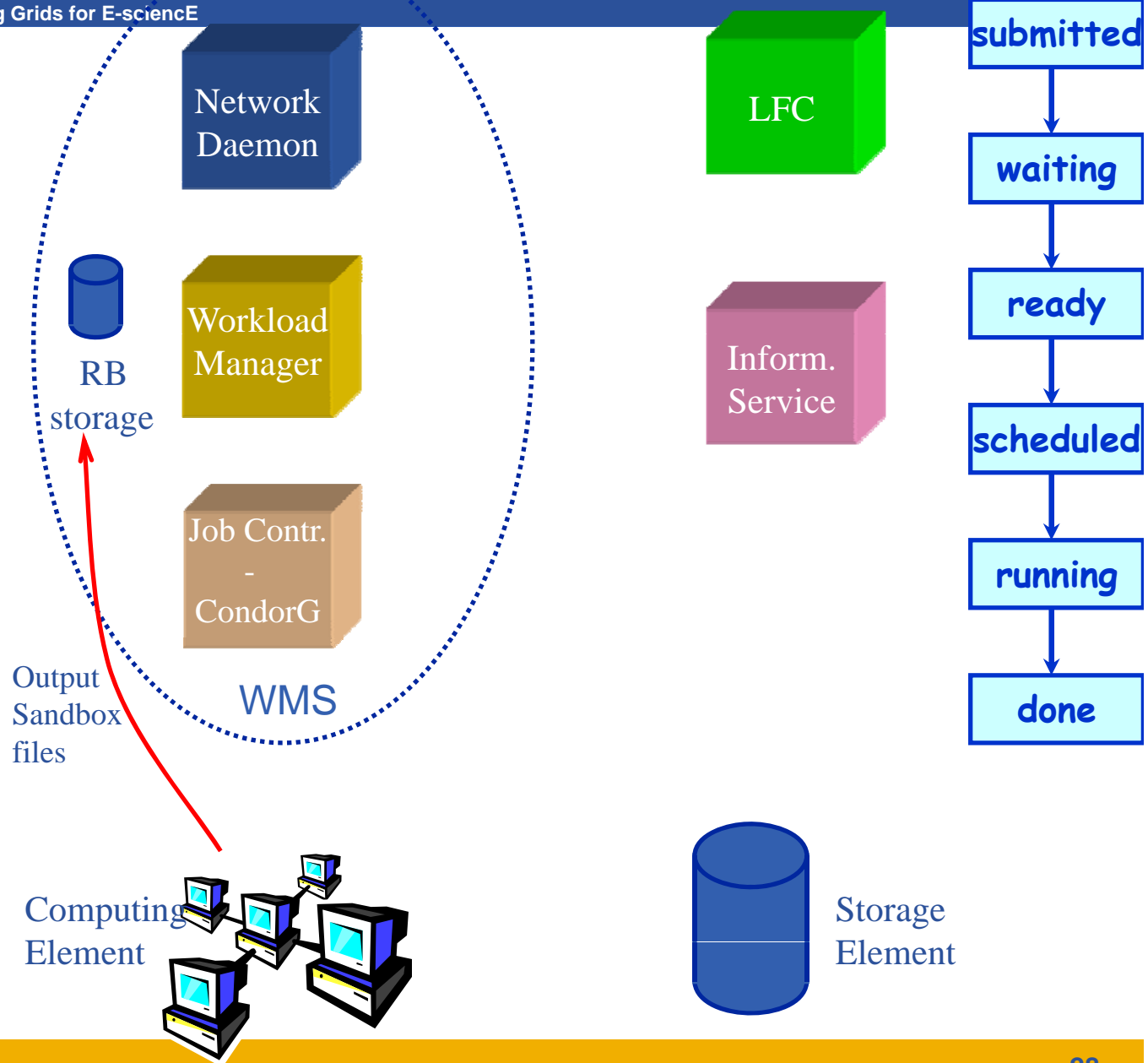
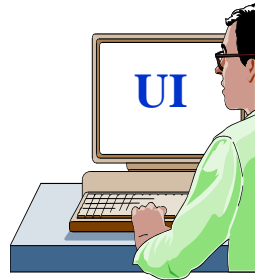


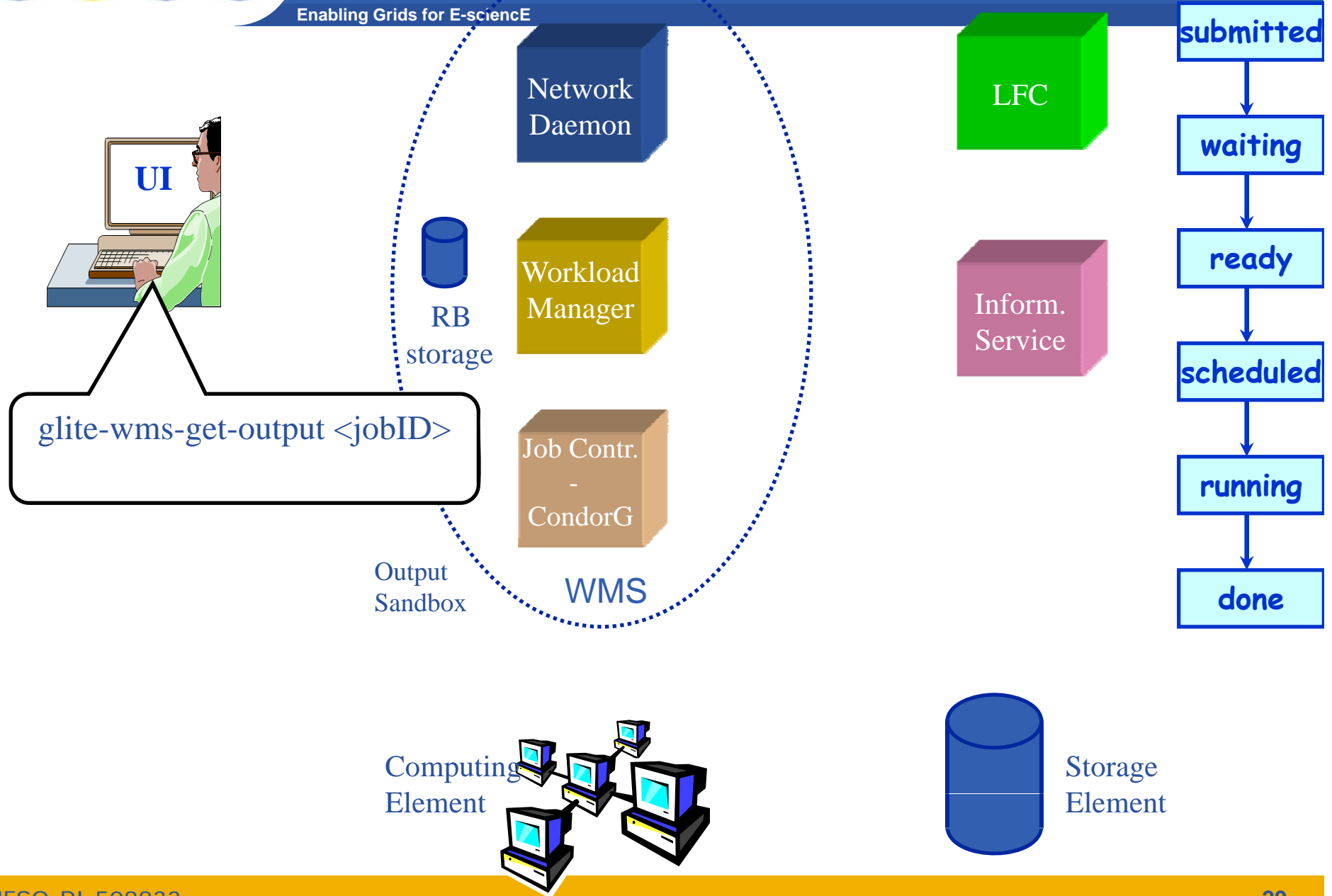
JC: responsible for the actual job management operations (done via CondorG)

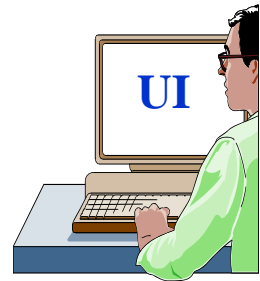












Output Sandbox files



RB storage

Network Daemon

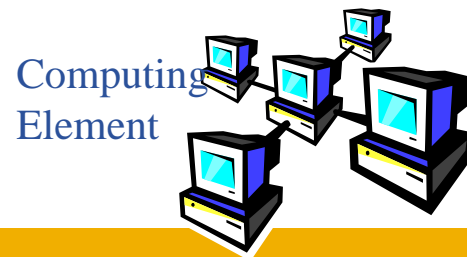
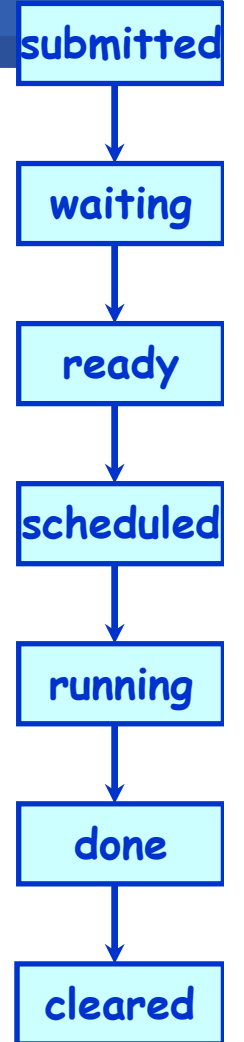
Workload Manager

Job Contr. - CondorG

WMS

LFC

Inform. Service



Computing Element

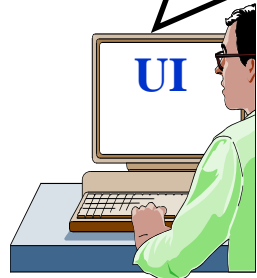


Storage Element



Job monitoring

glite-wms-job-status <jobID>
glite-wms-job-logging-info <jobID>



LB: receives and stores
job events; processes
corresponding job status

Job
status

Logging &
Bookkeeping

LB
proxy

Network
Daemon

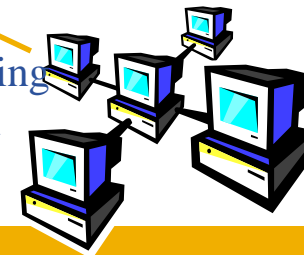
Workload
Manager

Job Contr.
-
CondorG

WMS

Log of
job events

Computing
Element



1. Meet CE requirements

2. Select CE which is close to InputData

- “Close” relationship is defined between CEs and SEs by site administrators
- “Close” is not necessarily physical distance – rather bandwidth
- “Close” typically means same site
 - SE: iceage-se-01.ct.infn.it
 - CE: iceage-ce-01.ct.infn.it:2119/jobmanager-lcgpbs-short

3. Select CE with highest rank

- **GlueCEUniqueID** – Identifier of a CE
Eliminating an erroneous CE:
`other.GlueCEUniqueID !=
"grid010.ct.infn.it:2119/jobmanager-lcgpbs-long"`
- **GlueCEInfoTotalCPUs** – max number of CPUs at a CE
`Rank = other.GlueCEInfoTotalCPUs;`
- **GlueCEStateWaitingJobs** – number of waiting jobs
- **GlueCEPolicyMaxCPUTime** – job will be killed after this number of minutes
`other.GlueCEUniqueID > 300;`
- **GlueHostMainMemoryRAMSize** – memory size

<http://glite.web.cern.ch/glite/documentation/> → JDL specification

- *Rank =*
(other.GlueCEStateWaitingJobs == 0 ? other.GlueCEStateFreeCPUs :
-other.GlueCEStateWaitingJobs);

if there are no waiting jobs,

- then the selected CE will be the one with the most free CPUs
- else the one with the least waiting jobs.

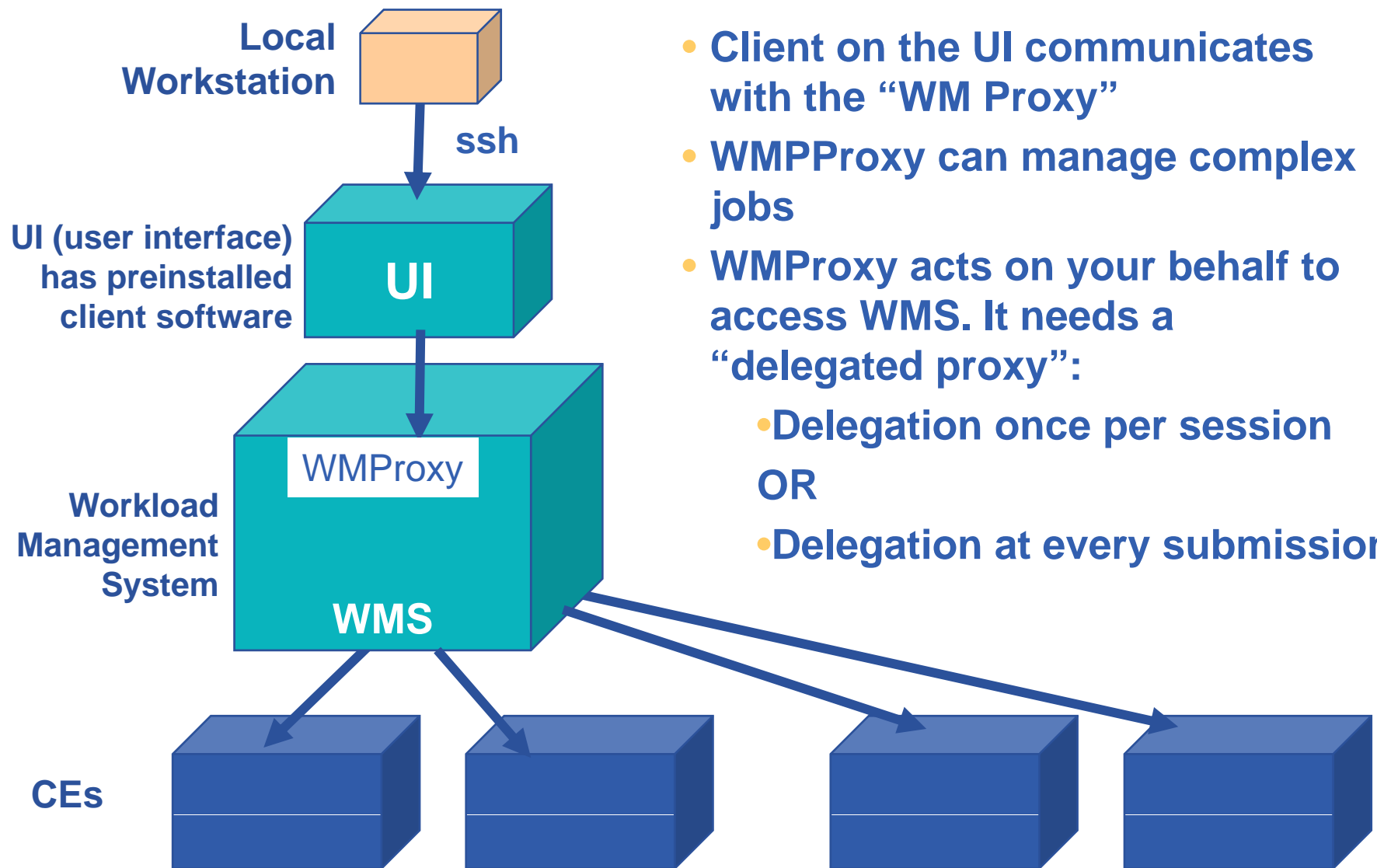
- *Requirements =*
(Member(„IDL2.1”, other.GlueHostApplicationSoftwareRunTimeEnvironment))
&& (other.GlueCEPolicyMaxWallClockTime > 10000);

CE where,

- IDL2.1 software is available
- At least 10000s can be spent on the site (waiting + running)

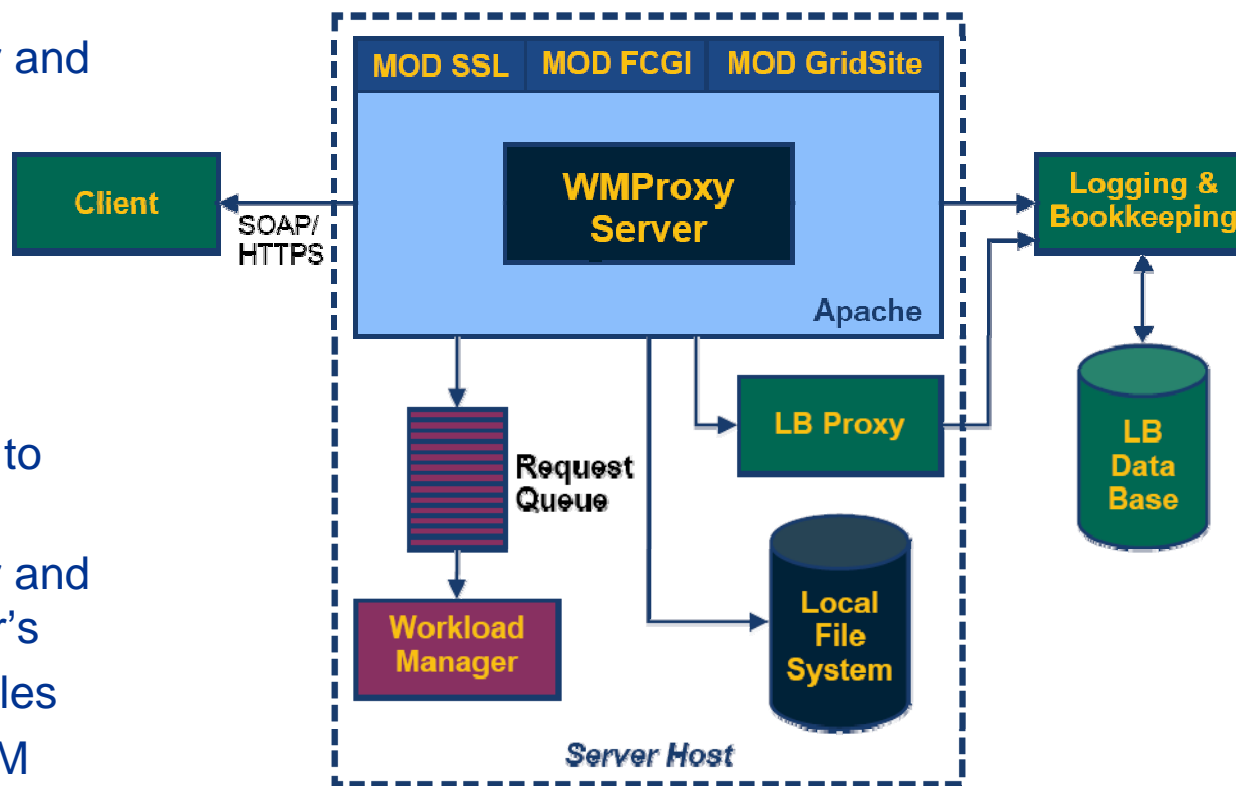
```
[
JobType = "normal";
Executable = "lexor_wrap.sh";
StdOutput = "dc2.003020.digit.A8_QCD._01730.job.log.3";
StdError = "dc2.003020.digit.A8_QCD._01730.job.log.3";
OutputSandbox {"metadata.xml",
  "lexor_wrap.log","dq_337704_stagein.log","dq_337704_stageout.log",\
  "dc2.003020.digit.A8_QCD._01730.job.log.3" };
RetryCount = 0;
Arguments = "dc2.003020.simul.A8_QCD._01730.pool.root,\
  dc2.003020.digit.A8_QCD._01730.pool.root.3 100 0";
Environment = {
  "LEXOR_WRAPPER_LOG=lexor_wrap.log","LEXOR_STAGEOUT_MAXATTEMPT=5
  ","LEXOR_STAGEOUT_INTERVAL=60","LEXOR_LCG_GFAL_INFOSYS=atlas-
  bdii.cern.ch:2170","LEXOR_T_RELEASE=8.0.7","LEXOR_T_PACKAGE=8.0.7.5/Job
  Transforms","LEXOR_T_BASEDIR=JobTransforms-08-00-07-
  05","LEXOR_TRANSFORMATION=share/dc2.g4digit.trf","LEXOR_STAGEIN_LOG=d
  q_337704_stagein.log","LEXOR_STAGEIN_SCRIPT=dq_337704_stagein.sh","LE XO
  R_STAGEOUT_LOG=dq_337704_stageout.log","LEXOR_STAGEOUT_SCRIPT=dq_3
  37704_stageout.sh" };
MyProxyServer = "lxb0727.cern.ch";
VirtualOrganisation = "atlas";
rank = -other.GlueCEStateEstimatedResponseTime
```

Flag	Meaning
SUBMITTED	submission logged in the Logging & Bookkeeping service
WAIT	job match making for resources
READY	job being sent to executing CE
SCHEDULED	job scheduled in the CE queue manager
RUNNING	job executing on a Worker Node of the selected CE queue
DONE	job terminated without grid errors
CLEARED	job output retrieved
ABORT	job aborted by middleware, check <i>reason</i>



- Client on the UI communicates with the “WM Proxy”
- WMPProxy can manage complex jobs
- WMPProxy acts on your behalf to access WMS. It needs a “delegated proxy”:
 - Delegation once per session
 - OR
 - Delegation at every submission

- WMPProxy 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



- **glite-wms-job-delegate-proxy** -d delegID
- **glite-wms-job-submit**
[-d delegID] [-a] [-o joblist] jdlfile
- **glite-wms-job-status**
[-v verbosity] [-i joblist] jobIDs
- **glite-wms-job-logging-info**
[-v verbosity] [-i joblist] jobIDs
- **glite-wms-job-output**
[-dir outdir] [-i joblist] jobIDs
- **glite-wms-job-cancel**
[-i joblist] jobID
- **glite-wms-job-list-match**
[-d delegID] [-a] jdlfile

- **JobType**
 - Normal (sequential batch job), Collection, DAG, Parametric, Interactive, MPICH, Checkpointable
- **Executable**
 - The name of the executable (absolute path)
- **Arguments**
 - Job command line arguments
- **StdInput, StdOutput, StdError**
 - Standard input/output/error of the job (stdin absolute path; stdout & stderr relative path)
- **Environment**
 - List of environment variables to be set for the binary
- **InputSandbox**
 - List of files on the UI local disk needed by the job for running
 - The listed files will be staged to the remote resource
- **OutputSandbox**
 - List of files, generated by the job, which have to be retrieved
 - Files will be transferred back

- **Input Data**
 - For the broker, WMS does not transfer these files
- **Output Data**
 - For the broker, WMS does not transfer these files
- **Requirements**
 - Required CE characteristics
- **Rank**
 - “Goodness” value for compatible CEs
- **ShallowRetryCount**
 - in case of grid error, retry job this many times
 - “Shallow”: before job is running
- **RetryCount**
 - resubmit if the job failed in Running mode
 - If job fails after it has already done something (e.g. creating a Grid file) then resubmission can generate inconsistencies
- **MyProxyServer**
 - where to download proxy from in case of the existing proxy expires
 - Done by WMS

1. **Create JDL file**
2. **Create proxy**
3. **(Delegate proxy)**
 - glite-wms-job-delegate-proxy
4. **Check some CEs match your requirements:**
 - glite-wms-job-list-match
5. **Submit job**
 - glite-wms-job-submit
6. **Do something else for a while! – gLite is not written for short jobs!**
7. **Check job status - occasionally**
 - glite-wms-job-status
8. **When job is “done”, get output**
 - glite-wms-job-output

- **Login to GILDA User Interface machine**
 - Open SSH client and connect to
 - **glite-tutor.ct.infn.it**
 - *User name:* **debrecenXX**
 - *Password:* **GridDEBXX**
- **XX is your unique, two digit number**
- **Open Practical from Agenda page**
<http://indico.cern.ch/conferenceDisplay.py?confId=26383>

- **Follow links on the agenda page**

- Practical1 – Security**

- Investigate your certificate
 - Create proxy
 - Investigate your proxy
 - (Do not) delete your proxy

- Practical2 – practice commands**

- Create a simple JDL file
 - copy&paste JDL file from tutorial into a file
 - Delegate proxy (*JobID saved in file*)
 - List the CEs that can accept it
 - Submit it
 - Check its status until its done
 - Retrieve output

- **Follow links on the agenda page**

Practical3 – practice JDL

- Submit a script from client side
- Submit a binary from client side (pre-compiled by tutor on UI)
- Requirements, Ranks

Proxy delegated as part of submission

JobID printed to terminal

Spare slides:

Complex job types

New features in gLite 3.1

- **A set of independent jobs**
- **For some reason must be managed as a single unit**
- **Possible reasons:**
 - Belong to the same experiment
 - Share common input files
 - Optimize network traffic
- **Sharing of sandboxes**

[

```
Type = "collection";
```

Transfer from UI only once

```
InputSandbox = {
    "sharedFile1"; . . .; "sharedFileM" };
```

```
nodes = {
```

```
[ JobType = "Normal";
  InputSandbox = {root.InputSandbox, . . .}
  ...; ],
```

JDL of 1st job

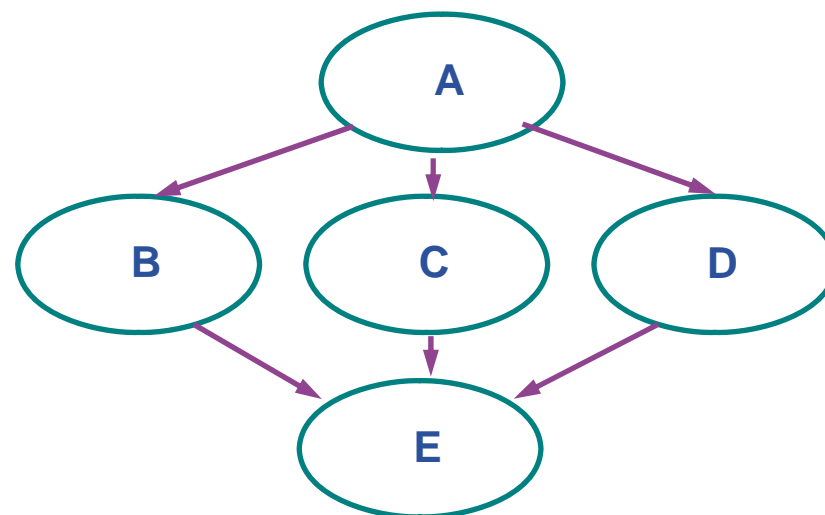
```
. . .
[ JobType = "Normal";
  ...; ],
```

JDL of Nth job

```
. . .
};
```

]

- **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
- Sharing and inheritance of sandboxes
 - **Include sandbox output in the next inputsandbox**
- Dependencies defined between pairs of jobs



```
[ Type = "dag";
```

Transfer from UI only once

```
  InputSandbox = {
    "sharedFile1"; . . . ; "sharedFileN" };
```

```
  nodes = [
```

```
    job1 = [
```

```
      description = [
```

```
        JobType = "Normal";
```

```
        . . . ; ],
```

```
    ]; . . .
```

```
    . . .
```

```
  ];
```

```
  dependencies = {
```

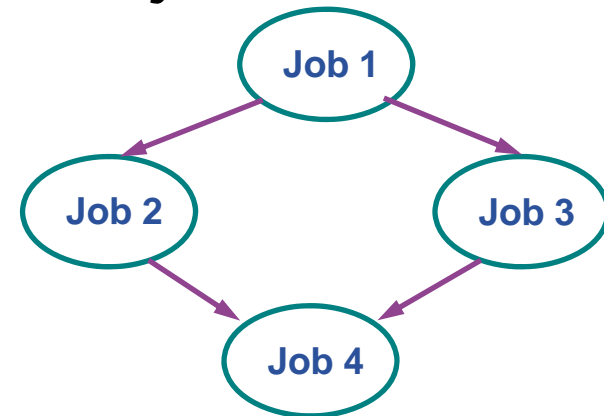
```
    {job1, {job2, job3}}, {job2, job4}, {job3, job4}
```

```
  };
```

```
};
```

```
]
```

JDL of 1st job



Graph structure

- A set of jobs generated from one JDL
- Useful where many similar (but not identical) jobs must be executed
 - Parameter study, parametric sweep applications
 - Majority of grid applications fall into this category!
- One or more parametric attributes in the JDL:
 - Use the `_PARAM_` keyword
 - E.g. `InputSandbox = "input_PARAM_";`

```
[  
  Type = "Parametric";  
  . . .  
  
  ParameterStart = 0;  
  ParameterStep = 2;  
  Parameters= 6;    →  _PARAM_ run from 0 to 10  
  
  Arguments = "inputfigure PARAM .jpg";  
  StdOutput = "transformed _PARAM_.jpg";  
  OutputSandbox = {" transformed _PARAM_.jpg ",...};  
  . . .  
]
```

- For simple jobs: **glite-wms-...** becoming the recommended way to use the WMS
- History:
 - Before the **glite-wms-** commands we had **glite-** commands
 - used the WMS without WMPProxy
 - Before the **glite-** commands we had
 - **edg-** commands (edg-job-submit....)
 - *European Data Grid – project before EGEE*
 - Used the “resource broker”
 - Still very widely used
 - You might see these commands still in use.
- Status
 - Complex jobs with WMPProxy: first stable version just released. Not yet in routine production use
 - Watch for news!

- **Follow links on the agenda page**

- **Practical3 – complex jobs**

- Execute a job collection
 - Execute a DAG
 - Execute parametric jobs
 - A bit of data management...

- **gLite Users Guide**
 - Follow <http://www.glite.org> and “Documentation”
- **GILDA wiki**
 - We are using some of these pages
 - <https://grid.ct.infn.it/twiki/bin/view/GILDA/>
- **EGEE Digital Library** <http://egee.lib.ed.ac.uk/>