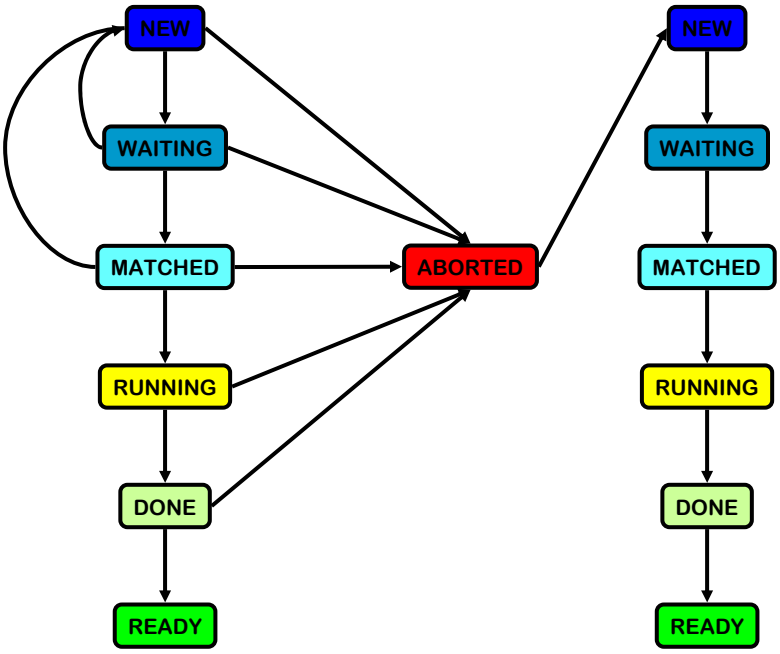
Comments to Andrei's proposal for new DIRAC States, by Ricardo.

1. Mayor points before actual Job Status description

1. In the simpler case (job without sandbox or with a "shared" sandbox), the job submission should be achieved in a single request to the server. This has several implications:
    a. The user submits a JDL that has no info about DIRAC JobID. This JDL should not be changed in any way by DIRAC. My proposal, for Job Matching purposes DIRAC builds a new DIRAC_JDL using the information provided by the user, but only those "Fields" relevant for the matching tasks are included. Ie "Owner": short VOMS alias (user name or group name) as authenticated in the DISET connection, "CPU": using some predefined queue lengths derived from the user request but avoiding a continuum, "Site": if the Data optimizer finds that the Input Data is only available at certain places.
    b. The submit method returns a Job unique identifier upon successful registration in the DB of the new Job or ERROR. This unique identifier should have the form of a Job URL since there are more than one WMS instances. I would also insist that the ID part of the Job URL is an unique ID, with none or little chance to be repeated even among different WMS instances (ie based on the timestamp plus a hash). Internally DIRAC could continue to use increasing numeric ids, if desired but they are not exposed to the "normal" user. What protocol should we use? https / dirac / diset ?
    c. If there is no sandbox (or it is "shared") the JobReceiver signals the JobOptimizer to start its processing of the Job, modifying the DIRAC_JDL if appropriated.
    d. Else, the sandbox is uploaded in a single server connection, making reference to the Job URL, and making a tar on the file if more than one file is referenced in the JDL. Once uploaded the "SandBoxServer" signals the Optimizer to start its processing of the Job.
    e. Special Jobs (with only the InputSandBox field in the JDL) can be used to upload "shared" sandboxes to DIRAC. This special job can also be used in the monitoring of all the depending jobs, to be thought about.
2. A DIRAC State Machine has to be defined. My proposal is that it is based in a reduced number of States (shown in the next figure). It is not necessary, but convenient from my point of view, to have its logic implemented in a single class. I support to include a new Job Attribute, "Minor State" for internal DIRAC sub States inside a reduce number of States. Furthermore another Job Attribute should be defined to hold the "Application State", these are those set by the user Application to allow the user a proper monitoring of its own code. The proposed list of DIRAC "Main/Mayor" states is:
    a. **NEW**: From the moment the Job enters the system until the Optimizer is signalled to introduce the job in a TaskQueue.
    b. **WAITING**: From the moment the Optimizer is signalled until it is matched by a CE resource via JobMatcher.



c. **MATCHED**: From the moment the Job is assigned to a given CE until the User Jobs starts to run on the WN.

d. **RUNNING**: From the moment the User Job starts to run until it finishes either successfully or in failure.

e. **DONE**: From the moment the User Job Execution ends until it is ready for the retrieval of its output: Boogkeeping info is sent, Output SandBox is uploaded, and Output DATA is transferred to requested destination.

f. **READY**: From the moment the Job is ready for the user to retrieve its output until it is Removed from the system.

g. **ABORTED**: Is the final State the Job reaches upon an Error condition, or an explicit Kill command. An Error in the execution of the User Job allows reaching the READY State, keeping the error condition on the Application Status Primary Parameter (or job Attribute).

Jobs can only be “Re-Scheduled”, using the same ID, if they are Waiting or Matched. If they have start Running or are Aborted, upon Re-Schedule a new DIRAC Job is created and thus a new Job ID is assigned. Should this new job inherit the History? Probably yes.

3. Minor States or DIRAC Sub-States. This does not intend to be a complete list, but will try to identify as many as I can (going over Andrei’s list). The first Minor State should report how the Job got to that state; other ones can represent a transition or a temporary situation.

Major State	Minor State	Comment
NEW	Job Inserted in WMS	The User Submits a new job to the System
	SandBox Uploaded	The Job SandBox is uploaded
	Job Rescheduled	Job has been rescheduled (keeps ID)
	Job Resubmitted	Job has been resubmitted (gets new ID & proxy)
	Job Sent to Optimizer	Signal sent to Optimizer to insert Job into TaskQueue

Required Software version need to be checked the job is to be Aborted if required version does not match any of the existing in the Repository. This will avoid entering into an infinite loop. Same thing with Input Data, Site or CPU requirements, they should match one of the match some known resource or TaskQueue.

Major State	Minor State	Comment
WAITING	Job Searching Software	Checking Required Software
	Job Matching InputData	Data Optimizer looking up Input Data
	Job Matching TaskQueue	Prioritizer placing Job in TaskQueue
	Job Inserted in TaskQueue	Job Ready to be Matched
	Pilot Agent Submitted	Agent Director Submits a Pilot
	Proxy Expired	Agent Director Finds an expired proxy

Major State	Minor State	Comment
MATCHED	Job Matched	Job Matched by Job Agent
	Job Received	Job Received by Job Agent
	Installing Software	Installing Requiered version of Software
	Job Scheduled	Job sent to Local Batch System
	Job Queued	Job Waiting in Local Queue
	Job Rescheduled	Job Agent Failed some of the above
	Job Started	Start Execution of Job Wrapper
	Getting SandBox	Start Download of SandBox
	SandBox Downloaded	SandBox sucessfully downloaded

Up to this point it is possible to reschedule the job using the same ID. Later on, errors are fatal and jobs have to be resubmitted, getting a new ID. Now the User Job starts its execution.

Major State	Minor State	Comment
RUNNING	Application Started	The Wrapper starts the User Job Thread
	Step 1 Started	Starts Execution of Job Step 1
	Module X Started	Starts Execution of Module X
	Module X Ended	Ends Execution of Module X
	Module Y Started	Starts Execution of Module Y
	…	…
	Step N Ended	Ends Execution of Step N
	Application Ended	Ends Job Execution
	Application Stalled	The Wrapper detects no Activity on the Application
	Job Stalled	The WMS detects no heart-beat from the Job
	Wrapper(T): X CPU, Y MB	CPU and Disk consumed by Application in last T interval
	Job Killed	Kill request sent to WMS

After the User Job (Application) has run the Job Wrapper still has to upload the Output SandBox and place the Bookkeeping and Transfer Requests.

Major State	Minor State	Comment
DONE	Application Finished Successfully	User Job Finished without Errors
	Uploading SandBox	Uploading Output SandBox
	Uploading Log Files	Uploading Log Files to LogSE
	Job Finished Successfully	Job Wrapper Ends Execution
	Running Bookkeeping Agent	Starts Bookkeeping Agent
	Running Transfer Agent	Starts Transfer Agent

The end of the Job in DIRAC is defined by the upload of the Output Data, as defined in the Transfer Request set by the Job. I would propose to define an empty transfer request so that the Transfer Agent is always responsible to define the Ready state at the end of the Transfer.

Major State	Minor State	Comment
READY	Job Finished Succesfully	Job ir Ready for user to retrieve output
	SandBox Retrieved	User has retrieved SandBox
	Job Succesfully Accounted	Job info has been copied to Accounting
	Job Failed Accounting	Failure copying Job to Accounting
	Deleted	Can be removed from WMS

If an error condition is found during Job execution, including a Kill signal sent to the Job, the job will be Aborted and the State changed accordingly. The corresponding reason is reported as Minor State.

Major State	Minor State	Comment
ABORTED	Required Software not Found	Failed to find requiered version of Software
	Required InputData not Found	Failed to find required InputData in FC
	Site Not Found	Failed to find Required Site
	Wrong CPU Requirement	Could not fit CPU requirement in any TaskQueue
	Error Downloading JDL	Job Wrapper Error getting JDL from WMS
	Error Downloading SandBox	Job Wrapper Error getting Input SandBox
	Error Execution Application	The user Job returns with error code
	Error Uploading SandBox	Error during Upload of SandBox
	Error Sending Bookkeeping	Error sending Bookkeeping Record
	Error Transferring OutputData	Error During Transfer of Output Data
	Error Registering OutputData	Error During Registration of Output Data
	Job Killed	Kill signal received by Job Wrapper
	Job Succesfully Accounted	Job info has been copied to Accounting
	Job Failed Accounting	Failure copying Job to Accounting
	Deleted	Can be removed from WMS

While the Application is running, it will update the “Application Status”.

4. The user is only allowed to request the following Transitions:
    a. **SubmitJob**: to initially enter the job into the system
    b. **KillJob**: If the Job is NEW or WAITING it si immediately set to ABORTED:”Job Killed”. If the Job is MATCHED or RUNNING, an error will be returned to the next component requesting a transition that should Abort the Job. If the Job is DONE, READY or ABORTED, it has not effect.

c. **RescheduleJob**: The user can only Reschedule an ABORTED Job. Then the Job gets a new Job ID and a new proxy are uploaded (Job Resubmission). Upon different Error conditions, several DIRAC components may reschedule a Job, if the error is considered transient. In this case the same Job ID is kept.

d. **DeleteJob**: Explicit request for the Job to be removed from the WMS Data Base.