GENIE GUI Design Notes

Ian Ross

4 May 2015

1 Use cases

Completely new job

- Create a new job in a selected job directory from a selected "base" and "user" configuration.
- Start job and monitor completion status.
- Post-process results.

Modification of existing completed job

- Clone an existing job configuration.
- Add parameter modifications on top of existing configuration.
- Start job and monitor completion status.
- Post-process results.

Continuation from existing job

- Clone an existing job configuration and set existing job as restart.
- Add parameter modifications on top of existing configuration.
- Start job and monitor completion status.
- Post-process results.

Mid-job parameter modification

- Pause an existing running job.
- Add parameter modifications on top of existing configuration.
- Restart job and monitor completion status.
- Post-process results.

Cluster jobs

- Create a job as usual.
- Run the job for a while to check for correctness.
- Pause the job.
- Submit job to cluster queue.

2 Requirements

2.1 Basic functionality

The idea here is to produce a simple GUI for configuring and managing GENIE jobs. It should allow a user to:

- quickly set up new jobs, either from scratch or based on existing jobs;
- start, stop and restart jobs;
- modify job parameters from the selected configuration (whatever that means in detail);
- monitor running jobs, both via the basic GENIE output and with real-time graphs of output variables;
- manage existing jobs: delete, rename, move, archive;
- access shared jobs (to clone or restart from existing shared experiments);
- manage job execution on clusters in a straightforward way.

2.2 Questions and possible answers

How to manage communication between GUI and running jobs?

GUI writes small command files into job directory; model looks for the existence of a command file on each timestep. Model writes response into small response files that the GUI scans for periodically (for running jobs). Writing and reading small files avoids any issues with file buffering.

There should probably be a configuration flag (`gui_control` in `data_genie`?) to switch this functionality on and off – jobs configured via the GUI can have this on. (Alternatively, this could be enabled all the time, which would allow the GUI to control *any* GENIE job... I like that solution better.)

What happens when a job is stopped and restarted?

Is it the same job as far as GENIE is concerned or should the GUI create a new copy of the job?

When a job is stopped, the model writes restart files with names based on the current timestep. When a job is restarted, the GUI writes a command file before starting the model that signals that a restart from the latest available restart files is required.

This means that GUI jobs should always have all NetCDF restart file generation options enabled – job setup should override any such options in the input configuration files to make sure that job pause and restart works.

Alternatively, maybe it would be good to remove the options for switching NetCDF restarts on and off and generate NetCDF restarts at the end of *every* job. This would make it possible to continue from the end of any job.

What happens to output files if a job is stopped and restarted?

Does the model just append to the existing output files or should it start new ones? What about if model parameters have been modified between stopping and restarting?

If the model is just stopped and restarted via the GUI, the GUI will write a command file to let the model know that it should restart from the latest available restart. In that case, all output files should be opened for append, i.e. they should not be recreated, since the job is just continuing from an earlier point.

Once a job has been started, the configuration is *locked*. This means that you can't just stop a job, modify its configuration and restart it. You can instead create a new job to restart from the existing job and modify *its* configuration.

Alternatively, maybe it should be possible to modify the configuration of *any* stopped job, restarting as needed. Configuration modifications would then be saved with a time range for which they are valid and graph views of model outputs would have annotations showing when configuration options were changed.

How do cluster jobs work?

Ideally, the GUI should detect whether or not it's running on the head node of a cluster. If it is, it ought to be possible to run jobs via the cluster queuing system (so maybe there needs to be some sort of GUI configuration options in `~/.cgenierc` to allow you to specify the queue submission command?).

The way that this *should* look is that a "cluster job" appears to the user to be identical to a "non-cluster job", except that they set a flag in the GUI saying that the job should be run through the queuing system. From an implementation perspective, the communication between the GUI and the model should work just as for a job run on a single node machine (assuming that the job directory is in the same location on the cluster compute and head nodes, as is usually the case, so that the GUI can write command files to the same location independent of how the job is actually running). One potential problem is managing buffering of model output – the GUI will need to pick up model standard output and error streams from the output files generated by the queuing system, and those streams need to be set up to be unbuffered so that model output can be displayed in the GUI as it appears. (This is actually an issue for non-cluster jobs as well, so maybe it just needs to be dealt with in a sensible way everywhere. At the moment, for `gfortran`, I set the `GFORTRAN_UNBUFFERED_PRECONNECTED` environment variable, which ensures that standard output and standard error are unbuffered, but maybe *all* output streams should be set to unbuffered, using the `GFORTRAN_UNBUFFERED_ALL` environment variable, because this may be needed to ensure that time series output is flushed after each timestep to allow for real-time plotting of values. I need to check though, because at least in BIOGEM, those time series files are reopened, appended to and closed each timestep, so buffering ought not to be an issue. Needs investigation...)

2.3 Job setup

The notes in this and the following sections should be read in conjunction with Figure~1, which shows the life cycle of a job as managed by the GENIE GUI. GUI buttons referred to here can be seen in Figure~2.

- New jobs are created by clicking on the "New job" button, and are initially created in an **UNCONFIGURED** state. Unconfigured jobs can be manipulated (moved, renamed, cloned, deleted, etc.) but cannot be run until the necessary configuration information is completed.
- For all job states except **RUNNING** and **ERRORED**, job configuration information can be edited under the "Configuration" tab in the GUI view. For **UNCONFIGURED** jobs, the base and user configuration files can be selected. For all editable job states, configuration "modifications" can be edited, as can the run length and the "T100" flag for the job.
- The full job configuration (i.e. the values that are used to generate the namelist files that GENIE reads at startup) is constructed from the base and user configuration files (as in the current system) plus superimposed "modifications" that allow for small changes to existing job configurations without the need to generate new base or user configuration files. The idea here is to make it easy to set up and run variants of existing jobs.
- Job configuration "modifications" allow for changing both namelist parameters and forcing parameters. (Not 100% sure how to do this cleanly yet, but it's clearly needed.)
- Once an **UNCONFIGURED** job has had its configuration information set up, it becomes a **RUNNABLE** job.

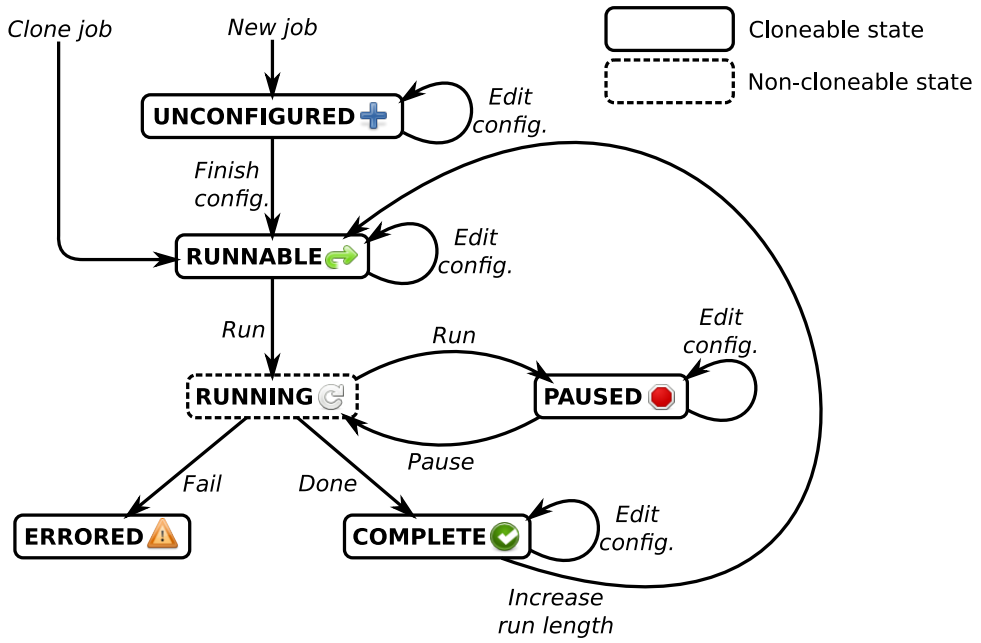

Figure 1: Life cycle of a job. The icons for each job state are those used in the GUI – see Figure~2.

2.4 Job control

- **RUNNABLE** jobs can be started by pressing the "Run job" button in the GUI. **RUNNING** jobs can be paused and restarted at will.
- Unless they are paused, **RUNNING** jobs run either to failure (in which case they become **ERRORED** jobs) or completion (**COMPLETE** jobs).
- It is possible to change the configuration of a job mid-run, either by pausing the job and editing its configuration information, or by editing the configuration information of a **COMPLETE** job and increasing its run length. In this case, restarting the job creates a new "run segment".
- At the end of each run segment (i.e. when a job is paused or it completes), NetCDF restart files are written for each GENIE component so that a new run segment can be started from the end of the previous one.
- Each run segment has its own output log, which can be viewed in the "Output" tab of the GUI, and plots of model output data can span multiple run segments, with changes in model configuration marked on the plots.
- Model time series output for adjacent run segments where the model configuration has not changed are concatenated – this makes it easy to simply run a job for a longer period than anticipated at its original setup if required. Adjacent run segments with different model configurations have time series data stored in seperate files to avoid confusion, although GUI plots concatenate the data with indicators to show where model configuration information was changed. (This approach is intended to make it easy to monitor jobs where instantaneous changes in forcing parameters are applied.)

2.5 Job monitoring

- Model log output will be captured and displayed in a scrolling window under the "Output" tab in the GUI. Model output buffering will be set up so that this output appears as the model runs to allow for real-time monitoring of the model state.

- For jobs with multiple "run segments" (i.e. jobs that have been paused and restarted, or restarted by extending their run length after they have completed), logs from previous run segments will also be saved and viewable.
- Real-time line plots of time series variables will be available under the "Plots" tabs (of which there can be more than one).
- Real-time plots will be zoomable and will show any number of selected variables on common axes.
- Plots for jobs with multiple run segments will indicate points in time where model configuration parameters changed.

3 GUI layout

Job tree Multi-rooted tree, one root per top-level job directory in use (with options to add and remove additional job directories to view); hierarchical job sub-directories for organisation; leaf job directories marked with status icon (new and unconfigured, configured, running, complete, errored).

Job actions Buttons and/or right-click menu in main job tree view:

- Clone (available for all configured jobs);
- Move, delete, rename (all jobs);
- Archive (stopped/finished jobs);
- Run (configured/stopped jobs);
- Pause (running jobs);
- Submit to queue? (configured/stopped jobs).

Configuration view Set and view base and user configuration files, run length, "T100" flag and restart information; view and edit "modifications" for both namelist variables and model forcing parameters; view resulting namelists and forcing data.

Progress view View GENIE output in real-time (and output logs for completed model runs and "run segments"); real-time graphing of selected model output (add/remove graph tabs, add/remove variables to plot in each tab); save selected graphs as "preset" for later use.

4 Miscellaneous points

- **All** GENIE jobs should write NetCDF restart files when they complete!
- Need to serialise state of job monitoring UI for use as "presets" in other jobs.
- Need reliable way to extract job status from jobs not managed by GUI (configured, running, finished).
- Need a way to pause and restart jobs – pause needs to write a restart file to allow for restarting current job and generating new jobs from existing ones.

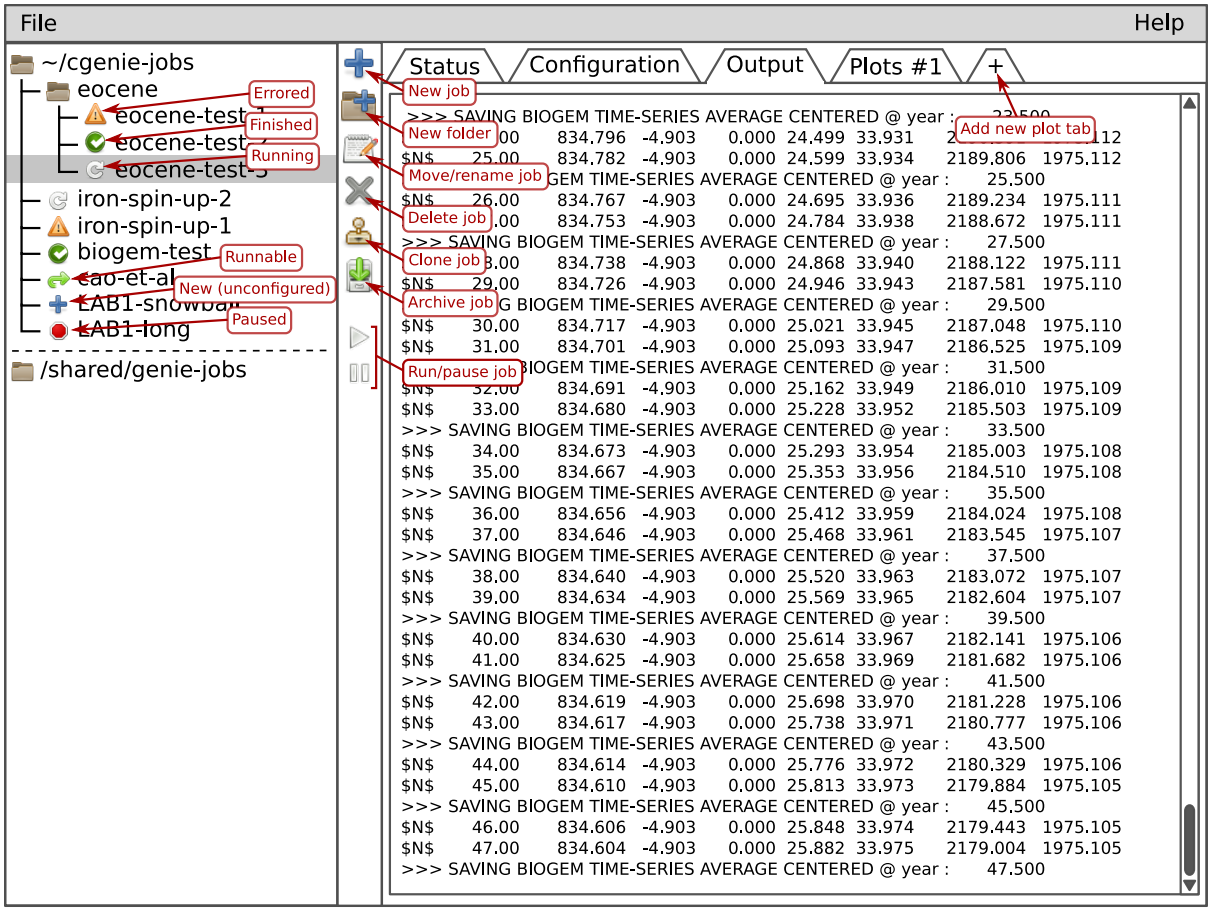

Figure 2: GUI layout: model output panel.

5 GENIE model changes

5.1 NetCDF restart generation

Namelist variables controlling NetCDF restart generation are:

```
atchem par_rstdir_name="restart/atchem"
atchem ctrl_ncrst=.TRUE.
atchem par_ncrst_name="_restart.nc"
ents rstdir_name="input/ents"
goldstein rstdir_name="restart/goldstein"
goldstein lout="rst"
goldstein rst_reset_T=.FALSE.
sedgem par_rstdir_name="restart/sedgem"
sedgem ctrl_ncrst=.TRUE.
sedgem par_ncrst_name="_restart.nc"
embm lout="rst"
embm rstdir_name="restart/embm"
biogem ctrl_ocn_rst_reset_T=.FALSE.
biogem par_rstdir_name="restart/biogem"
biogem ctrl_ncrst=.TRUE.
biogem par_ncrst_name="_restart.nc"
goldsteinseaice rstdir_name="restart/goldsteinseaice"
goldsteinseaice lout="rst"
rokgem par_rstdir_name="restart/rokgem"
```

6 GUI coding