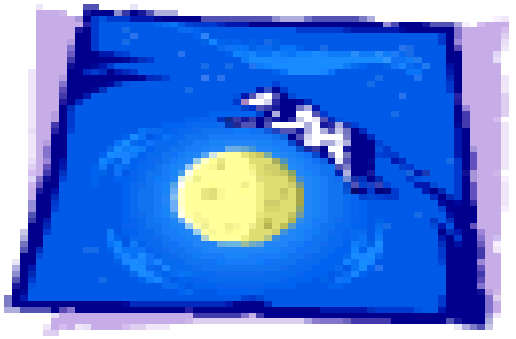
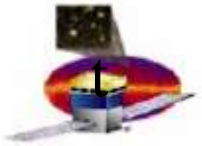
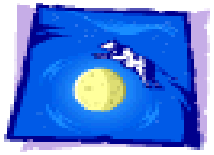


Configuration Tracking

Goals

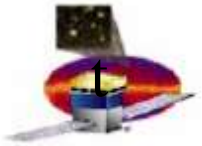
- Provide robust, reproducible method to configure the LAT for any supported mode of operation and a means to retrieve the description of any uploaded configuration, current or historical.
- Avoid creating and uploading redundant information
- Quickly implement something which satisfies above for upcoming calibration runs and smoothly evolves to handle test data runs and, ultimately, flight.



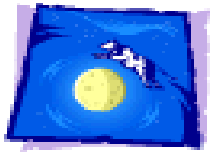


Conceptual Stages, MOOT Perspective

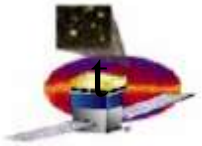
- Define and create a configuration. End product is collection of binaries.
- Determine destination, upload.
- Select among uploaded configurations; run.
- Record configuration from data stream.
- Analyze data, recovering configuration.



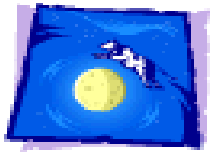
Config Contents



- Input to LATC
- Input to LCI for calibration runs
- Calorimeter pedestals and gains; input to Filter, etc., for data runs
- For now, at least, MOOT will not keep track of thermal parameters, power-on state, etc. These can be added later if needed.
- MOOT will not keep track of code modules.



MOOT Implementation



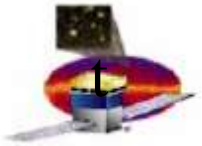
Database + code to provide the following services:

- Create a configuration
- Prepare for upload
- Record configuration extracted from data stream
- Support standard queries, such as
  - Recover config in effect at a particular time – not just files, but also intent for some kinds of information.
  - Find out when a particular file was in use
- Recover config values for use in simulation or recon

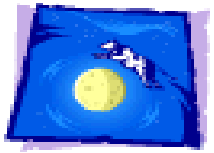
Status

- Database design is essentially done
- **Create config** code is written and checked out for initial simple case (all that is needed for calibration runs), except for calls to FSW.
- Remaining services are not yet implemented, but those needed soon (prepare for upload; record active config; queries) are straightforward compared to **create config**.

- Integration with FSW, Online looming.
- Recovering config – see next slide.



Recovering Config: Splits

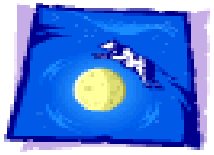


Recovering config information for use by Sim/Recon is a longer term project, but is supported by design.

Conside, e.g., tracker splits. They are uploaded to the instrument as the SPT component of LATC. Recon also needs them, but expects a different format.

The complete cycle for splits (creation, upload, use by Recon) could go something like this:

Splits cont'd

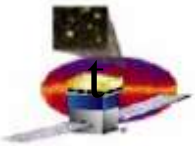


Person or proc creates

Splits recon style

*Register in Parameters table
*Look up corresponding FSW class & “from_precursor” procedure
*Run it to make SPT
*Run “to_binary” procedure
* **fmx add**; store logical id
* **fmx upload**; store file id

Splits uploadable



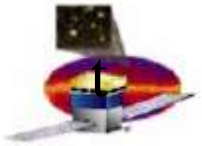
Data

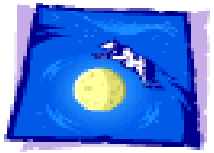
Extract log.ids

Log in History table
If changed splits id,
- trace to recon-style file
- make or update entry in calib. metadata dbs

Gleam calibration infrastructure can find correct file

Mostly MOOT
Other active agent



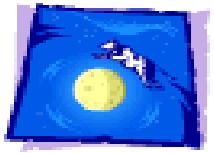


Appendix

Remaining slides describe some details of database design and flow for configuration processing.

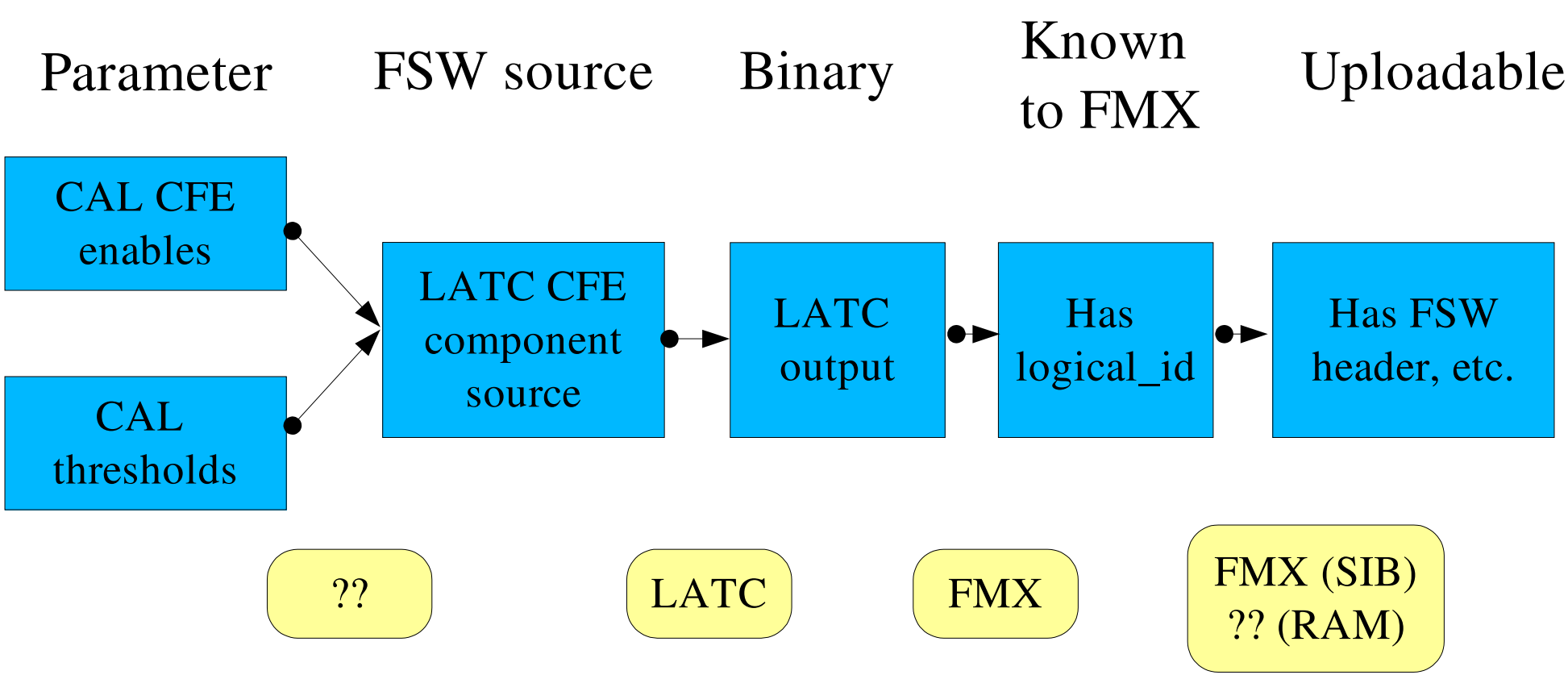
See also
http://www.slac.stanford.edu/exp/glast/ground/notes/moot/
an index of working documents concerning MOOT.

Database design



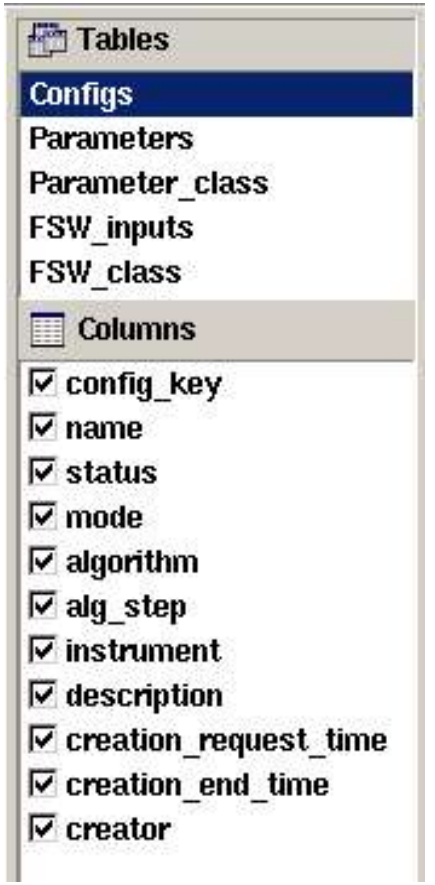
- Primary tables to represent **Parameters** (e.g. CFE thresholds), **FSW inputs** (e.g., complete CFE settings), and **Configs**; probably also **History** (or may be maintained elsewhere).
- Secondary tables to describe classes of parameter files and classes of FSW inputs.
- Secondary tables to relate Parameter file instances to FSW input instances; FSW input instances to Configs.

Config File Pipeline (MOOT view)

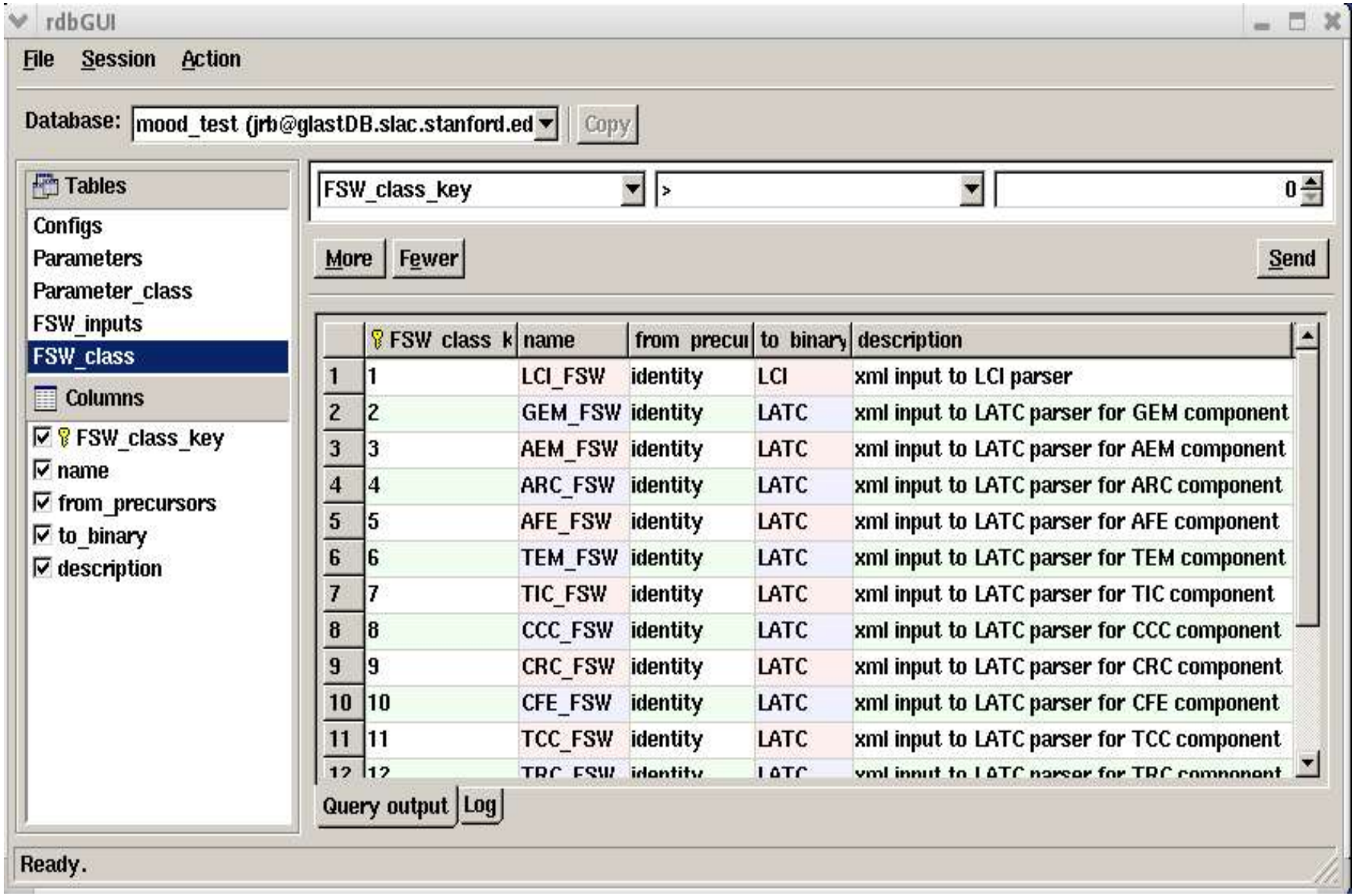


Only first stage may be many-to-one; all others are one-to-one.
Imagine parallel pipeline for each LATC component

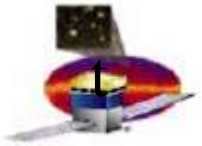
Primary Tables



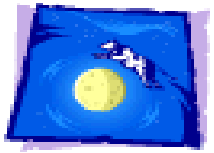
FSW File Classes



	FSW_class_k	name	from_precu	to_binary	description
1	1	LCI_FSW	identity	LCI	xml input to LCI parser
2	2	GEM_FSW	identity	LATC	xml input to LATC parser for GEM component
3	3	AEM_FSW	identity	LATC	xml input to LATC parser for AEM component
4	4	ARC_FSW	identity	LATC	xml input to LATC parser for ARC component
5	5	AFE_FSW	identity	LATC	xml input to LATC parser for AFE component
6	6	TEM_FSW	identity	LATC	xml input to LATC parser for TEM component
7	7	TIC_FSW	identity	LATC	xml input to LATC parser for TIC component
8	8	CCC_FSW	identity	LATC	xml input to LATC parser for CCC component
9	9	CRC_FSW	identity	LATC	xml input to LATC parser for CRC component
10	10	CFE_FSW	identity	LATC	xml input to LATC parser for CFE component
11	11	TCC_FSW	identity	LATC	xml input to LATC parser for TCC component
12	12	TRC_FSW	identity	LATC	xml input to LATC parser for TRC component



Recovering Config



- Assume input is a timestamp. Discover logical_ids of files in use from **History** table.
- Search **FSW_inputs** table to find (MOOD) keys for these files.
- Search **FSW_to_Parameters** to find keys for Parameter files
- Look up in **Parameters** table; discover characteristics of these files.