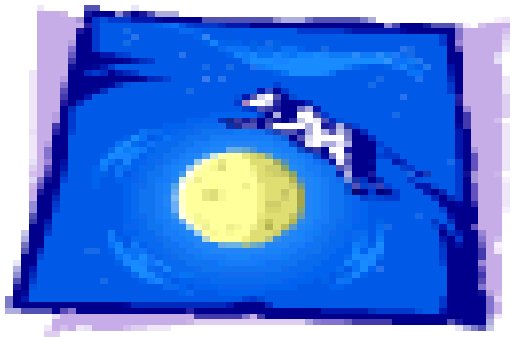

It's MOOT

Modes of Operation Tracking

- Goals and approach
- Database design
- Use cases
- What next?

Goals and Approach

MOOT's primary tasks for October 1st are to track configurations of the LAT– consisting of register settings, input to parts of FSW code such as CAL pedestals for the Event Filter, charge calibration parameters, etc. – and to provide some help in creating and uploading them.

At the lowest level, a configuration consists of a large number of atomic quantities, such as register settings, but this representation should only be resorted to when absolutely necessary.

Flight software concerned with creating and uploading configuration files tends to group the atomic quantities one way; people using the LAT for some particular purpose, another.

Translation

Part of MOOT's job is to translate between the two views.

This xml file represents both.

```
- <config>
  - <notes>
      Information on the whiteboard after morning meeting of Aug. 8. File contains a
      list of <delegates> (in the MOOT sense; see e.g.
      http://www.slac.stanford.edu/exp/glast/ground/notes/moot/moot21july05.shtml)
      followed by a list of <FSWinputs>.
    </notes>
  + <delegates></delegates>
  + <FSWinputs></FSWinputs>
  </config>
```

FSW inputs

```
- <FSWinputs>
  - <notes>
      An <FSWinput> represents one of the kinds of files to be input to a FSW
      service. These should almost map 1-1 to files output by FSW. LATC outputs
      15 different kinds of files, by last count. LCI outputs one kind, and CDM will
      ultimately transform all other kinds of input (housekeeping, event filter
      parameters,...)
    </notes>
  - <FSWinput name="LHKout" service="CDM">
      <notes>Single output for LAT Housekeeping parameters</notes>
    </FSWinput>
    <FSWinput name="GEMout" service="LATC"> </FSWinput>
    <FSWinput name="AEMout" service="LATC"> </FSWinput>
  - <FSWinput name="ARCout" service="LATC">
      <notes>For ACD readout controller registers</notes>
    </FSWinput>
```

Delegates

```
- <delegate name="Timing">
  - <notes>
      Output for this one may be scattered among different LATC output files.
    </notes>
  - <parmGroup name="TKR timing">
      <parm name="OR stretch"/>
    </parmGroup>
    <parm name="CAL timing"/>
    <parm name="GEM timing"/>
  - <parm name="ACD timing">
    - <notes>
        Is this contained within a single LATC output, or is it split, e.g., between AEMout and
        ARCout?
      </notes>
    </parm>
</delegate>
```

Each parm should correspond to a source file (needs work)

Database design

MOOD will have primary tables with a row for each instance of the fundamental objects: configs, FSW input files and parameters. One more primary table is needed for history. It will gain a row each time a new config upload is requested. These tables are known as Configs, FSW_input, Parameters and History.

Secondary tables Config_FSW and FSW_Parm are needed to keep track of relations between the objects in the primary tables and to store information about types of, e.g. FSW inputs (FSW_class).

History table

History fields

Field name	Explanation, typical contents
history_key	Primary key, auto_increment; unique identifier for the row
config_fk	Foreign key, referencing an entry in **Configs**.
request_time	Timestamp for request to load
ack_time	Timestamp when notification was received of success or failure of load
active_start	Same as **ack_time** for successfully loaded configurations; else null.
active_end	Timestamp when config was overwritten by the next one
load_status	One of 'in_progress', 'loaded', 'superseded', 'failed',...

Configs table

Configs fields

Field name	Explanation, typical contents
config_key	Primary key, auto_increment; unique identifier for the row
mode	Major mode, one of an enumerated set (data, calibration,..)
name	Convenient short name for interactive use
notes	Longer description of what the configuration is supposed to do.
who	Who asked to create it
request_time	Time of start of process to create config
creation_time	Time of creation (or of end of attempt)
status	At what stage of processing is this config? E.g., 'OK', 'abort', 'in_progress', ...
instrument	Which LAT-like instrument does it refer to?

FSW_input table

FSW_input fields

Field name	Explanation, typical contents
FSW_input_key	Primary key, auto_increment; unique identifier for the row
source	reference which may be used to find the file; e.g. file path or path and version in CVS archive
class_fk	Denotes type of file; e.g., CAL gains or GEM input to LATC. Note this value is *not* unique within the table, but it refers to a primary key in the **FSW_class** secondary table.
status	Keeps track of processing stages. May make row before file has actually been created. Once created, it probably gets committed to CMX. Finally, it is processed by FMX and put into the **Logical** table.
FSW_id	If the input file is transformed to binary successfully and inserted into the FMX **Logical** table, it will it will have an id assigned to it.

Secondary tables

Configs_FSW fields

Field name	Explanation, typical contents
config_fk	Reference to element in **Configs**
FSW_fk	Reference to element in **FSW_input**

FSW_class fields

Field name	Explanation, typical contents
class_key	Primary key, auto_increment; unique identifier for the row
process	reference to program or script which turns parameter input into the source for FSW. Goal is to automate the transformation from parameter files to FSW input file. May require additional fields in this table.

Use cases

- Activate pre-existing config – see below
- Find config constituents – easy
- Find active config by time – easy
- Add parm (precursor) file – rdbGUI-like
- Create config – for Oct. 1 an entirely offline process. MOOT might provide some help

Assume....

- All (precursor) parameter files exist and have been registered in the Parameters table
- All FSW inputs have been constructed and entered in FSW_input.
- FSW inputs have been transformed to binaries, entered in FMX Logical table.
- Entry in Configs exists; secondary tables are also up to date.

Activate (approximate)

- MOOT queries FMX – binaries uploaded?
- If not,
  - Report missing binaries to LICOS
  - LICOS reformats into packets, uploads
  - Informs MOOT, FMX when complete
- LICOS selects files via telecommand
- When complete, LICOS invokes MOOT service to update History, relay to FMX

What next?

- Complete and correct parameters description
- Pin down handling of LATC master config.
- Find a home for parameter source files (precursors to FSW_input files)
- Close the loop: analysis outputs →precursors
- Think through inter-db issues
- Write the thing!

More details

- Go to MOOT index: http://www.slac.stanford.edu/exp/glast/ground/notes/moot/
- Follow link "Report for Aug. 11" for most material in this talk
- Follow link "Delegates" to see the xml file excerpted in early slides.