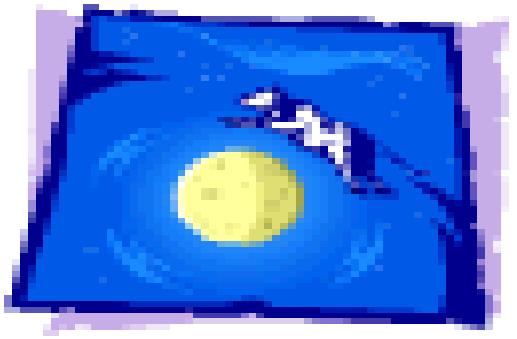
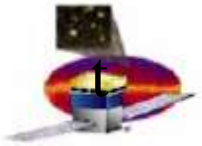
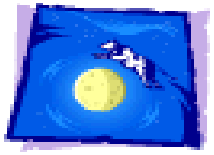


Building Configurations for Initial Charge-injection Calibration Runs

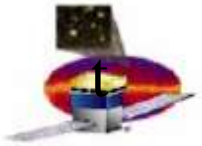
J. Bogart



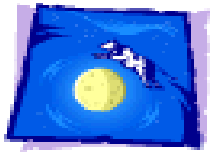


Conceptual Stages Involving MOOT

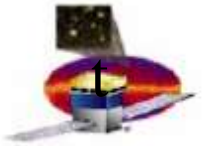
- Define and create a configuration.
  - Provide service, e.g. *configId*=makeConfig(*file-list*)
  - *file-list* is collection of source files for LATC, LCI, etc.
  - Caller may retrieve output binaries from *configId* for use in next stage.



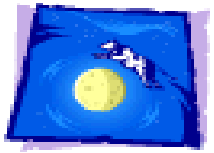
Stages cont'd



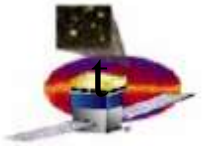
- Determine destination, prepare for upload.
  - *status* = prepare(*configId*, *dest*)
  - For each binary belonging to config, ask fmx to prepare for upload. Make LATC master file if necessary. Result is set of uploadable binaries, known to fmx.
  - Procedure is straightforward: invoke other services; do some bookkeeping.



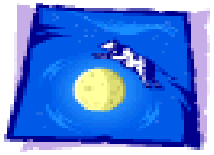
Stages cont'd



- Upload
  - retrieveForLoad(*configId*, &*fileIdList*, &*needLoadList*)
  - fileIds will be readily retrievable from configId
  - For sib at least, will be able to determine for each file whether it is already uploaded and avoid re-upload.
- Select among uploaded configurations; run
  - Ability to query Configs (to, e.g., display menu)
  - retrieveForRun(*configId*, &*fileIdList*)
  - Does not return same list as previous service; doesn't necessarily include all arguments to start-run



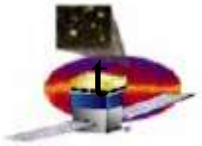
Status

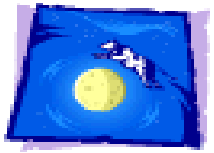


- The most complex part of the logic in MOOT is in the first step; that part of it is done. Remaining work there includes
  - Handling of other source types; no surprises expected.
  - Invoking fmx to do the rest. Interface is defined and will do the job.
- Remaining steps are not much more than specific database queries and additional interactions with fmx.
- Total amount of work is noticeable (weeks?) but not huge. However much of it can't begin without the new fmx.

Inputs needed, issues, etc.

- Fmx. It's on its way.
- What happened to the “ignore for readback” facility?
- Need to revisit single versus multiple databases for MOOT. May want to have a travelling instance for LAT and a fixed SLAC one for testbed. They would be essentially disjoint.

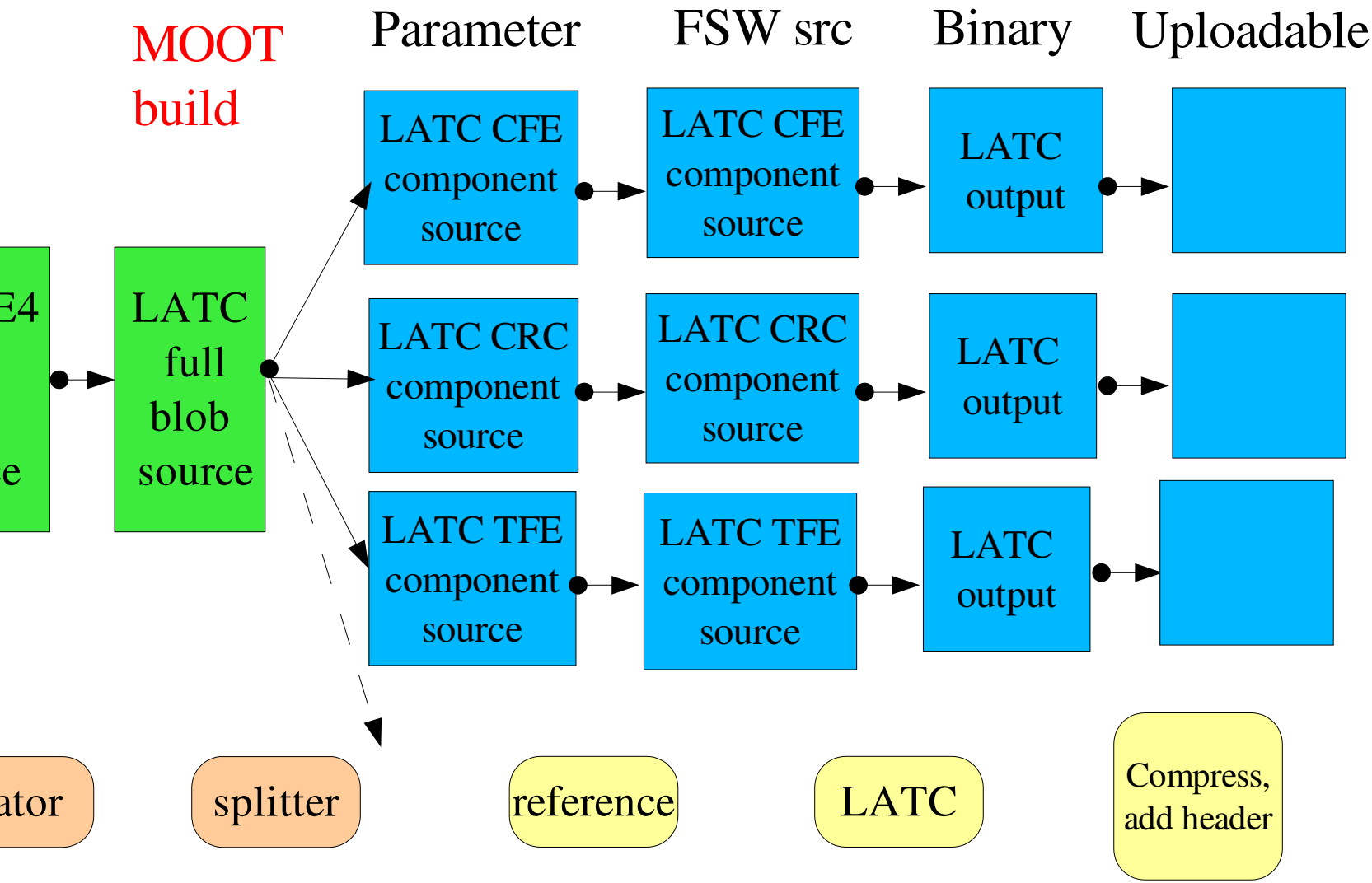




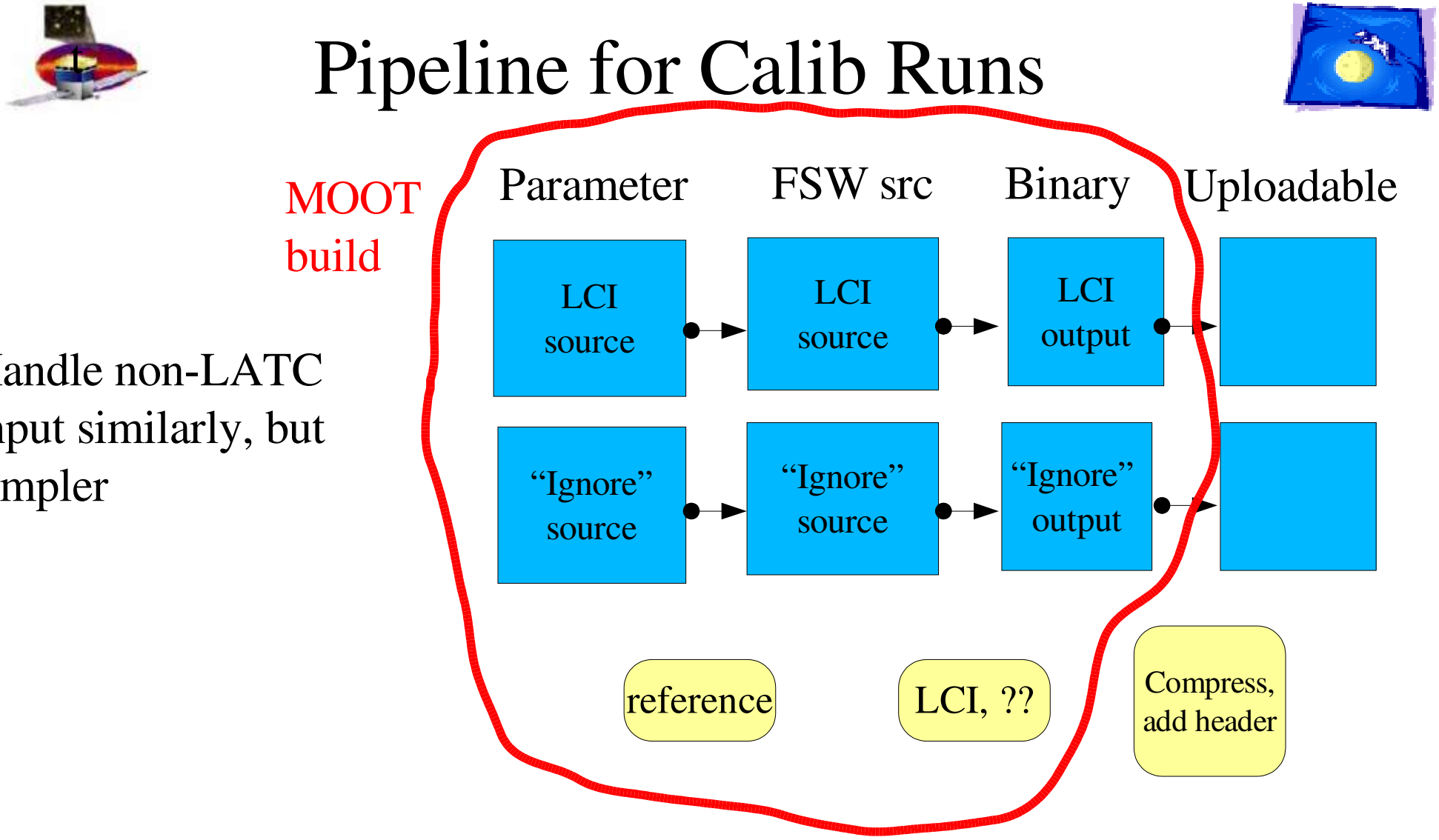
Old stuff

Remaining slides were part of an earlier presentation and are just here for reference.

Pipeline for Calib Runs



Pipeline for Calib Runs



Handle non-LATC input similarly, but simpler

Status

Excerpt from test program:

```
MOOT::MoodConnection* moodCon =
   new MOOT::MoodConnection(host, user, pw, dbname);

 MOOT::MootBuild moot(moodCon);

 std::vector<ParmDescrip> pd;
 pd.reserve(3);

 pd.push_back(ParmDescrip("$(MOOTROOT)/data/daq_aem.xml", "AEM_parm"));
 pd.push_back(ParmDescrip("$(MOOTROOT)/data/daq_gem.xml", "GEM_parm"));
 pd.push_back(ParmDescrip("$(MOOTROOT)/data/daq_dft.xml", "DFT_parm"));

 unsigned configKey = moot.simpleConfig(pd, "AnotherFirstConfig",
                                        MOOT::MootBuild::MODE_data,
                                        "emptyAlg");
```

..results in creation of many entries in dbs, including a row in the Configs table, and three binary files