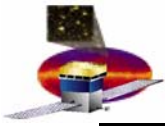**Gamma-ray Large
Area Space
Telescope**

GLAST Large Area Telescope
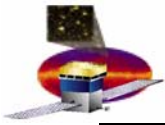
SVAC Pipeline

**Warren Focke
SLAC
I&T Science Verification Analysis and Calibration
Engineering Physicist
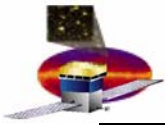focke@slac.stanford.edu
650-926-4713**

Outline

- **Pipeline products**

- **Issues affecting implementation**

- **Implementation**

- **Wish list**

Pipeline Products

- **5 Files**
  - **LDF**
    - **From Online**
  - **Digi**
  - **Merit**
  - **Recon**
  - **SVAC**
- **3 Reports**
  - **Configuration**
    - **Depends on schema, snapshot (from Online)**
  - **Digi**
    - **Depends on digi file**
  - **Recon**
    - **Depends on recon, digi files**
- **Put it all on the web**
  - **Enter in eLog DB**

Reprocessing

- **A review is required to decide when to do it**
  - **CCB chaired by Richard Dubois**
- **Reasons for reprocessing**
  - **Major software update (EM package, Calibration algorithms)**
  - **New calibration constants**
  - **Tower swapping**
- **Procedure to do the reprocessing**
  - **Create a new version of task in the pipeline**
  - **Run the new task**
  - **GINO does not understand multiple versions of the "same" data product (we'll revisit this later)**
- **May not need to redo everything**

Logical Dependency Flow
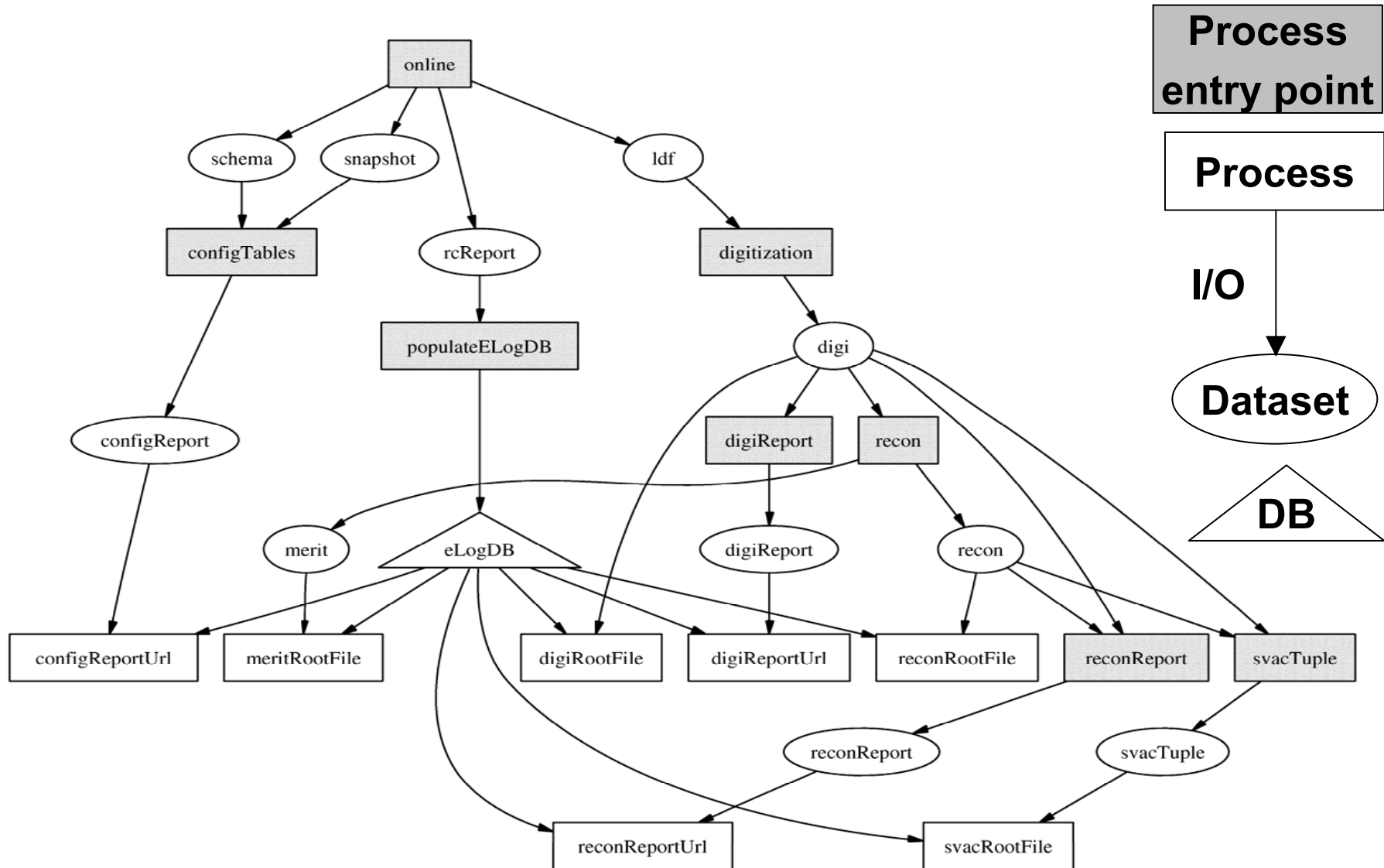
Make It So – Proof of Concept

```
BASE = /nfs/slac/g/svac/focke/make

FINISH = $(BASE)/finished/$(RUN)

#### (((( Online ((((
RAWDIR = $(BASE)/rawFiles/$(RUN)

RCREPORT = $(RAWDIR)/online-EM2-v1r0_$(RUN)_rcReport_rcReport.xml
SCHEMA = $(RAWDIR)/online-EM2-v1r0_$(RUN)_schema_text.xml
SNAPSHOT = $(RAWDIR)/online-EM2-v1r0_$(RUN)_snapshot_text.xml
LDF = $(RAWDIR)/online-EM2-v1r0_$(RUN)_ldf_LDF.fits
#### )))) Online ))))


#### (((( SVAC ((((
ROOTDIR = $(BASE)/rootFiles/$(RUN)

DBUPDATE = $(ROOTDIR)/dbUpdate

CONFIGDIR = $(ROOTDIR)/configReport/$(configReportVersion)
CONFIGREPORT = $(CONFIGDIR)/configReport-EM2-v1r0_$(RUN)_tarBall_Analysis.tgz
CONFIGREPORTWEB   $(CONFIGDIR)/web

EMDIR = $(ROOTDIR)/$(EngineeringModelVersion)

DIGIDIR = $(EMDIR)/grRoot
DIGI = $(DIGIDIR)/digitization_EM2_v1r0_$(RUN)_digi_DIGI.root
DIGIWEB = $(DIGIDIR)/web

DIGIREPDIR = $(EMDIR)/grRoot/digiReport/$(digiReportVersion)
DIGIREP = $(DIGIREPDIR)/digiReport-EM2-v1r0_$(RUN)_tarBall_Analysis.tgz
DIGIREPWEB = $(DIGIREPDIR)/web

CALIBDIR = $(EMDIR)/$(calibVersion)

RECONDIR = $(CALIBDIR)/grRoot
RECON = $(RECONDIR)/recon_EM2_v1r0_$(RUN)_recon_RECON.root
RECONWEB = $(RECONDIR)/reconWeb
MERIT = $(RECONDIR)/recon_EM2_v1r0_$(RUN)_merit_merit.root
MERITWEB = $(RECONDIR)/meritWeb
RECONTARGETS = $(RECONWEB) $(MERITWEB)

RECONREPDIR = $(CALIBDIR)/reconReport/$(reconReportVersion)
RECONREP = $(RECONREPDIR)/reconReport-EM2-v1r0_$(RUN)_tarBall_Analysis.tgz
RECONREPWEB = $(RECONREPDIR)/web

SVACDIR = $(CALIBDIR)/svacRoot/$(svacTupleVersion)
SVAC = $(SVACDIR)/svacTuple-EM2-v1r0_$(RUN)_svac_svac.root
SVACWEB = $(SVACDIR)/web
#### )))) SVAC ))))

$(FINISH): $(SVACWEB) $(CONFIGREPORTWEB) $(DIGIWEB) $(DIGIREPWEB) $(RECONTARGETS)
        echo '--->' Done with run $(RUN)!
        date >> $@


$(DBUPDATE): $(RCREPORT)
        mkdirhier `dirname $@`
        echo '===>' Update DB for run $(RUN).
        date >> $@


$(CONFIGREPORTWEB): $(DBUPDATE) $(CONFIGREPORT)
        echo '===>' Put config report for run $(RUN) on the web.
        date >> $@
```

```
$(CONFIGREPORT): $(SCHEMA) $(SNAPSHOT)
        mkdirhier `dirname $@`
        echo '===>' Make config report for run $(RUN).
        date >> $@


$(DIGIWEB): $(DBUPDATE) $(DIGI)
        echo '===>' Put digi file for run $(RUN) on the web.
        date >> $@

$(DIGI): $(LDF)
        mkdirhier `dirname $@`
        echo '   >' Digitize run $(RUN).
        date >> $@


$(DIGIREPWEB): $(DBUPDATE) $(DIGIREP)
        echo '--->' Put digi report for run $(RUN) on web.
        date >> $@

$(DIGIREP): $(DIGI)
        mkdirhier `dirname $@`
        echo '--->' Make digi report for run $(RUN).
        date >> $@


$(RECONWEB): $(DBUPDATE) $(RECON)
        echo '--->' Put recon file for run $(RUN) on web.
        date >> $@

$(MERITWEB): $(DBUPDATE) $(MERIT)
        echo '--->' Put merit file for run $(RUN) on web.
        date >> $@

$(RECON) $(MERIT): $(DIGI)
        mkdirhier `dirname $@`
        echo '--->' Reconstruct run $(RUN).
        date >> $(RECON)
        date >> $(MERIT)


$(RECONREPWEB): $(DBUPDATE) $(RECONREP)
        echo '===>' Put recon report for run $(RUN) on web.
        date >> $@

$(RECONREP): $(RECON) $(DIGI)
        mkdirhier `dirname $@`
        echo '--->' Make recon report for run $(RUN).
        date >> $@


$(SVACWEB): $(DBUPDATE) $(SVAC)
        echo '--->' Put SVAC tuple for run $(RUN) on web.
        date >> $@

$(SVAC): $(RECON) $(DIGI)
        mkdirhier `dirname $@`
        echo '--->' Make SVAC tuple for run $(RUN).
        date >> $@


clean:
        rm -rf $(FINISH) $(ROOTDIR)
```

- **Makefile that implements dependency flow from previous graph.**
- **Reproduces SVAC pipeline directory structure.**
- **Doesn't actually do any work.**

Factors Influencing implementation

SVAC Needs

- **Robust in presence of failure**
    - **logical parallelism reduces dependencies**
    - **want it to work like "make -k"**

- **Multiple entry points**
    - **Reprocessing**
    - **Saves disk & CPU**

GINO Features

- **Processes in a task are sequential**
    - **Each depends on the last**
    - **Must start with the first**

Current Implementation "Features"

- **Dependencies ensured "by hand"**
- **Having multiple tasks gives us**
  - **Some degree of parallelism**
  - **Multiple entry points**
- **Almost twice as many processes**
  - **Extra processes required to launch other tasks**
- **Duplicate data sets**
  - **Necessary due to naming conventions**
  - **Actually symbolic links (may have implications for archiving data)**
- **Artificial dependencies**
  - **Introduced by sequential ordering**
- **Some actions must be taken without GINO's knowledge**
  - **Writing links**
  - **Launching tasks**
- **Calibration is done manually**
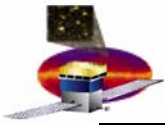 - **Until we understand instrument better**
  - **Not GINO's fault**

Current Implementation
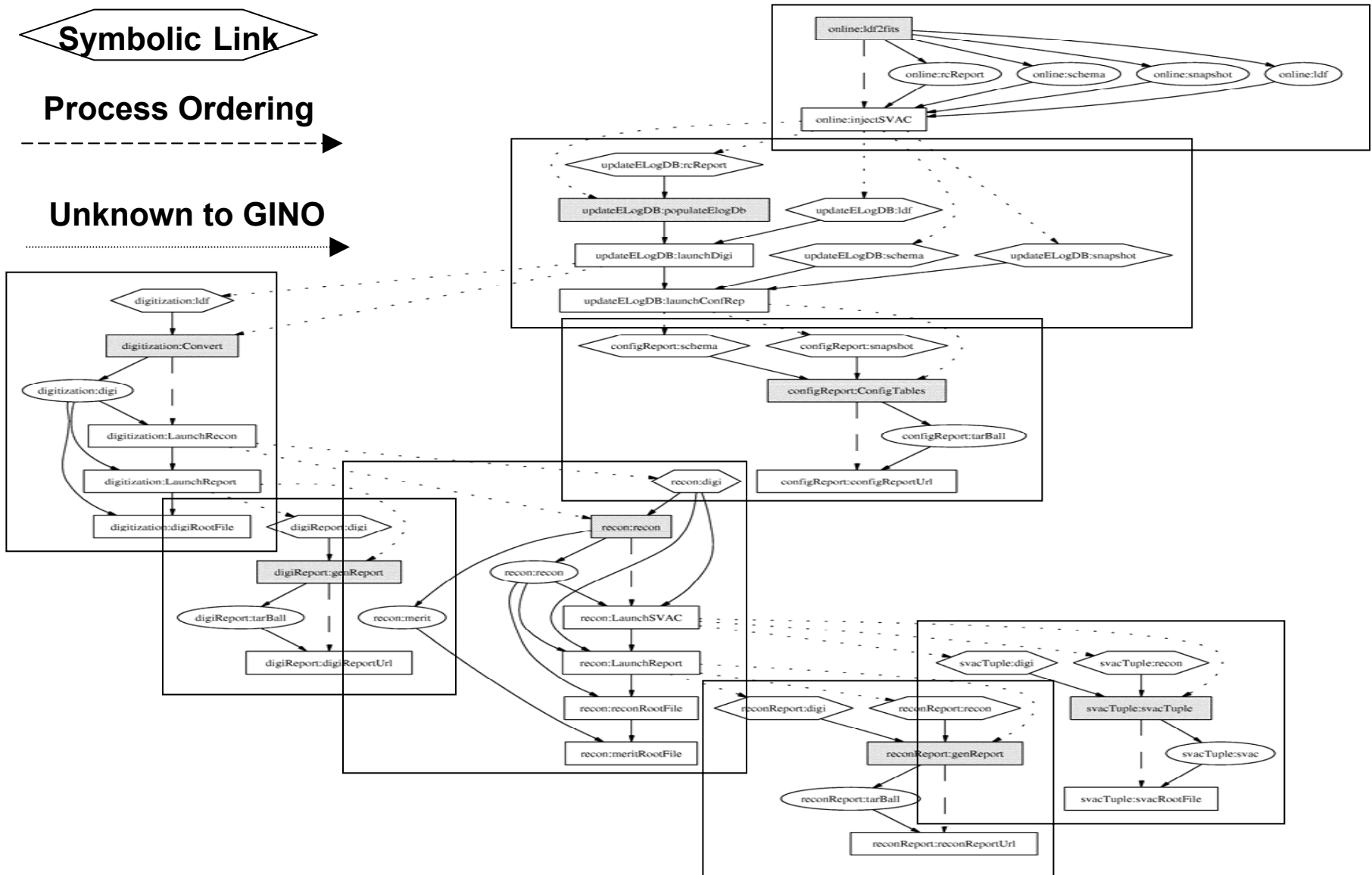
Wish List (at the moment)
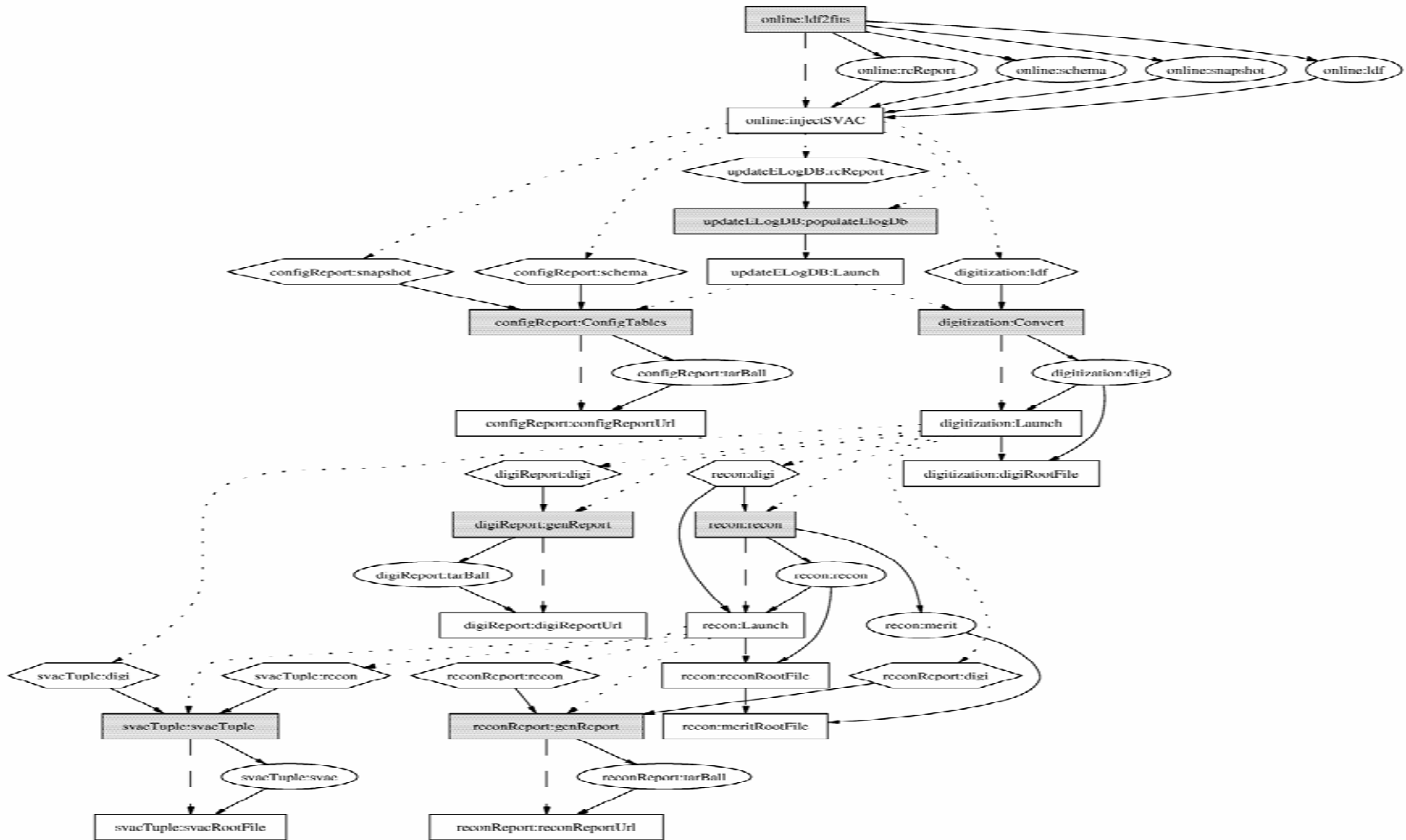
- **Short Term**
  - **Back end**
    - **Work reliably**
    - **Archive/Back up all products (including Online's)**
    - **OldFail status**
      - **New status, manually set**
    - **Wait timeout**
      - **Could start tasks in parallel, they'll run when their dependencies are complete**
  - **Front end**
    - **Filter on status (per process), run #, date**
    - **Sort on status, run #, date**
- **Next Version**
  - **Work like "make –k"**
  - **Task version #**
    - **Would simplify reprocessing**
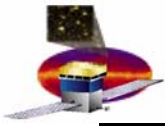
End

- **Following slides are backup**

New Launching Scheme

Next Version