SoLID Software Framework

Ole Hansen

Jefferson Lab

SoLID Collaboration Meeting
May 5, 2016

Framework Pros & Cons

Framework	Pros	Cons
art (FNAL)	• Large user base • Developed by experts • Very good documentation • Modern • ROOT6 support • Best match to our requirements	• Not multi-threaded, not distributed (but multi-threading planned) • Heavy binary installation by default • In-house build system • Somewhat complex
FairROOT (GSI)	• Familiar ROOT environment • Large user base (incl. EIC a.t.m.) • Distributed processing extension (experimental) • Good built-in simulation support	• Absent documentation • Poor API definition • Old code base • Existing code tends to be a mess • Single-threaded (unlikely to change) • Heavy dependency requirements
Fun4All (PHENIX)	• Lightweight • Well-tested, proven performance • Familiar ROOT environment	• One-man project • Very PHENIX-centric • Absent documentation • Very old code base • Many missing standard features • Single-threaded (unlikely to change)
JANA (JLab Hall D)	• Multi-threaded • Lightweight • Local expertise	• Small user base • Too many technical limitations • In-house DST format (HDDM)
Clara (JLab Hall B)	• Multi-threaded and distributed • Local expertise	• Small user base • Java based • Very complex • Performance concerns • In-house DST format (EVIO)

art Test Installation @JLab

```
ifarm1101[1] ls /work/halla/solid/FNAL/products/
art/            g4abla/       g4tendl/          python/          toyExperiment/
boost/          g4emlow/      gcc/              qt/              upd/
cetbuildtools/  g4neutron/    gccxml/           root/            ups/
cetlib/         g4neutronxs/  gdb/              setup            valgrind/
cetpkgsupport/  g4nucleonxs/  geant4/           setups           xerces_c/
clhep/          g4nuclide/    git/              setups_layout    xrootd/
cmake/          g4photon/     gitflow/          setup-solid.csh
cppunit/        g4pii/        libxml2/          setup-solid.sh
fftw/           g4radiative/  messagefacility/  sqlite/
fhiclcpp/       g4surface/    ninja/            tbb/

ifarm1101[2] source /work/halla/solid/FNAL/products/setup-solid.csh
ifarm1101[3] which ups
/work/halla/solid/FNAL/products/ups/v5_2_0/Linux64bit+2.6-2.12/bin/ups

ifarm1101[5] ups list -aK+ art
"art" "v1_18_05" "Linux64bit+2.6-2.12" "debug:e9" ""
"art" "v1_18_05" "Linux64bit+2.6-2.12" "e9:prof" ""

ifarm1101[5] du -sh /work/halla/solid/FNAL/products/
58G /work/halla/solid/FNAL/products/
```

- The above is a (mostly) self-contained software environment
- This installation includes debug builds, source code and Geant4
- The contents are portable — just copy the directory to your workstation (must be RHEL 6, CentOS 6, or compatible. RHEL 7 to follow.)
- When porting, update setup-solid.{sh,csh} scripts as appropriate

art Test Installation @JLab

```
ifarm1101[1] ls /work/halla/solid/FNAL/products/
art/            g4abla/       g4tendl/          python/          toyExperiment/
boost/          g4emlow/      gcc/              qt/              upd/
cetbuildtools/  g4neutron/    gccxml/           root/            ups/
cetlib/         g4neutronxs/  gdb/              setup            valgrind/
cetpkgsupport/  g4nucleonxs/  geant4/           setups           xerces_c/
clhep/          g4nuclide/    git/              setups_layout    xrootd/
cmake/          g4photon/     gitflow/          setup-solid.csh
cppunit/        g4pii/        libxml2/          setup-solid.sh
fftw/           g4radiative/  messagefacility/  sqlite/
fhiclcpp/       g4surface/    ninja/            tbb/

ifarm1101[2] source /work/halla/solid/FNAL/products/setup-solid.csh
ifarm1101[3] which ups
/work/halla/solid/FNAL/products/ups/v5_2_0/Linux64bit+2.6-2.12/bin/ups

ifarm1101[5] ups list -aK+ art
"art" "v1_18_05" "Linux64bit+2.6-2.12" "debug:e9" ""
"art" "v1_18_05" "Linux64bit+2.6-2.12" "e9:prof" ""

ifarm1101[5] du -sh /work/halla/solid/FNAL/products/
58G /work/halla/solid/FNAL/products/
```

- The above is a (mostly) self-contained software environment
- This installation includes debug builds, source code and Geant4
- The contents are portable — just copy the directory to your workstation (must be RHEL 6, CentOS 6, or compatible. RHEL 7 to follow.)
- When porting, update setup-solid.{sh,csh} scripts as appropriate

Getting Started: The *art* Workbook (I)

- Download workbook PDF
  `https://web.fnal.gov/project/ArtDoc/Shared%20Documents/art-documentation.pdf`
- Install the workbook code (with JLab patches)

```
ifarm1401[70] source /work/halla/solid/FNAL/products/setup-solid.csh
ifarm1401[74] mkdir workbook; cd workbook
ifarm1401[76] tar xf /work/halla/solid/FNAL/art-workbook.tar.bz2
ifarm1401[78] mkdir build-prof; cd build-prof
ifarm1401[86] source ../art-workbook/ups/setup_for_development $ART_WORKBOOK_QUAL -p
.... (snip) ....
ifarm1401[87] buildtool -j4
.... (go have a coffee) ....

------------------
INFO: Stage build successful.
------------------
ifarm1401[88]
```

Getting Started: The *art* Workbook (II)

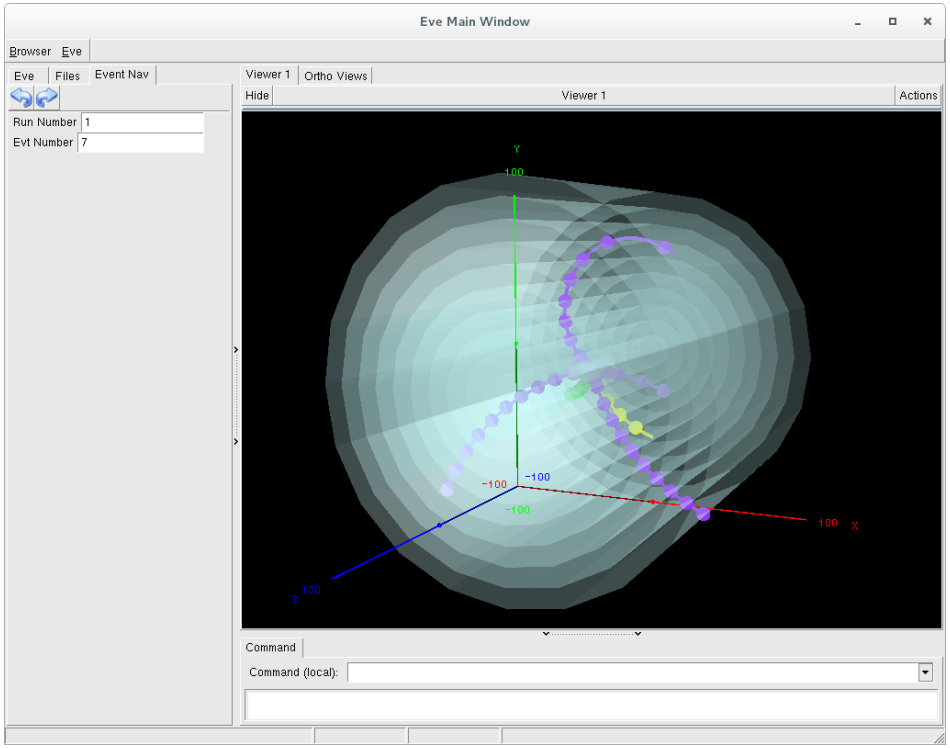
- Start using the examples

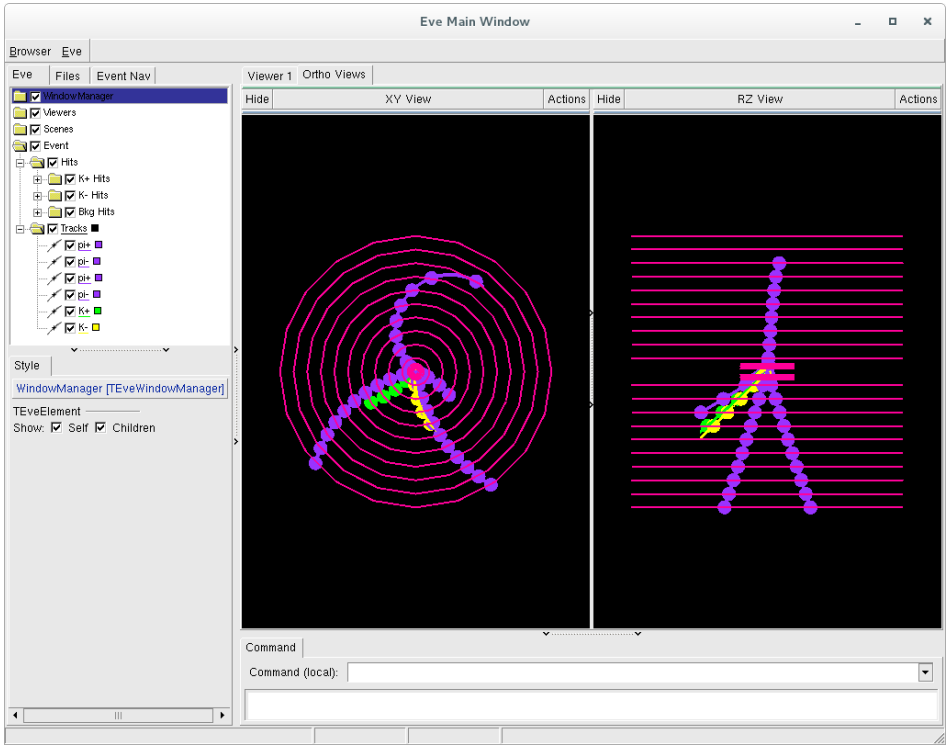
```
ifarm1401[89] art -c fcl/FirstModule/first.fcl |& tee output/first.log
%MSG-i MF_INIT_OK:  art 05-May-2016 15:03:56 EDT JobSetup
Messagelogger initialization complete.
%MSG
05-May-2016 15:04:10 EDT  Initiating request to open file inputFiles/input01.art
05-May-2016 15:04:13 EDT  Successfully opened file inputFiles/input01.art
Hello from First::constructor.
Begin processing the 1st record. run: 1 subRun: 0 event: 1 at 05-May-2016 15:04:38 EDT
Hello from First::analyze. Event id: run: 1 subRun: 0 event: 1
Hello from First::analyze. Event id: run: 1 subRun: 0 event: 2
.... (snip) ....
Hello from First::analyze. Event id: run: 1 subRun: 0 event: 10
05-May-2016 15:04:38 EDT  Closed file inputFiles/input01.art

TrigReport ----- Event  Summary ------
TrigReport Events total = 10 passed = 10 failed = 0

TrigReport --- Modules in End-Path: end_path ------
TrigReport  Trig Bit#    Visited     Passed     Failed      Error Name
TrigReport     0    0         10         10          0          0 hi

TimeReport ----- Time  Summary --[sec]--
TimeReport CPU = 0.000456 Real = 0.000000

Art has completed and will exit with status 0.
```

Getting Started: The *art* Workbook (III)

- `art-workbook` is based on `toyExperiment`
- Event display demo – uses ROOT's Eve viewer[1]

```
[ole@haplix1a build-prof]$ art -c fcl/EventDisplay3D/eventDisplay01.fcl
```





[1] Requires OpenGL. Does not work over ssh

Resources, Documentation

- The *art* website (brand new!):

  `http://art.fnal.gov`

- Wiki (lots of developer information)

  `https://cdcvs.fnal.gov/redmine/projects/art/wiki`

- August 2015 software workshop (many informative talks)

  `https://indico.fnal.gov/conferenceDisplay.py?confId=9928`

- Mu2e documentation

  `http://mu2e.fnal.gov/public/hep/computing/gettingstarted.shtml`

Draft Task List for *art*@SoLID

1. Implement geometry service
2. Implement producer module(s) that call Geant4, similar to `artG4`
3. Define a draft of a data model for SoLID (digits, hits, clusters, ...). Implement corresponding classes.
4. Write a conditions database service, if possible with CCDB backend
5. Start porting/implementing algorithms, starting with simple ones
   - GEM & calorimeter digitization
   - GEM cluster finder
   - Basic calorimeter cluster finder
   - Similar for Cherenkovs
6. Decide on preferred software packaging, platform support, build system, etc.
7. Research tracking algorithms (NB: our tracking problem has already been solved somewhere!)

Conclusions

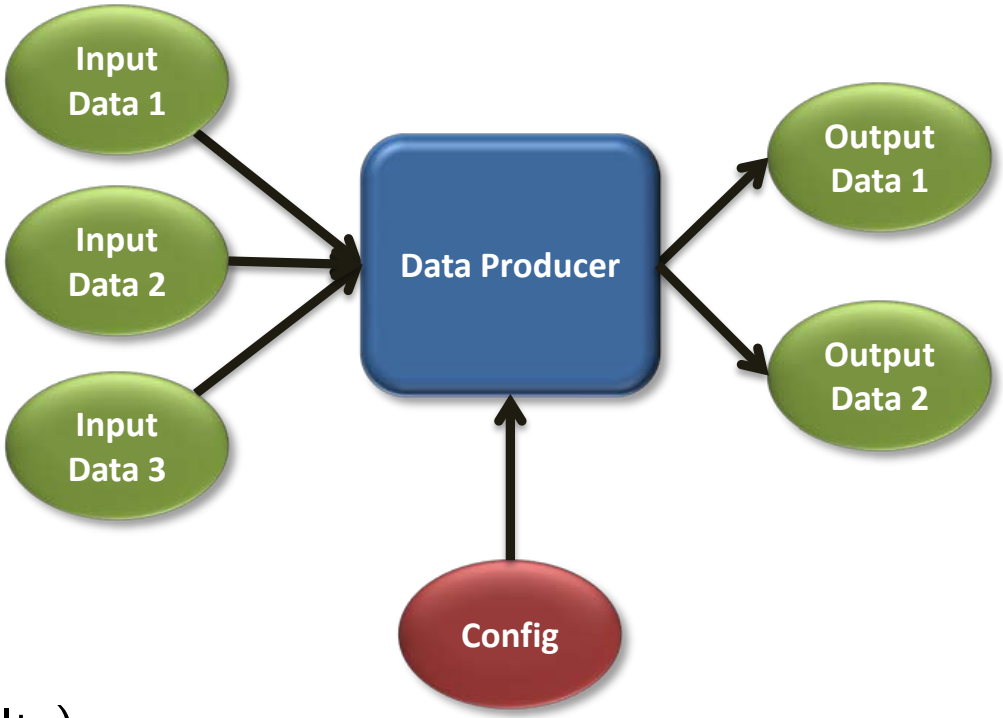
- Considering many factors, overall *art* appears to be the most suitable software framework for SoLID that is readily available at the moment
- Test installation available at JLab
- Need to start using the framework for small prototypes to identify potential issues
- First priority should be to port the existing simulation chain into *art*. Unfortunately, this is also one of the more tricky parts.

Backup Slides

What's In An Event-Processing Framework?

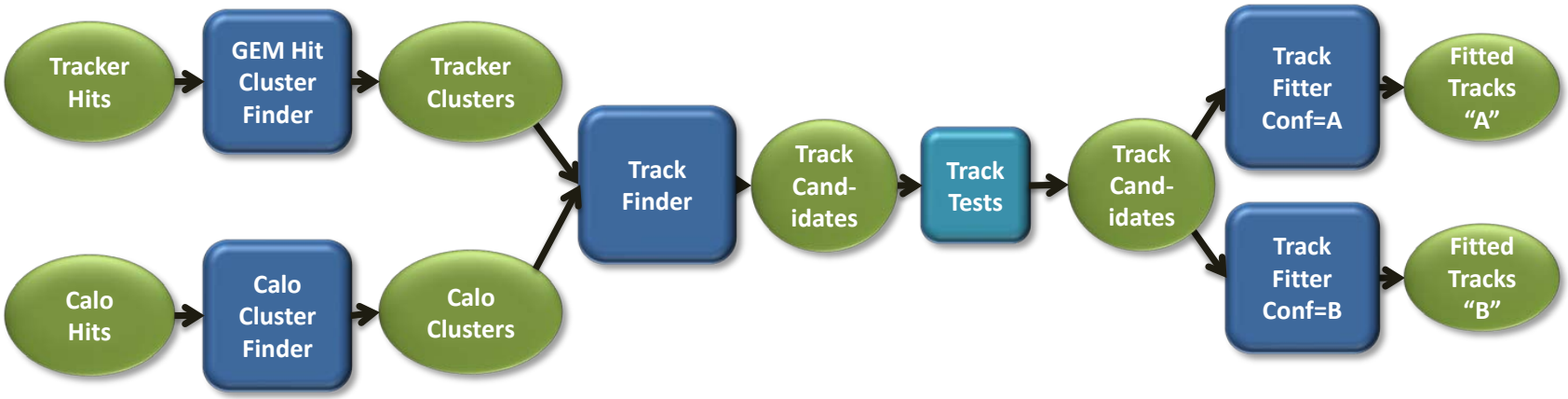
- Standardizes access: API (Application Programming Interface)
  - Event store
  - Databases (e.g. geometry, conditions, configuration)
  - Services (e.g. histogramming, messages)
  - Users should have to learn API only once
- API enforces certain restrictions (i.e. implements paradigms)
- Implements event loop (scheduler)
- Provides persistency I/O (data serialization)
- Frameworks tend to be purely technical. No Physics Here

Things We Want: Decoupled Algorithms & Data Objects



- Data objects (inputs & results)
  - Mostly "dumb data" (structs)
  - May reference other data objects
  - Hold metadata
- Data consumers/producers (algorithms)
  - Run-time configurable
  - Single algorithm per module

Things We Want: Analysis Chains



- Modules communicate exclusively via data objects
- Module relationships configurable at run time
- Multiple chains per job
- Support for condition testing modules
- Output modules for DST and histogram/ntuple files

Apparent State Of the Art Architecture

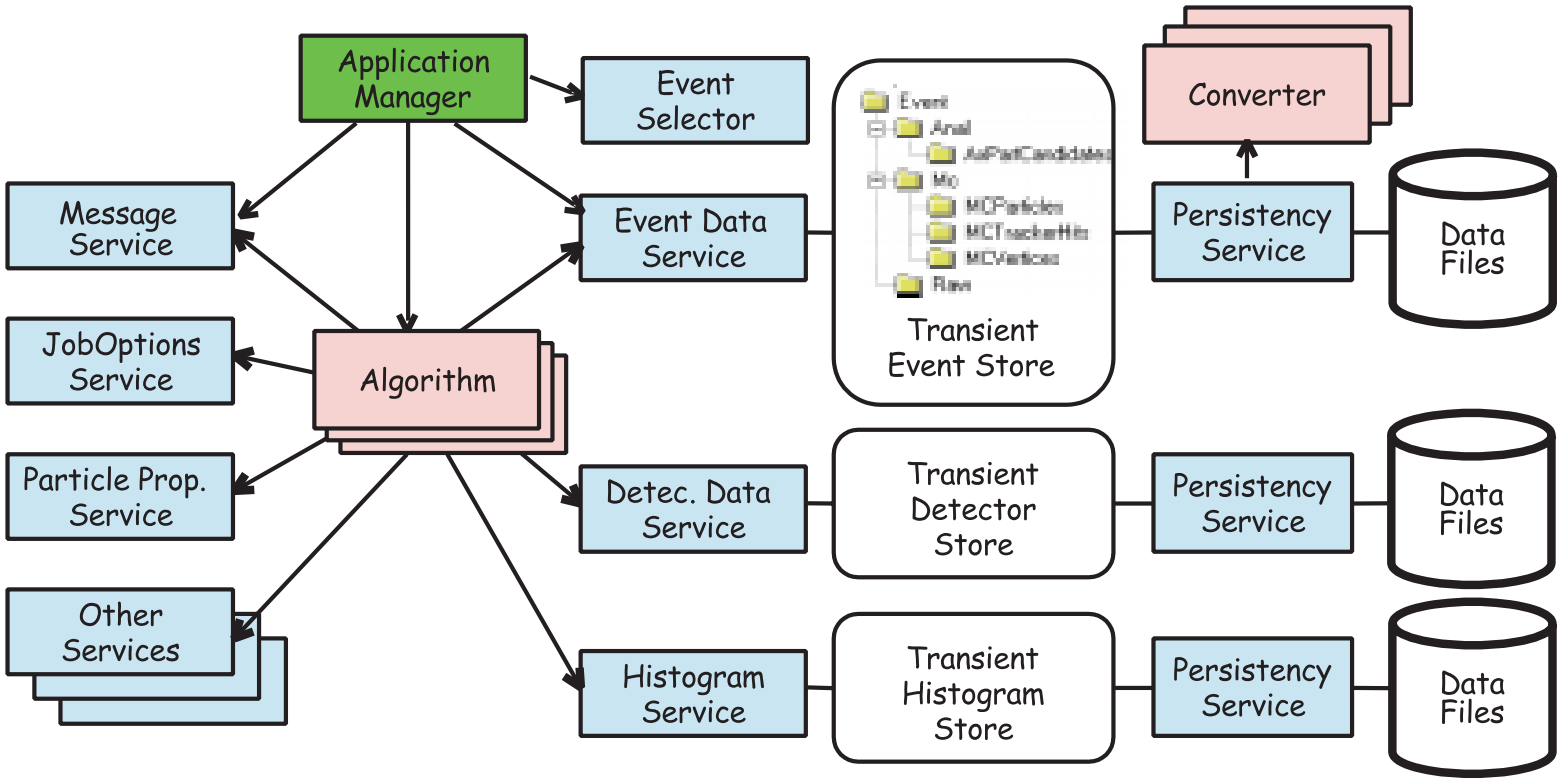


Figure 2: Object Diagram of the GAUDI Architecture

From G. Barrand *et al.*, "GAUDI - A software architecture and framework for building LHCb data processing applications", CHEP2000

Recap: Lighter-Weight Frameworks Comparison

Feature	art (FNAL)	FairRoot (GSI)	Fun4All (PHENIX)	JANA (JLab)
Origin	CMSSW (CMS)	AliRoot (ALICE)	In-house	In-house
First release	2009	2005	2003	2005
Collaborations using framework	~9	~10	1	1
Language	C++11/14	ROOT C++ (pre STL)	ROOT C++ (pre STL)	C++98
Base framework	self-contained	ROOT	ROOT	self-contained
Output, object persistency	custom ROOT	plain ROOT	custom ROOT	HDDM (XML)
Steering, configuration	FHiCL	ROOT macro	ROOT macro	command line & compiled in
Reusable/multi-instance modules	yes	(user)	(user?)	very limited
Multiple analysis chains	yes	yes	yes	very limited
Data product identification	type + 3 keys	type + producer	name	type + tag
Complexity of data object search	O(logN)	O(1)	O(N)	O(M>N)
Data provenance tracking	yes	no	no	no
Test/filter modules to skip event	yes	output module	output module	output module
Thread-safe code	yes	no	no	yes (partial)
Main dependencies	cet-is (3.5 GB)	FairSoft (2.8 GB)	ROOT, boost (1 GB)	Xerces XML
Preferred installation	Binary via UPD	Source (GitHub)	Source (GitHub)	Source (GitHub)
Unit tests	425	39 (high-level)	0	0
User documentation	User Guide (500p), workshops	Examples, Wiki	Examples, Wiki	Examples, Wiki, User Guide (old)
User code reusable for SoLID	some (DB, I/O)	much (Panda, EIC)	some (PHENIX)	much (GlueX)

Recap: Lighter-Weight Frameworks Services Features

Feature	art (FNAL)	FairRoot (GSI)	Fun4All (PHENIX)	JANA (JLab)
Transient event store	Event, run, subrun objects	ROOT folders	Phool Node Tree	With producers
Persistency Service	custom ROOT I/O	plain ROOT I/O	custom ROOT I/O	(not part of JANA)
Folders in event store	no	yes	yes	no
Event Data Service	template function	TClonesArray	template function	template function
Message service	yes	yes	no	yes
JobOptions Service	FHiCL API	FairRuntimeDb	no	ParameterManager
Geant4 integration	artG4	VMC	yes (?)	no
Detector Data Service (geo)	no (service API)	no (geo classes)	no	JGeometryXML
Detector Data Service (cond)	no (service API)	no	no	JCalibrationCCDB
Histogram Service	TFileService	no	HistoManager	no
Interactive mode	no	yes	yes	no
Configuration test	yes	no	no	no
Memory tracker	yes	ROOT memcheck	no	no
Polymorphic data objects	yes	yes	no (?)	yes
Inter-object references	art::Ptr, art::Assns (1-1, 1-N, N-N)	TRef, FairLink (?)	integer indices (?)	integer indices