Overall SoLID Software Design

Ole Hansen

Jefferson Lab

SoLID Collaboration Meeting
June 8, 2020

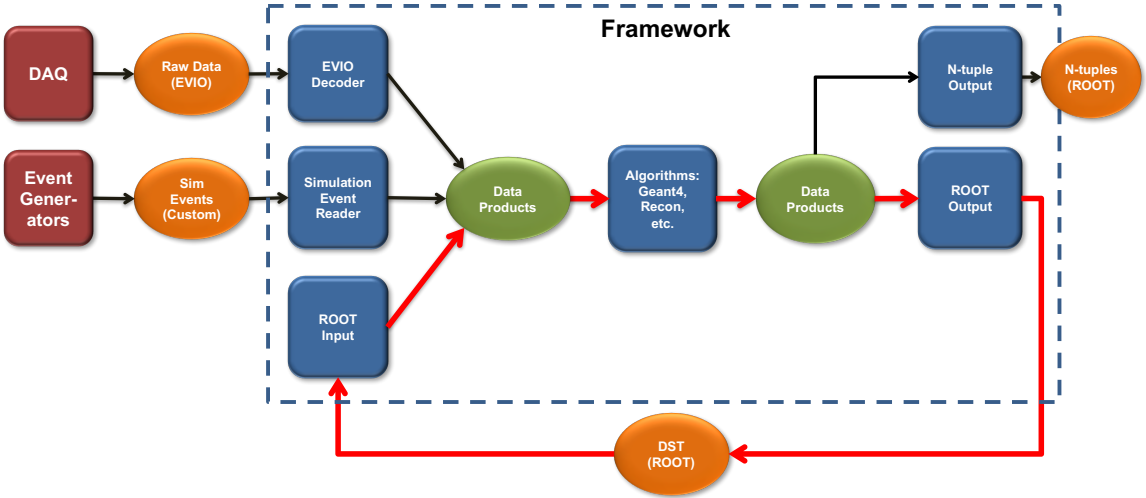
Guiding Principles

- User-friendly
  - Reduce software-related overhead as much as possible
  - Flatten the learning curve for new collaborators
  - Do physics, not coding
- Flexible
  - Assume experimental configurations and computing environments will change.
  - Avoid tying design to current fads, custom technology etc.
  - Pick long-lived data formats.
- Performant
  - Pick scalable designs. Data volumes will grow, possibly up to 10-fold, over current estimates.
  - Generally, don't let physicists write time-critical code ;)

Mid-Term Goals

- **Infrastructure:** Put in place an end-to-end simulation and reconstruction chain
  - Integrated software environment for (almost) all parts of data processing
  - Standardizes development environment, file format, database system etc.
  - Addresses recommendations from prior Director's Review
- **Algorithms:** Develop SoLID-specific prototype implementations
  - Gain experience (both with development environment and algorithms proper)
  - Evaluate performance with simulated input
  - Assist with bench tests
  - Guide the way toward production-level code

Integrated Software Environment



Software Infrastructure Desiderata Driven by Engineering Considerations

Capability	Motivation
End-to-end simulation and reconstruction chain	Director's Review Recommendation. Consistency. User productivity
Programming language C++	Collaboration expertise
Multi-pass support (output format = input format)	Processing efficiency
Multiple independently configured instances of same algorithm	Efficiency. Comparison studies
Multiple analysis chains per job	Analysis trains. Comparison studies
Algorithm encapsulation	Portability to other frameworks
Data provenance tracking	Data management
Components run-time configurable	Usability
ROOT file format support	HEP standard. User expertise

Performance Requirements Driven by Science

Tracking and PID detector subsystems determine the actual performance, but the reconstruction software must deliver results on par with the detectors.

These are major performance parameters to be verified by simulations.

Measured Quantity	SIDIS	J/ψ	PVDIS
Momentum resolution (%)	2	2	2
Polar angle resolution (mrad)	2	2	1
Azimuthal angle resolution (mrad)	6	6	– not spec'd –
Pion contamination (%)	1	1	0.1

Building the SoLID Software Environment

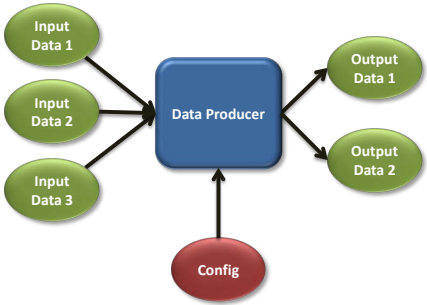
- Approach
  - Adopt pre-existing systems wherever possible
  - Mind adaptability/portability to future technologies
- Infrastructure choices
  - Pick existing HEP/NP event processing framework
  - Use ROOT for visualization, event display, interactive & physics analysis
- Algorithm choices
  - Keep physics event generators standalone, outside of framework, for performance
  - Start with standard Kalman filter-type algorithms for track reconstruction

Core Infrastructure: Event Processing Framework

Several good frameworks exist. Probably wasteful to re-invent, esp. if from scratch

Successful approach in HEP frameworks:

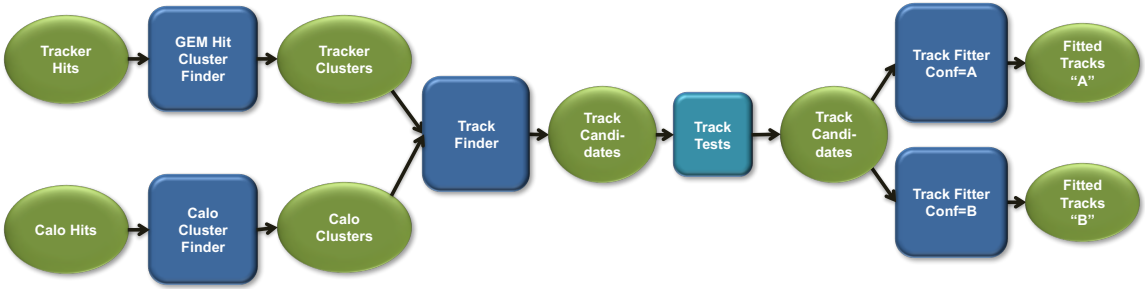
- "Separation of concerns" — data products and algorithms
- Data products are stored in files
- 20+ years track record of handling PB of data efficiently
- Yet, still not adopted in Hall A/C



Good early choice will minimize development costs.
No currently available framework offers truly full feature set.
Possible candidates:

- JANA2 (JLab, in-house). Updated version of existing JANA software from GlueX. C++. Multithreaded.
- art (FNAL). Widely adopted by FNAL neutrino program, including DUNE. C++. Multithreaded.
- Gaudi (CERN). Old, but being modernized. Widely used in HEP. C++. Multithreaded? Documentation?
- CLARA (CLAS12). Java-based. Multithreaded & distributed.

Analysis Chain Illustration



- Modules communicate via data objects → encapsulation, portability
- Module relationships configurable at run time
- Multiple chains per job & multiple module instances
- Condition testing modules select subset of results and/or skip further processing
- Output modules write user-configured subset of available data objects

Software Tasks: Infrastructure

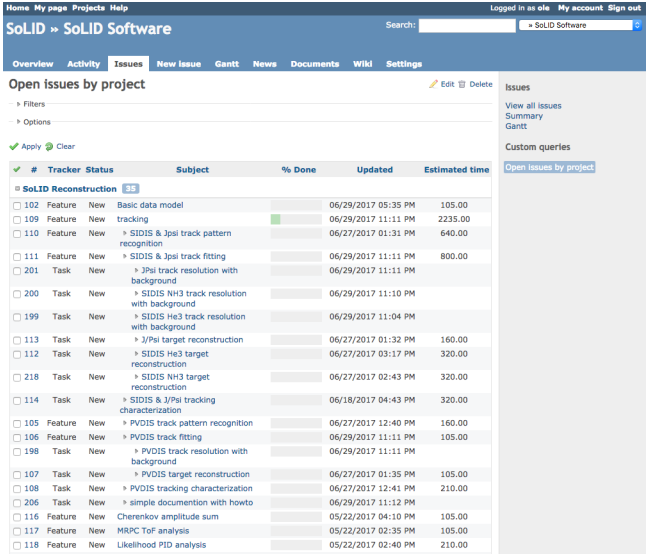
Component	Notes
Databases (geometry, conditions, mapping, materials, alignment)	Choose from several existing systems
Data model	Will evolve, need version tracking
Integrate Geant4 into framework	Reference exists *(artg4)*, test and tune
Flat n-tuple ROOT output module	Universal runtime-configurable module
Event display	ROOT/Eve reference implementation exists
Raw data format decoder	Several C++ implementations exist

Software Tasks: Algorithms

Component	Notes
Physics event generators	Several already in use for design studies
Digitization	Early versions exist. Evolve over time
Trigger emulation	Handle DAQ timing effects etc.
Track reconstruction (requires different configurations for SIDIS and PVDIS)	Prototype codes developed
Calorimeter cluster reconstruction	Pick off-the-shelf algorithm
Cherenkov amplitude summation	Pick off-the-shelf algorithm
Particle ID analysis	Traditional algorithms readily available. Good candidate for ML approach.

SoLID Software Task Tracking & Management

- Redmine task/issue tracker. Oldish, but does the job.
- Discussions boards (forums), per-issue discussions, Git support (including GitHub), Wiki, document database, etc.
- High-level task list compiled for all expected software development and data analysis work in 2017.
- Most work not yet started ...
- Need to revisit & update.



Software Project Sizes

Number of source files and Lines of Code (LOC), excluding documentation, databases and third-party tools included in repository.

```
% cloc --include-lang=C,C++,"C/C++ Header",Python,Perl,CMake,make,"Bourne Again Shell" .
```

Project	Files	LOC	Note
Hall D Sim & Recon	2358	347373	Excludes JANA
art suite 3.05	1621	136513	Bare framework
Gaudi v33r1	1501	125166	Bare framework
Hall A Analyzer 1.7	475	62806	Includes basic reconstruction
JANA2	227	24774	No/limited I/O (left to clients)
NPDet	47	7937	Excl. 3rd party & non-SoLID code

Typ. development speed 100 lines/day/developer $\rightarrow \approx$ 2 FTE-years for 50,000 lines.

Next Steps — My Preferred Scenario

- Draft a data model.
- Update `art-jlab` project (= *art* without UPS) to *art* 3
- Make simulations work with `art-jlab`. Take input from
  - `GEMC`
  - `artg4`
  - `NPDet`
- Make it user-friendly (SDK, guides etc.)

Backup

Addressing Recommendations from Previous Director's Review

- Several recommendations were made regarding software and simulations, all essentially suggesting that we implement an end-to-end processing chain.

  $\rightarrow$ We are planning to integrate practically all SoLID simulation and reconstruction code in an overarching software framework that offers common
  - data file format
  - database system
  - event store and database APIs
  - job configuration system
- Another set of recommendations suggested development of functional simulation and reconstruction routines and a simulation framework with realistic reconstruction and analysis
  - SoLID currently successfully uses GEMC as its simulation package
  - Prototype reconstruction algorithms have been developed, in particular for track reconstruction, and have been tested under realistic conditions
  - Integration of these currently separate packages into a common framework as well as further development of all software components is underway