Status of Experiment Software Architecture

Claude Saunders, AES/SSG

2011-01-20

APS Building 401, Room A1100

Presented as part of the APS InterCAT Technical Workgroup (TWG) Seminar series

ICMS: APS_1417005

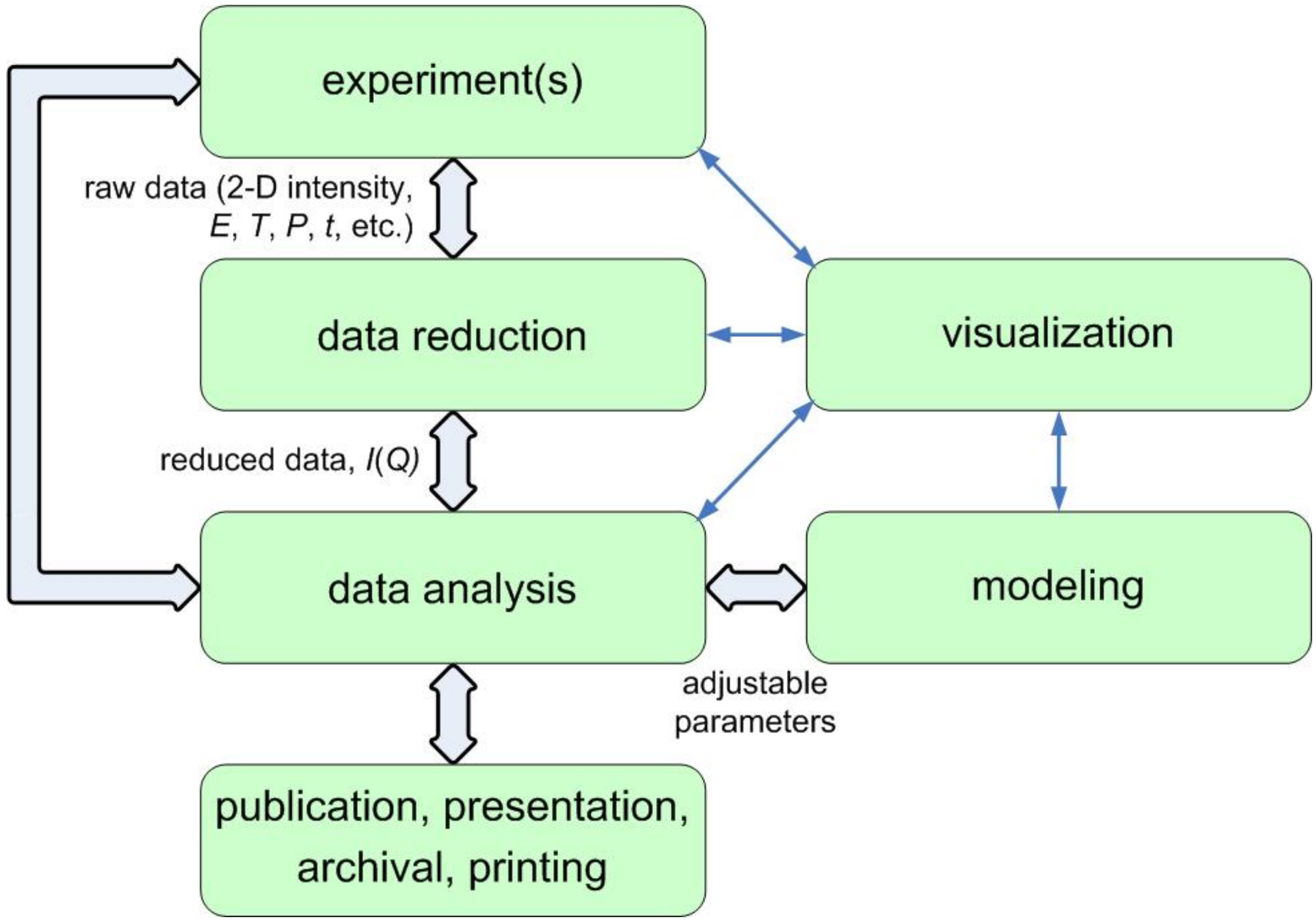
Outline

- Brief review of some previous material
- Progress and plans in selected areas
  - Standard Data Formats
  - Data Reduction and Visualization
  - High Level Experiment Control
  - Low Level Experiment Control
  - Automation/Workflow
- Questions?
- Note: Wiki pages that contain details on what SSG is doing
  - http://confluence.aps.anl.gov/display/SSG

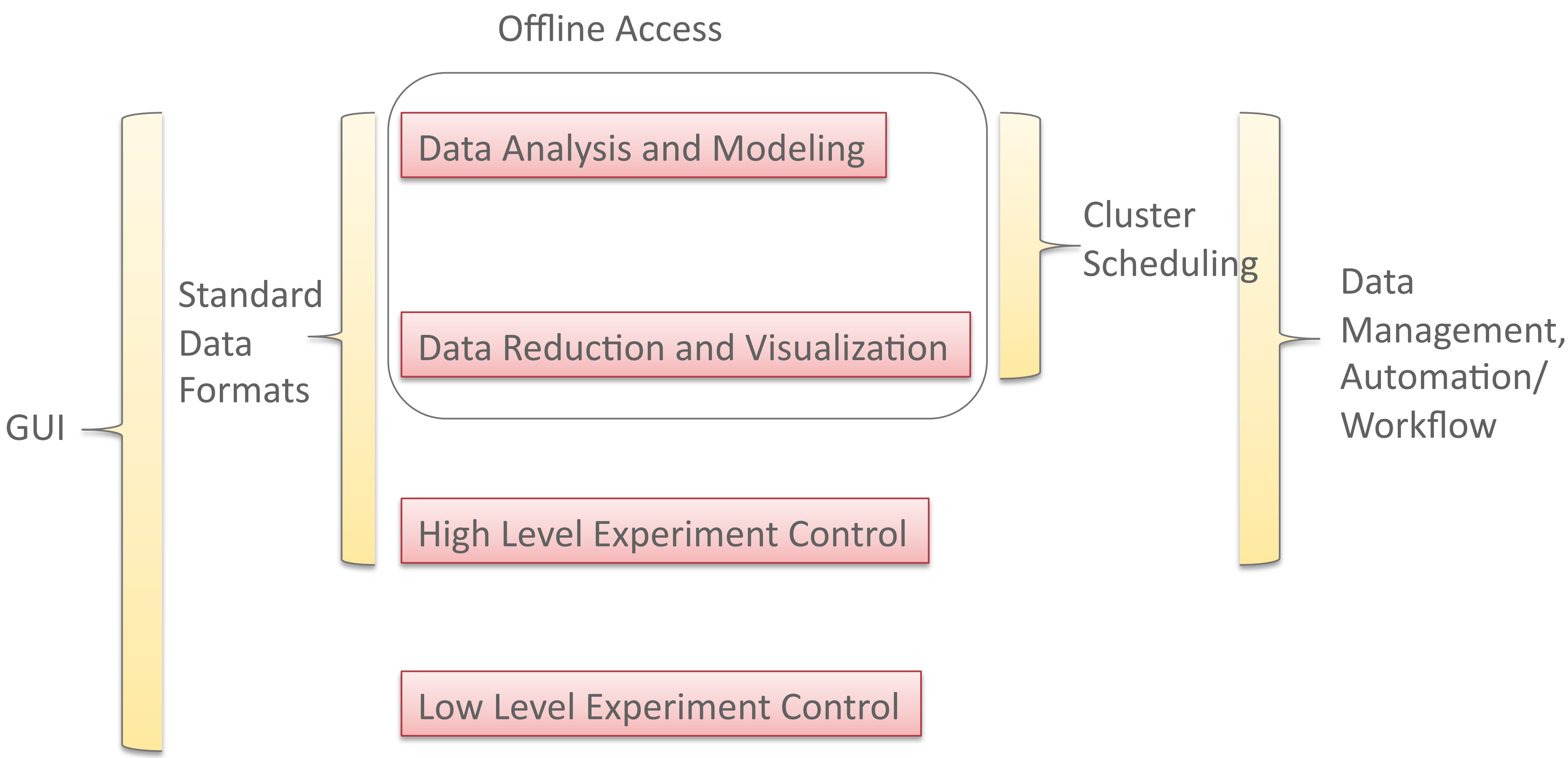
Brief Review of Previous Material

- Workflow of the Integrated Beamline
- Software Infrastructure Categories
- Current "Stakes in the Ground"

Workflow of the “Integrated Beamline”



Software Infrastructure Categories



10 Categories Total

Current "Stakes in the Ground"

- **Graphical User Interfaces (GUI)**
  - Select and develop best-of-breed, which is **Eclipse**, and make this the standard window onto other infrastructure functionality.
  - A common starting point and center…
    - "The Scientific Workbench Concept"
    - Common look-and-feel, help mechanism, distribution and upgrade mechanism, etc.
    - Start with existing CSS tool which serves as MEDM replacement
  - … but not the only provider
    - We will accommodate pure python code, including python-rendered GUI
    - We will accommodate execution of other external tools
- **Standard Data Formats**
  - **HDF5**, except where other formats necessary (MDA, custom, proprietary, etc.)
  - … with simplified NeXus compliance (more later)
  - Use of direct HDF5 libraries rather than NeXus API (NAPI)
- **Low Level Experiment Control**
  - EPICS + synApps + asyn + SPEC (no surprises here)
  - **CSS-BOY** for medm replacement

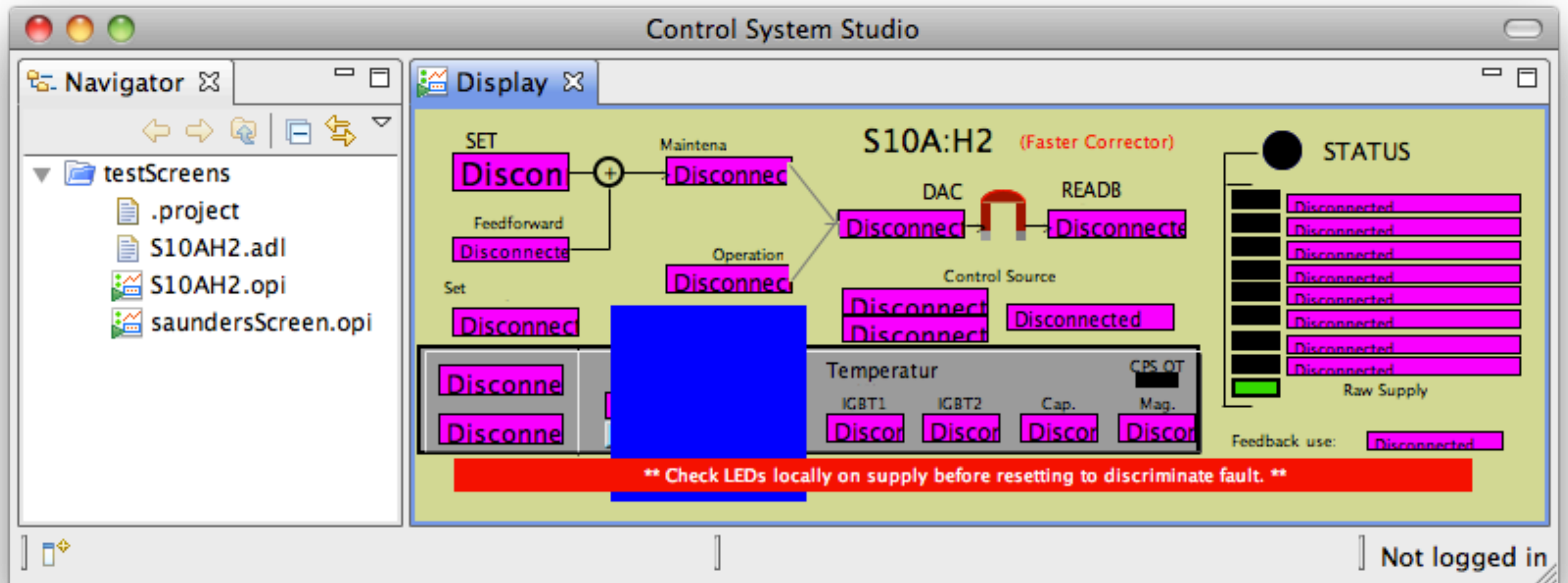
Progress and Plans in Selected Areas

- Low Level Experiment Control
- High Level Experiment Control
- Data Reduction and Visualization
- Standard Data Formats
- Automation/Workflow

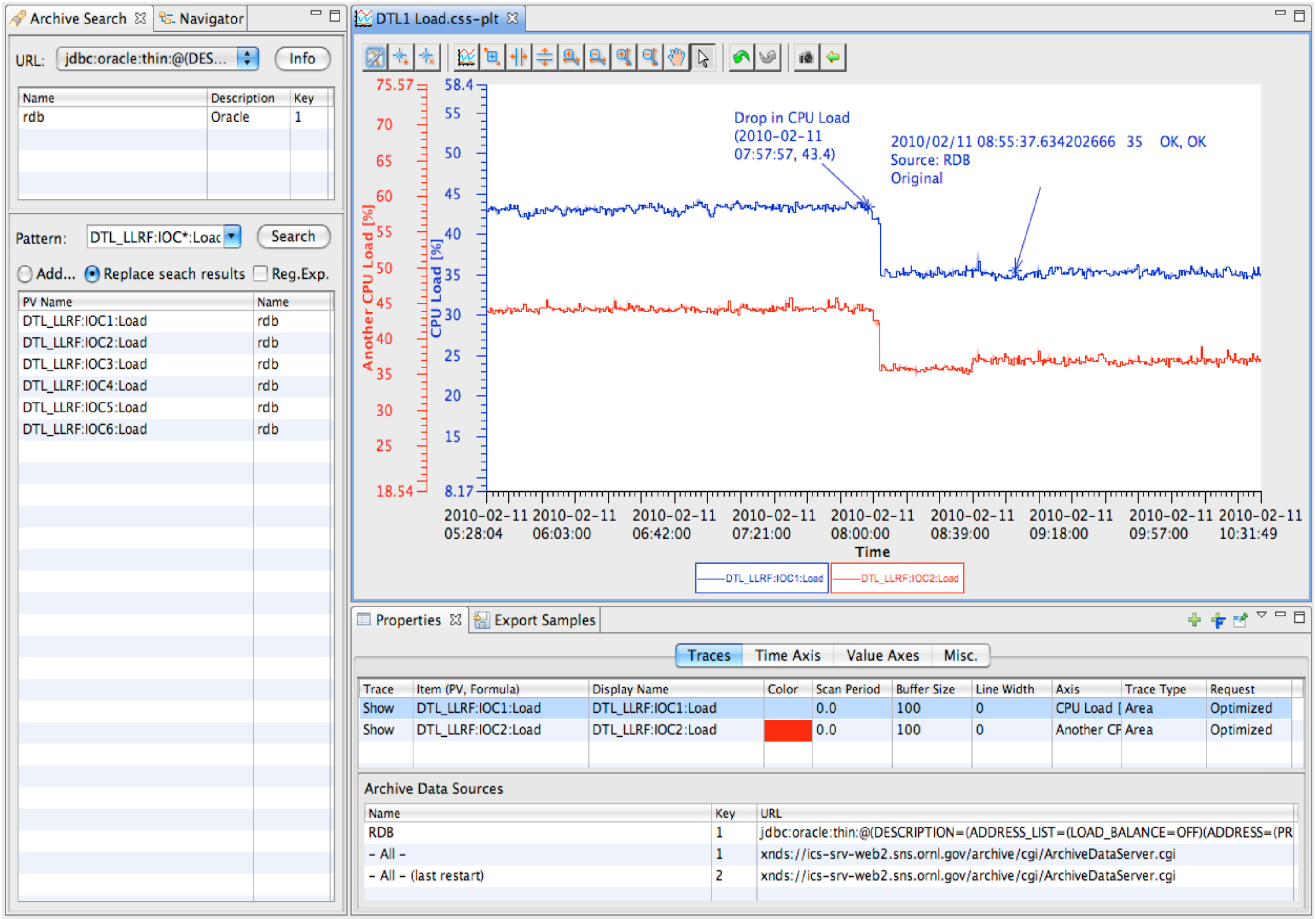
Low Level Experiment Control

- Main effort here is to come up with MEDM replacement
  - **CSS (Control System Studio)**
    - General control-system-aware platform with many capabilities
  - Eclipse-based
  - Incorporates MEDM functionality (+/-) and offers many other useful CSS plugins
- In particular, we are interested in CSS-EPICS product that contains
  - **BOY (Best OPI Yet)** – this is the plugin which replaces MEDM
  - Data Browser – a PV trending tool with live plus archived data
  - **ADL to OPI** converter created and maintained by John Hammonds
- **Current Work**
  - Further enhance ADL to OPI converter with more advanced color and font mapping
  - Convert synApps screens and make OPI versions available alongside ADL
  - Explore enhanced functionality for synApps using advanced CSS-BOY features
  - Get SSG/BCDA blessed version available for use on beamlines

CSS BOY Screenshot
(of converted SR corrector screen)



CSS Data Browser Screenshot



High Level Experiment Control

- Goal:
  - to make the task of designing, setting up, and executing data acquisition easier…
  - … while at the same time integrating seamlessly with data reduction, visualization, and analysis (including cluster scheduling, data management, etc.)
  - … and making sure all relevant meta-data is kept with raw detector data

- Early effort involved evaluating openGDA (Generic Data Acquisition) project from Diamond Light Source. Decision not to use GDA was made recently.
  - Difficult to build and deploy – hence difficult for us to support with small staff
  - Not easily integrated with other clients such as IDL, Matlab, python
    - Very important at APS with existing legacy of EPICS integrated tools

- But! – GDA has very good ideas about scripted acquisition and high level scanning support which we have learned from

High Level Experiment Control (2)

- Plan B – Develop our own Scanning Engine
- What is a **Scanning Engine**?
  - Note: we're still working on defining this, so bear with me here…
  - Similar to synApps sscan and saveData, only an "outside-the-IOC" solution
  - Supports:
    - Scriptable as well as static configuration of scans
      - Step scans, fly scans, oscillation scans, tracking scans
    - manages their execution (abort, rewind)
    - exposes status of scans for integration with external clients like CSS/medm, python, matlab, IDL
      - Soft PVs for control/monitoring of scans
      - Async notification of scan start, step start/end, end of scan dimension, scan end, etc…
    - Tracks scan data set – any client of scan engine can persist in required form

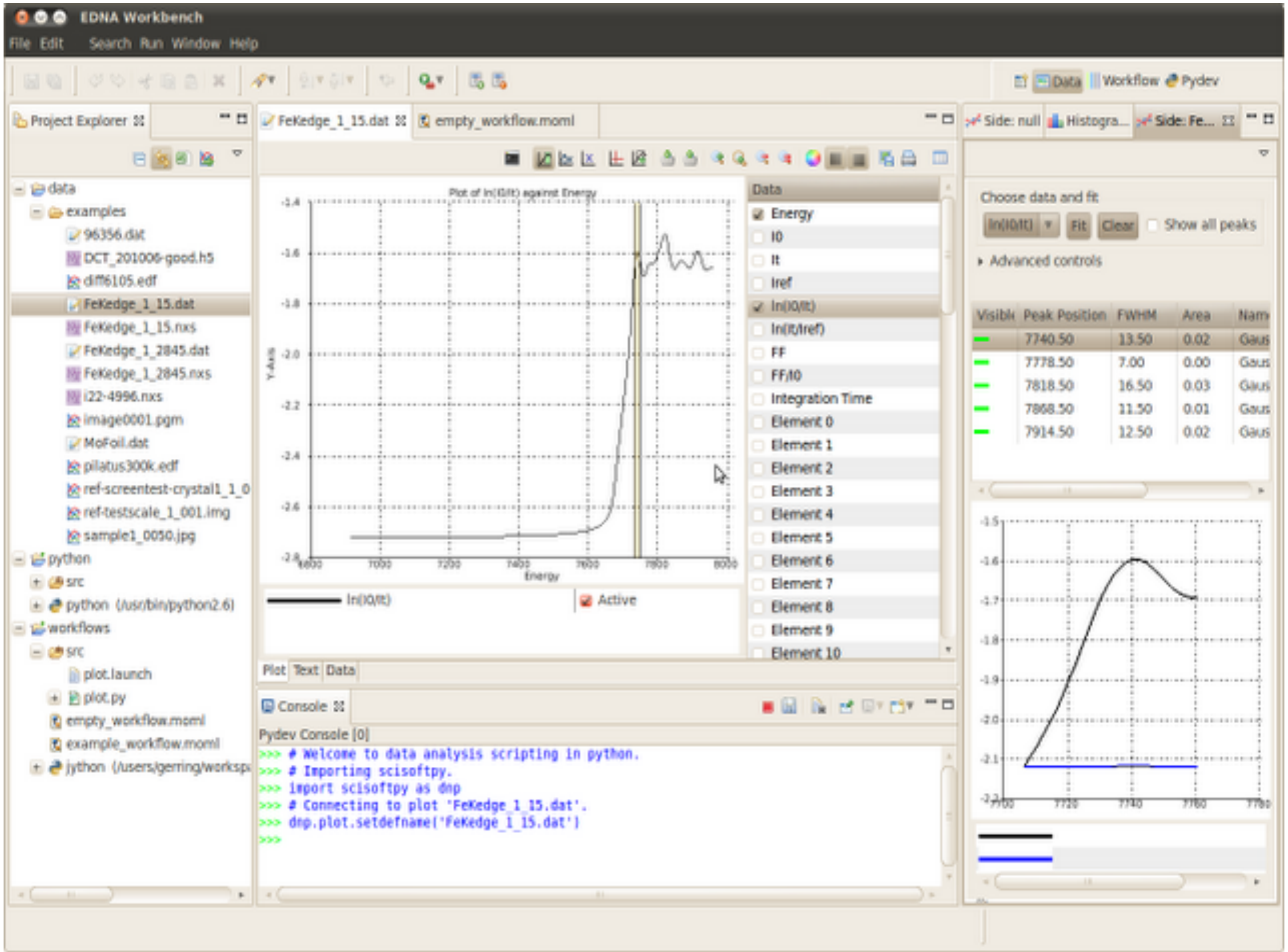
- **Current Work**
  - Building prototype scanning engine for simple step scans initially
  - Evaluate prototype with respect to larger requirements
  - Characterize prototype's performance envelope

Data Reduction and Visualization

- Evaluating SDAWorkbench (Scientific Data Analysis) project from Diamond Light Source
  - SDA is offline (post-acquisition) data analysis
  - Part of larger GDA codebase, but governed as a separate project
  - Developed under guidance of collaboration MOU between Diamond, ESRF, and now APS
  - Eclipse-based workbench
    - But not part of CSS (Control System Studio)
- SDAWorkbench
  - www.sda-workbench.org
  - Not formally released yet
  - Initial features set
    - Plotting, Diffraction image viewer, NeXus explorer, workflow, jython and python scripting
  - First collaboration meeting in February to establish quarterly development cycle
    - Will decide on development plan for first formal 3-month development cycle

- Main interest here is a console from which to create and run analysis sequences

SDA Workbench Screenshot



Standard Data Formats

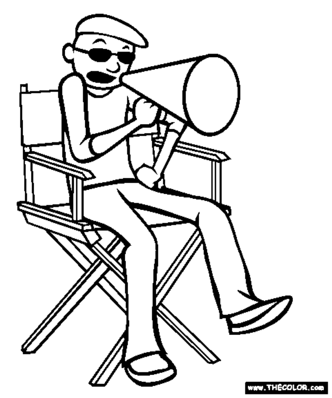
- Goal:
  - develop standards, conventions, and libraries to support direct use of HDF5 with NeXus storage conventions
- Important
  - NeXus is a combination of
    1. A set of schema that establishes standards for how to store data
       - Ex. How to store n-dimensional dataset with axes fully described
       - Ex. How to store a description of a detector
    2. Governance to allow introduction of new schema
    3. An API called NAPI that supports writing NeXus compliant files to XML, HDF5, and HDF4
  - We don't have to do (3), and we can pick and choose subset of (1) that works for us

- **Current Work**
  - **NXexchange** – simplified NeXus schema for simple image and spectra exchange
  - Lightweight libraries on top of raw HDF5 libraries for reading/writing NeXus-compliant data/meta-data to HDF5 files – no NAPI
  - Documentation to make the above easy to use

Automation/Workflow

- **Condor** evaluated
  - Very powerful tool, but very much geared towards large jobs (10 mins +)
- **Swift** (for post-acquisition analysis workflow)
  - Developed at Argonne's MCS division
  - Lightweight analysis workflow engine with simple scripting language(s)
  - Good at automatically exploiting basic parallelism
  - Easy to run same job locally on laptop, or on collection of high end systems
- **Passerelle** (part of SDA and also used by Soleil for data acquisition)
  - Lightweight, based on flexible PtolomeyII workflow engine (so is Kepler)
  - Can be used to orchestrate reduction/analysis pipelines during acquisition (unlike Swift)
  - Can also be used post-acquisition to orchestrate analysis jobs

- **Current Work**
  - Downloading and learning above tools
    - Demo of Condor and Swift done at SSG BLT (lunchtime tech talk)
    - Would like to do this with Passerelle as well
  - Exploring use of Swift for LDRD: Next Generation Data Exploration



Areas Not Actively Being Addressed
(other than existing work on XPCS, HEDM, and Tomo)

- Cluster Scheduling
- Data Management
- Offline Tools
- Data Analysis and Modelling

Data Analysis and Modeling (extra slides from previous presentation)

- Participate in cross-disciplinary, technique-specific working groups
  - Educate and support MVC (Model-View-Controller) design in analysis/modeling/simulation codes.
  - Educate on other design patterns to improve modularity in codes.
  - Recommend and support libraries for data formats (NeXus, others), parallel processing (MPI, openCL?), etc.
  - Support execution environment to allow for both single-host and cluster-based execution.
  - Support for writing new codes or re-factoring existing codes.
  - Support for parallelizing sequential codes.
  - Support for wrapping codes for inclusion in workflow engine.
  - Support for making use of relational databases for metadata storage and query

Data Management
(extra slides from previous presentation)

- Experiment Data Catalog
  - Implemented using relational database
  - Not for storing all data, but as index to file-based data
  - For each sample put in beam, track:
    - Sample id
    - Acquisition parameters
    - Originating proposal, scheduled experiment, and ESAF
    - Data reduction and analysis history
    - Analysis execution logs
    - Data file generation and movement
- Generic web portal to locate experiment data
- Beamline Notebook
  - View and annotate context of an entire experiment
- Data Transfer
  - Agents such as GridFTP for data transfer
    - Within APS facility (ex. From beamline data acquisition disks to HPC cluster)
    - Between APS and MCS and ALCF
    - Between APS and experimenter's home institution

Cluster Scheduling
(extra slides from previous presentation)

- 2 Modes
  - Reservation-based Scheduling
    - Compute cores and disk space set aside for dedicated use
    - For on-demand computations during experiment
    - For on-demand computations before/after experiment
    - This is current mode for existing APS HPC clusters
      - Batch scheduling queue fronting n cores dedicated to a sector
    - Expand to support reserving cluster resources using beamline scheduling system
  - Opportunistic Scheduling
    - For computations outside scheduled experiment
    - Compete for unused resources
    - Can include single-host systems (ie. Using Condor)
  - Will evaluate and deploy cluster scheduling that meets both needs.

  - Also will support job submission to Argonne MCS and ALCF clusters.

Offline Tools
(extra slides from previous presentation)

- Support for packaging up
  - Data
  - Applications
  - Execution Logs
- For use at home institution

- This will require:
  - Offsite software distribution mechanism
    - Eclipse update site
  - Packaging code into one click installers where possible
  - Designing (and testing) applications so they don't depend on APS network environment