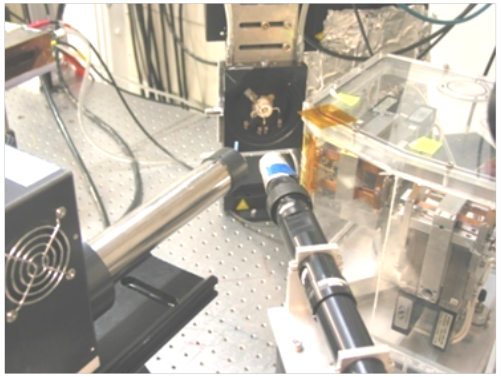
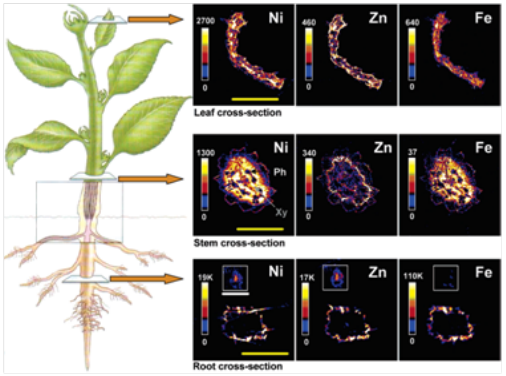
# Data Collection for the GSECARS X-Ray Microprobe

Matthew Newville

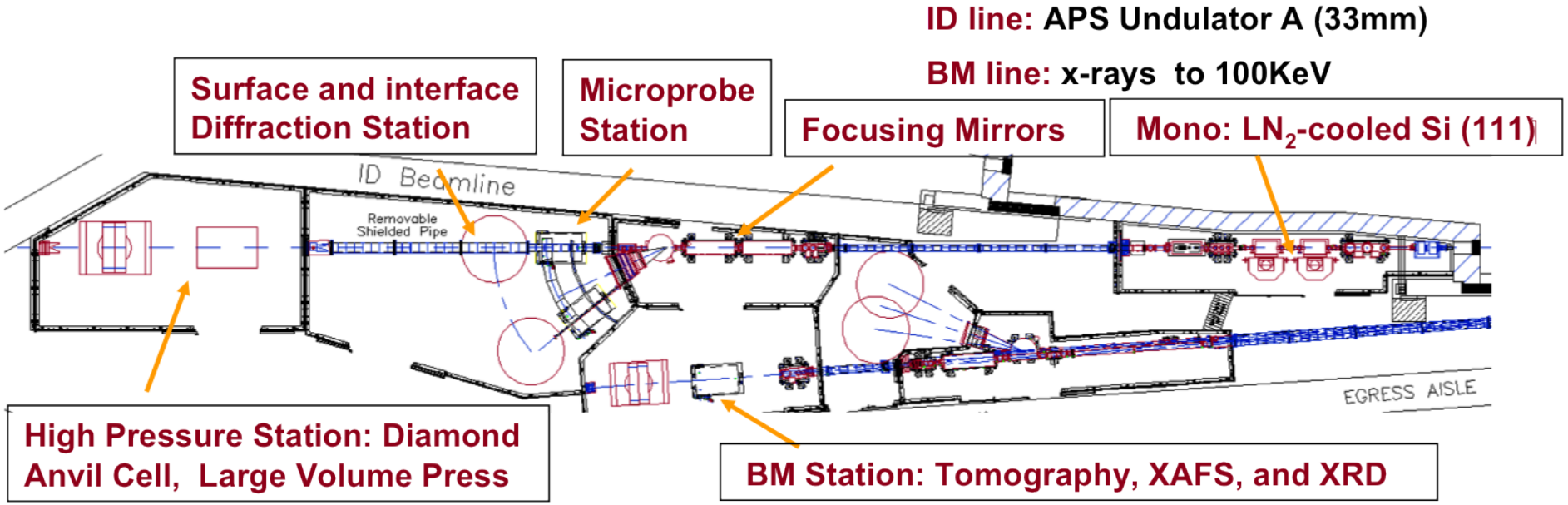
Center for Advanced Radiation Sciences
University of Chicago

May 25, 2011

# GSECARS

GSECARS is an APS sector, running a National User Facility (NSF and DOE support) for Geological, Soil, and Environmental Sciences.



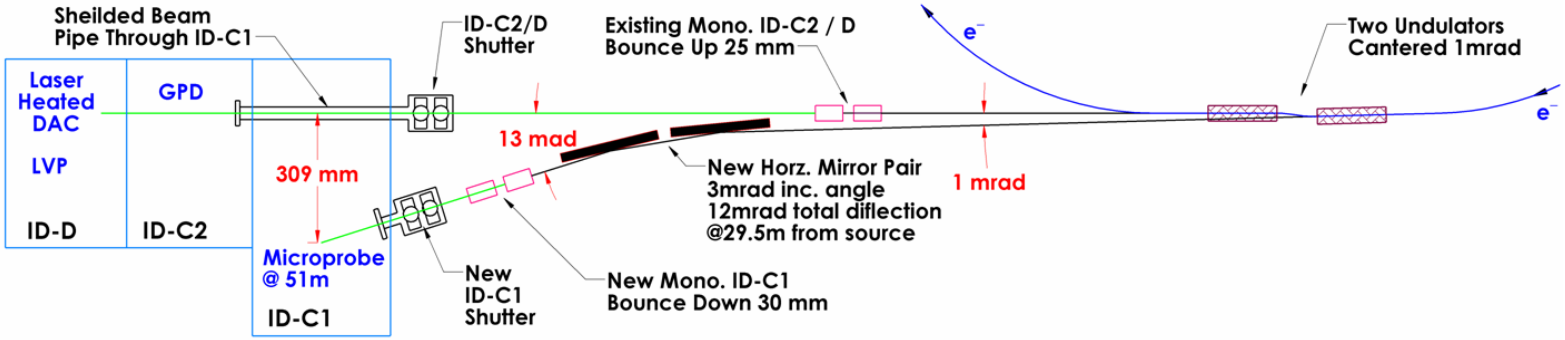

- All beamtime scheduled by reviewed General User Proposals.
- $\sim 1/2$ High Pressure: Large Volume Press, Diamond Anvil Cell
- $\sim 1/2$ Geochemistry: Surface diffraction, X-ray Microprobe, Tomography.
- X-ray Microprobe ($\mu$XRF, $\mu$XAS, $\mu$XRD) runs $\sim 1/4$ of ID time and $\sim 1/4$ of BM time.
- In operation since 1997, heavily over-subscribed.

# GSECARS Upgrade: Canted Undulators

Encouraged by funding agents and the APS SAC, GSECARS is upgrading

APS will install 2 undulators at sector 13. Cant angle $= 1\,\mathrm{mrad}$.

GSECARS will build dedicated X-Ray Microprobe, upgrade optics for high-energy line.




## High Energy Beamline

- 30 mm ID
- 5.5 – 60 keV, Si(111), Si(311)
- current beamline mirrors
- $15 \times 40\,\mu\mathrm{m}$ ($\sim$ full ID source)
- DAC, LVP, Surface Diffraction

## Dedicated X-Ray Microprobe

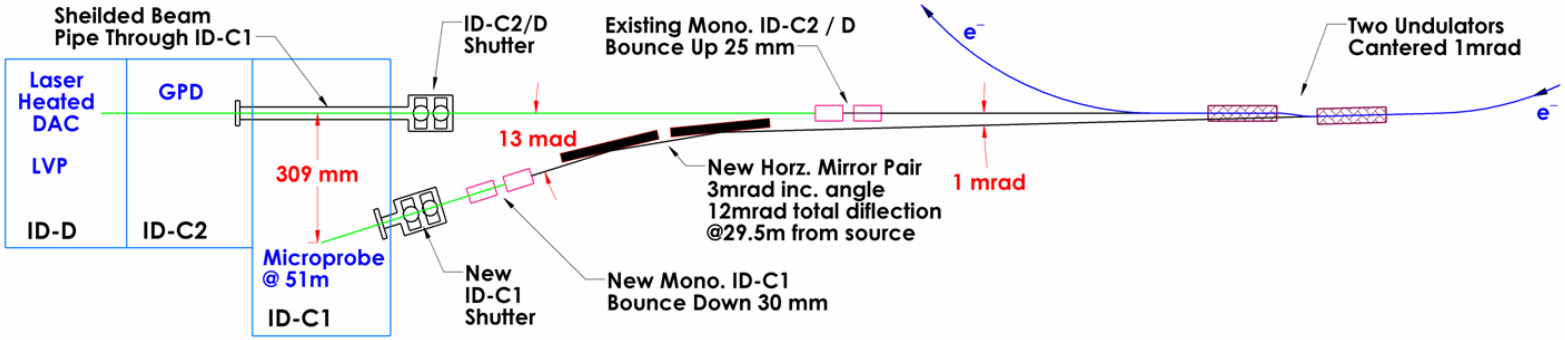
- 36 mm ID
- 2.35 – 27 keV, Si(111), Si(311)
- 2 horizontal mirrors, 3 mrad each
- Secondary Source Aperture.
- $0.5 \times 0.5\,\mu\mathrm{m}$ beam-size
- 309 mm beam separation in end-station.

# GSECARS Upgrade: Canted Undulators

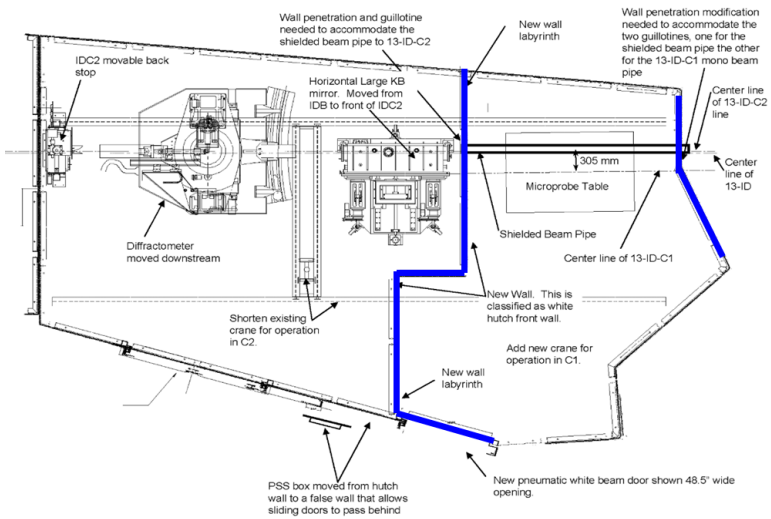
Encouraged by funding agents and the APS SAC, GSECARS is upgrading

APS will install 2 undulators at sector 13. Cant angle $= 1\,\mathrm{mrad}$.

GSECARS will build dedicated X-Ray Microprobe, upgrade optics for high-energy line.




| | |
|---|---|
| Summer 2009 | Project awarded (NSF, DOE, NASA). |
| Fall 2010 | Hutch modifications: ID-C $\rightarrow$ ID-C, ID-E. |
| Fall 2011 | Install ID, Optics No ID Beamtime. |
| Winter 2012 | Commissioning. |
| Summer 2012 | Normal Operations, 2 independent ID stations. |



# GSECARS X-ray Microprobe: APS 13-ID-E

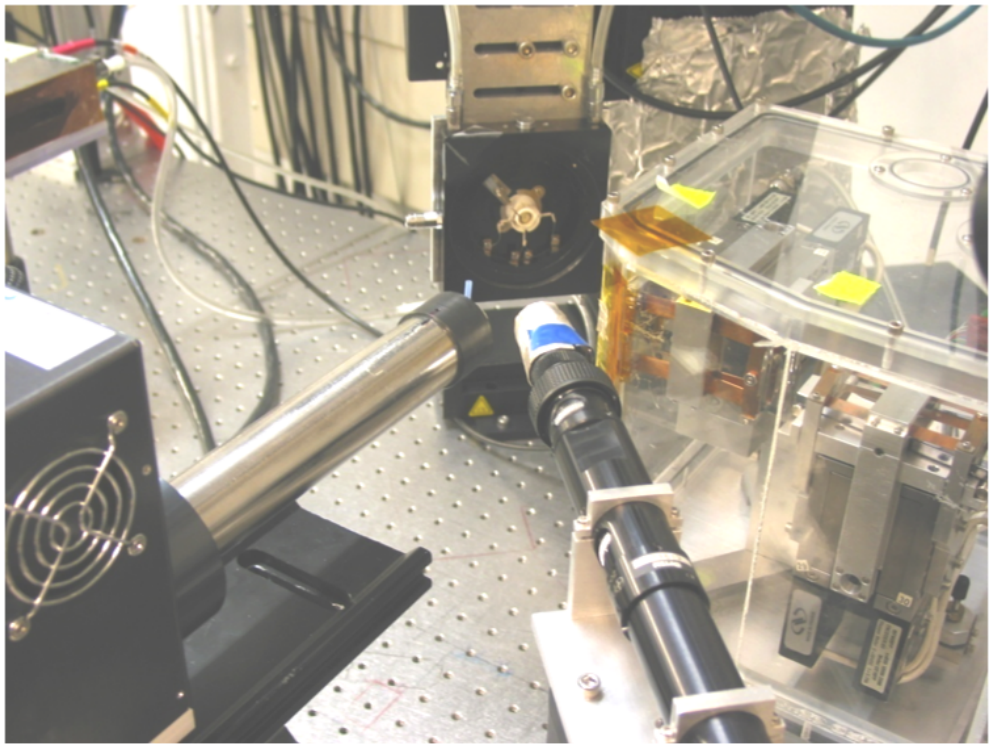
**Focusing Optics** K-B Mirrors, 100mm

**Sample Stage** x-y-z-$\theta$, 0.1 $\mu$m resolution,
150mm travel range,
heating, cryo stages.

**XRF Detectors** Vortex ME-4 silicon drift,
16-element germanium,
Wavelength-Dispersive Spectrometer.

**XRD Detectors** MAR 165 CCD

**Microscope** 5-20x video system



**Before Upgrade**

- 25% ID time (50 days / year)
- 4.5 – 35 keV, Si(111)
- $1 \times 2\,\mu$m ($\sim 10^{11}$ photons/sec)
- poor mono stability

**After Upgrade**

- 100% ID time (200 days / year)
- 2.4 – 27 keV, Si(111), Si(311) *S, Cl, Ca!*
- $0.5 \times 0.5\,\mu$m ($\sim 10^{11}$ photons/sec)
- greatly improved mono stability

# What about the Control and Data Acquisition Software?

The data acquisition software is ¿ 10 years old, and also needs an upgrade!

System for XRF Mapping, XAFS Scans, XRD collection (as of 2009):

- EPICS as basic control system.
- SScan Record for Most Data Collection: XRF Mapping, XAFS Scans
  Gives some limitations, greatly influences system.
- GUIs in IDL: OK, but clunky.
- Macros in IDL: Challenging for novice users.

---

This approach lacks many desirable features:

- Poor Remote Access: 1 IDL session on 1 machine is in charge.
- Proprietary Tools: Uses IDL, might as well be Matlab, LabView, or Spec.
- Difficult to Port: ported to 1 non-GSECARS beamline, abandoned.
- Not Well-Coordinated: with analysis tools. Challenging "Workflows".
- Scan-centric: Doesn't think of Sample or Experiment as a whole.
  Epics SScan record too influential (how motors and detectors are selected).
- Difficult Scripting Even "easy IDL" macros difficult for users to write, modify.

# What do we want for Data Acquisition Software?

Design Criteria for Ideal Data Acquisition Software:

- **Free Software:** for the entire stack – maximize portability.
- **Distributed Client-Server:** Many clients should have access to server: Desktop GUIs, Web Pages, Smartphones
- **Script/Command-Based:** Clients send high-level commands to server which does all the real work.
- **Scriptable:** high-level commands can become acquisition scripts.
- **Easy Configuration:** of beamline-specific and experiment-specific features.
- **Data Management:** Organization of data files, metadata into electronic lab book.
- **EPICS agnostic?:** At least try to separate EPICS layer when possible.

---

**Use a Relational Database Management System**

Relational Databases (MySQL, Postgres, SQLite, Oracle) are robust data storage and retrieval systems, and solve many issues for modern data acquisition.

- **Distributed Client-Server:** Multiple Clients, no problem. Communication layer done.
- **Security Model:** Done. Can limit access by user, machines.
- **Other Benefits:** Automatic History, Logging, Data Integrity, Standard Technology.

# Relational Database for Data Acquisition Software

Relational Databases (MySQL, Postgres, SQLite) solve many issues for modern data acquisition and are excellent for storing "metadata".

Many Protein Crystallographers beamlines use database systems:

- **Bar-Coded Sample Vials**
- **Robots** Look up which Sample is next, load it.
- **Quick Measurement** can be analyzed in a separate process to decide if more data collection is worthwhile.

---

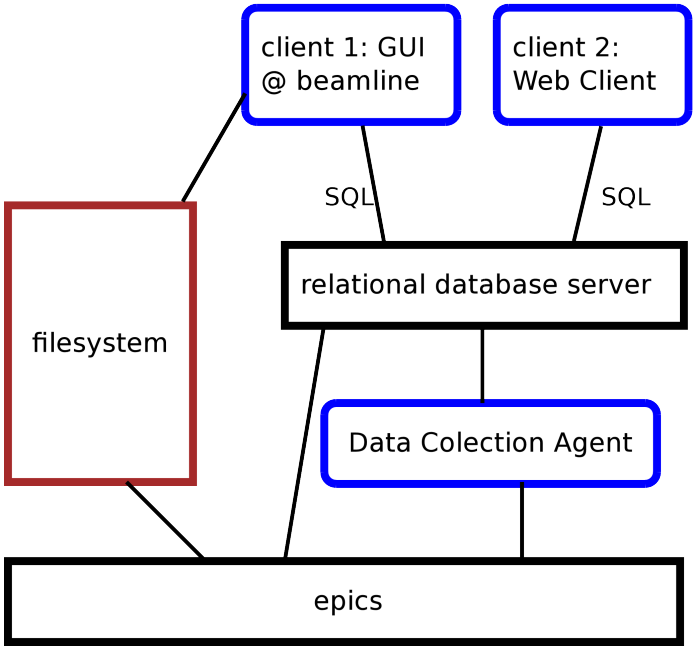
What goes into the acquisition database?

- **Configuration** of detectors, monochromators, motors, other instruments.
- **Macro code** that users are allowed to run.
- **Command Queue** list of commands that have run, yet to be run.
- **State information** what is being done now, what has been done, current scan data
- **Ancillary Data** external data (Ring Current, Room Temperature).

Real Scientific Data (ie, from detectors) could go to database.
Better to let detector write data files, with filenames in the database.

# RDBMS for Data Acquisition

## Data Collection Schematic




Client sends commands (macros) to DB with SQL over the network.

Commands are Queued into a Table.

Collection Server sees next command, launches a Collection Process or Thread, may wait for it to finish, then looks for next command to run.

Clients can read status, data from DB with SQL or file system, or use EPICS.

Collection Processes and Threads write datafiles (TIFF, ASCII, HDF5, . . . ) to file system and status info back to DB.

Users could takes home SQLite version of the Database for the whole experiment.

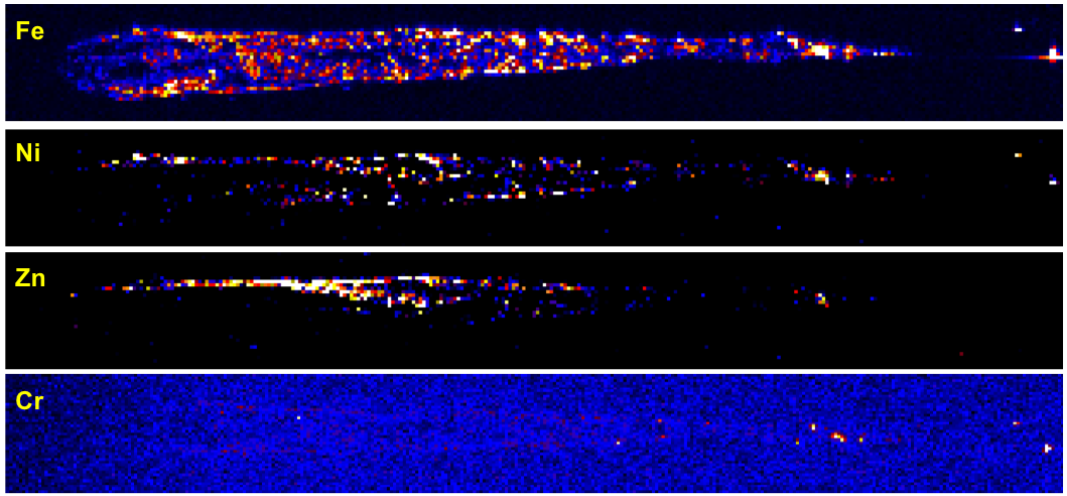
Data Analysis Programs can use SQL to query this, and add analysis information.
Data sets could be extracted to other databases or applications.

A lot of work, but step-wise progress possible.

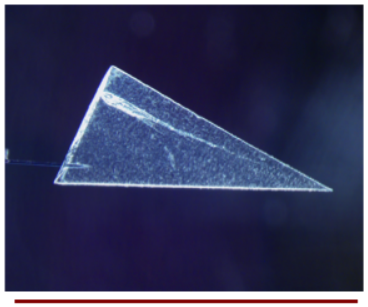
# XRF Mapping: the Old Way - Step and Dwell

Improvements in technology have changed the way we collect XRF Maps.

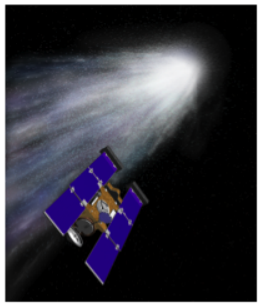
2006:




XRF Maps of samples returned from NASA's Stardust mission provided fragments of the comet Wild-2: tiny tracks in aerogel.



5 mm



Detector: Single-Element Vortex silicon-drift detector

Map Details: 331 x 44 pixels, 3 $\mu$m

Acquire Time: 2 sec/pixel + $\sim$ 0.5 sec/pixel to save full XRF spectra.

Total Time: 10 hours.

Minimum Time Per Pixel: 0.5 second

# Better XRF Detectors, Better Motion Control

## Detectors

High-Speed Detectors: SIINT Vortex-ME4: 4 silicon-drift diodes have decent total area ($\sim 170\,\mathrm{mm}^2$ ), and total count rates up to $8 \times 10^5\,\mathrm{Hz}$ with very good ($< 150\,\mathrm{eV}$) energy resolution.

MCAs with Memory: XIA xMAP/Mercury DXPs have on-board, double buffered memory, allowing continuous collection of 4 x 2048 XRF spectra at 1 kHz:

1 msec dwell time for 4 XRF spectra: $\sim$ 800 counts

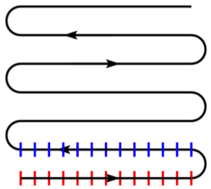
## Motion

High Precision Stages / Smart Controllers: Newport VP-25XL with XPS Controller: 25mm travel at $\sim 20\,\mathrm{nm}$ resolution).

bi-directional scans continuous back-and-forth scans

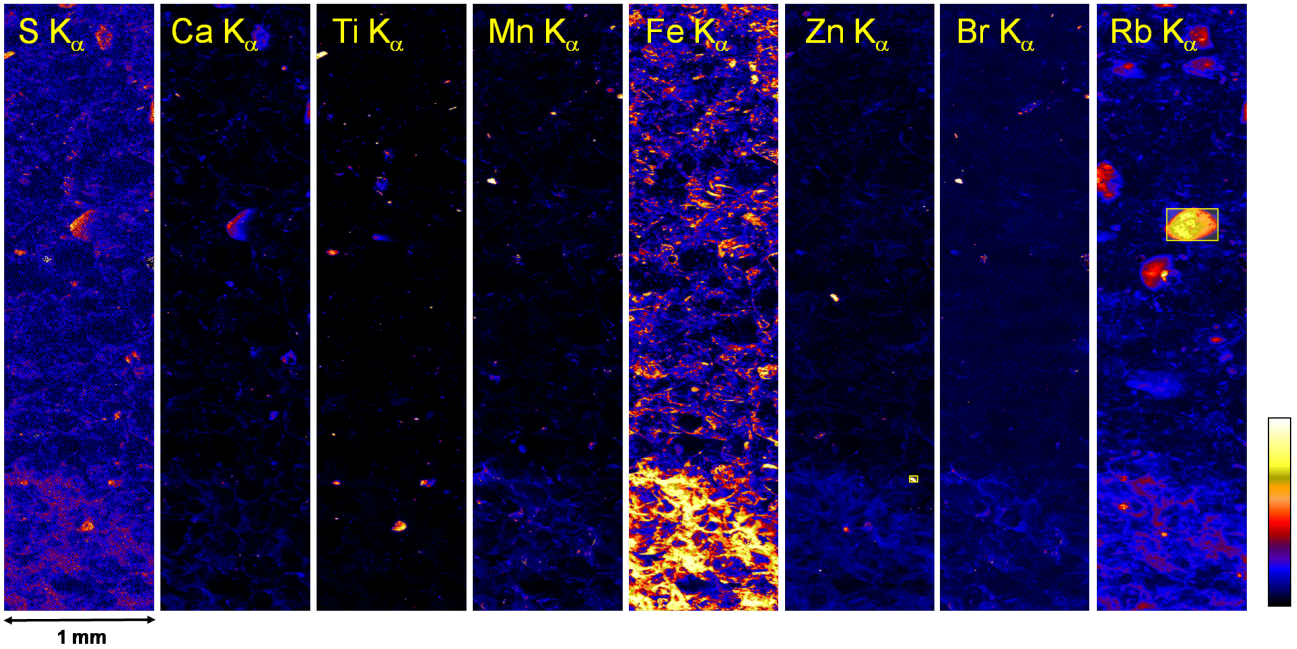
high accuracy user-defined "bin size" down to 20 nm

drive data collection output pulses trigger the XRF collection



# Fast XRF Mapping: Continuous Scanning

2010:




XRF Map (1 x 4 mm) of natural sediment sampled near zinc smelter facility in Mortagne du Nord, France

G. Morin, F. Juillot
Univ Paris VI

Detector: 4-Element Vortex ME4 silicon-drift detector

Map Details: 201 x 801 pixels, 5 $\mu$m bin size

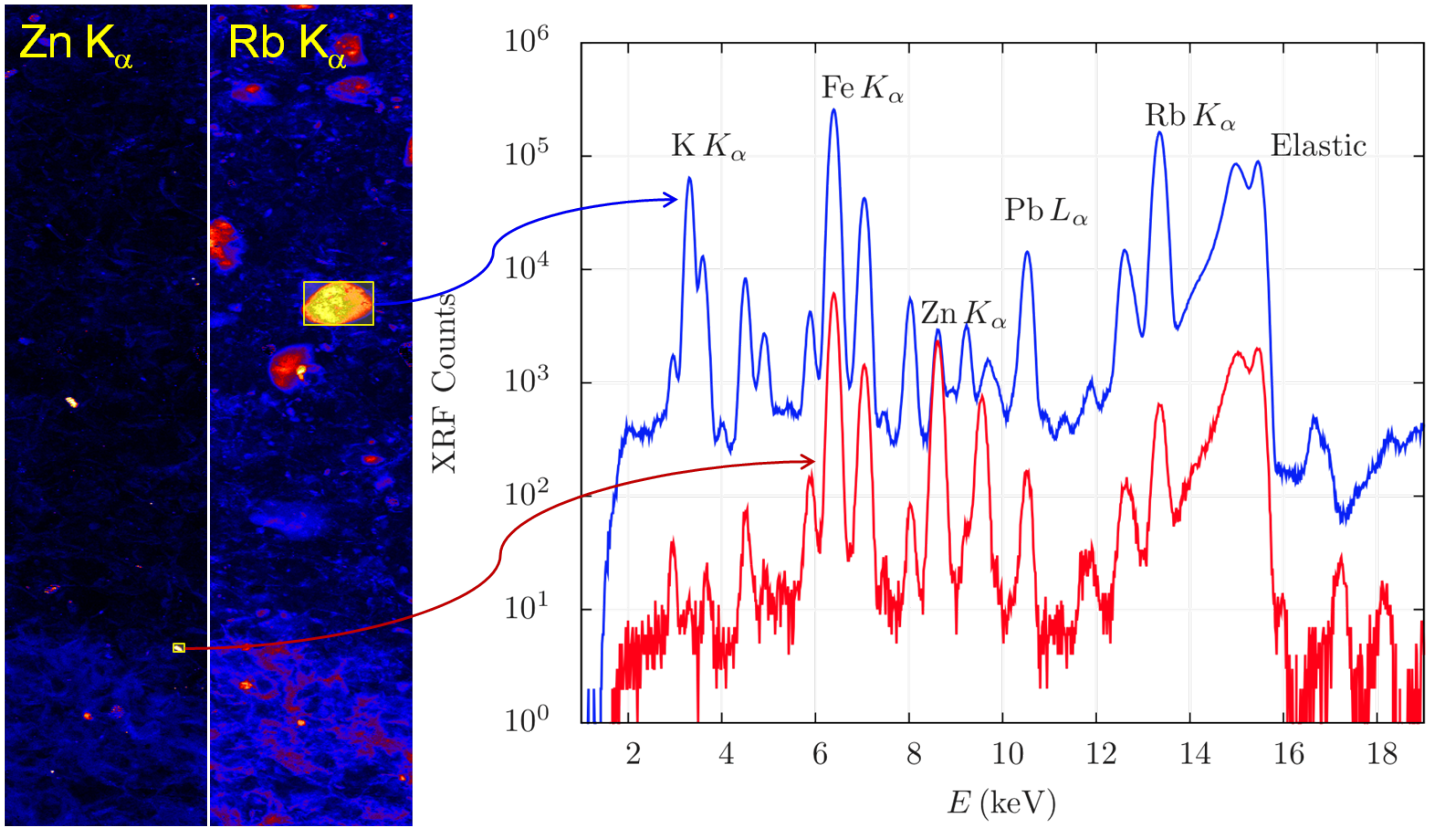
Acquire Time: 25 ms / pixel + $\sim$ 1.5 sec/row

Total Time: 95 minutes. (can now do in under 75 minutes: *2 Gb/hour*)

Improvement: At 0.5 sec / pixel, Total Time would be *22 hours*.

# Fast XRF Mapping: XRF Extraction

Full XRF spectra extracted from map is sufficient for trace element quantification, and allows discovery of elements not possible with ROI maps alone.




Rb spot : 1 min (40 × 60 pixels, 200 × 300 $\mu$m)

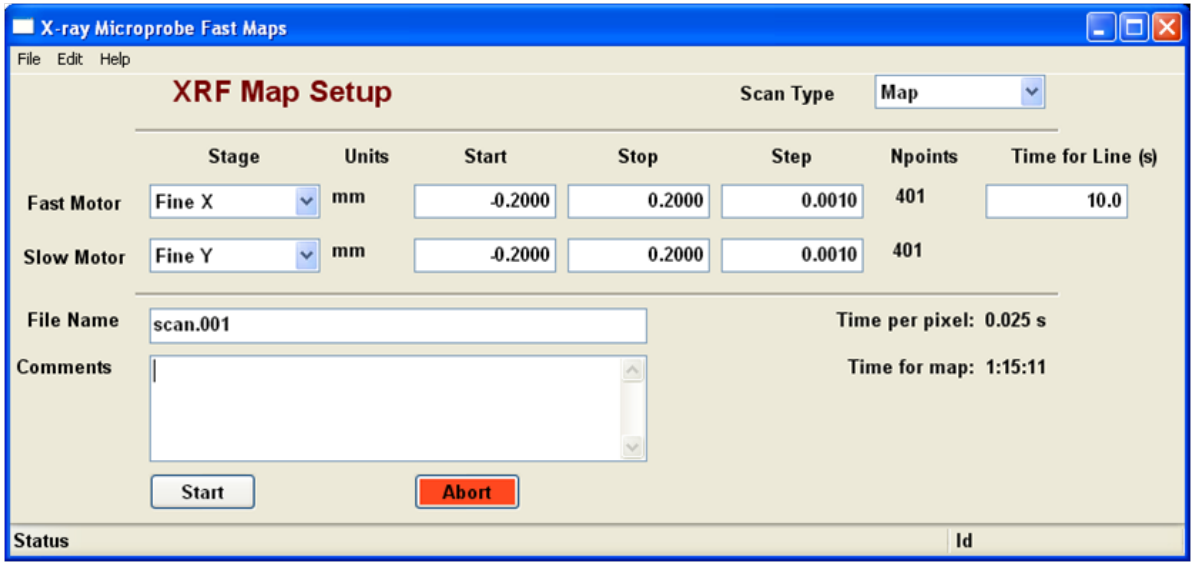
Zn spot : 1.5 sec (6 × 10 pixels, 30 × 50 $\mu$m)

# Software for Fast XRF Mapping: Client Side

## Client: User-Control

Simple GUI lets users choose:

- map size
- fast stage (limited by XPS)
- slow stage (any motor)
- pixel (pulse) size
- time per row – shows time per pixel, time for map.




---

## Server: Version 1 – hybrid solution (Epics not SQL), used in production so far.

- Client GUI writes .INI file describing scan, puts filename into Epics Record.
- Server watches Epics Record, executes when "Start" is pushed.
- Epics layer allows access to maps from IDL, Spec macros.

## Server: Version 2 – MySQL on Server, in alpha stage.

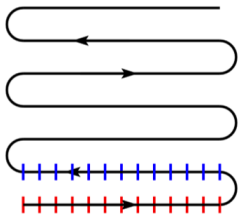
- Client GUI writes .INI configuration file describing scan into SQL database.
- Server watches SQL Table, executes when Client puts *FastMap(config_file_name)*
- Epics layer to allows IDL, Spec access (in progress).

# Software for Fast XRF Mapping: Motion Control

The Newport XPS supports *Trajectories*: complex motions of multiple stages, with external TTL pulses sent periodically (space or time). Many similar Smart Motor Controllers:

- Each with their own low-level programming interface.
- Poor support from EPICS Motor class.

The Newport XPS provides a socket-based programming library, with interfaces provided for C, C++, Tcl, Python, Matlab, Visual Basic.



For each map, *Forward* and *Backward* Trajectories (here, just 1 linear stage moving at constant velocity) are defined and uploaded to XPS Controller (via ftp!).

Triggers (TTL Pulses) are sent from XPS at requested step size.

These are connected (BNC) to detector trigger inputs.

**XPS Code to set up triggering, and launch a trajectory**

```
xps.XYLineArcPulseOutputSet(socketID, group_name, 0, range, step)
xps.EventExtendedConfigurationTriggerSet(socketID,
                    ('Always', 'FINE.XYLineArc.TrajectoryPulse'))
xps.EventExtendedConfigurationActionSet(socketID,
                                        ('GatheringOneData',))
eventID = xps.EventExtendedStart(socketID)
xps.XYLineArcExecution(socketID, group_name,
                       traj_file, speed, accel, 1)
```

# Software for Fast XRF Mapping: Detectors

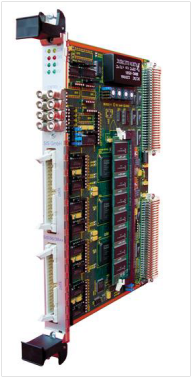
XPS Pulses go to

SIS-3820 Multi-Channel-Scaler for Ion Chamber data.

XIA xMAP or Mercury DXP for Vortex ME-4 spectra.

Mark Rivers' Epics driver (dxp-3.0) controls DXP from an Epics IOC Process on a Win32 or Linux (Mercury only) PC.

This Epics DXP Process saves 4 full spectra (2048 channels) + deadtime stats to disk every 124 pixels without stopping (and at end of row) to an uncompressed netcdf file. Data rates can reach 10 Gb / hour



---

At the end of each row (using a few separate processes and threads):

- Slow Positioner sent to next position.
- Positioner data from XPS is saved to a file (thread).
- Ion Chamber data from SIS-3820 is saved to a file (thread).
- XRF spectra flushed to netcdf file (process).

Total overhead between rows: 1 second.
Dominated by deceleration/acceleration of fast stage.

# Software for Fast XRF Mapping: Data Files

Data for each Map (or 1-D Scan) goes into a sub-directory (name based on users filename) with files written at the end of each row. Files are Plaintext (ASCII) except as noted:

- Scan.ini — scan configuration file
- ROI.dat — ROI definitions
- Environ.dat — Extra Epics PV name/values at start of scan
- Master.dat — values of "slow positioner" and scan files per row
- xps.NNNN — Position of stages from XPS for row NNNN
- struck.NNNN — Ion Chamber intensities for row NNNN
- xmap.NNNN — XRF spectra for row NNNN (netcdf)

Visualization – interim solution. Needs Work!

ROI Data is extracted to ASCII "ROI Map Data" for old GSE DataViewer.

After Row N+1 has started, a thread reads data for Row N and saves to Plaintext data file as we made with Epics SScan Record scans.

Kludgy, but allows near-real-time visualization..

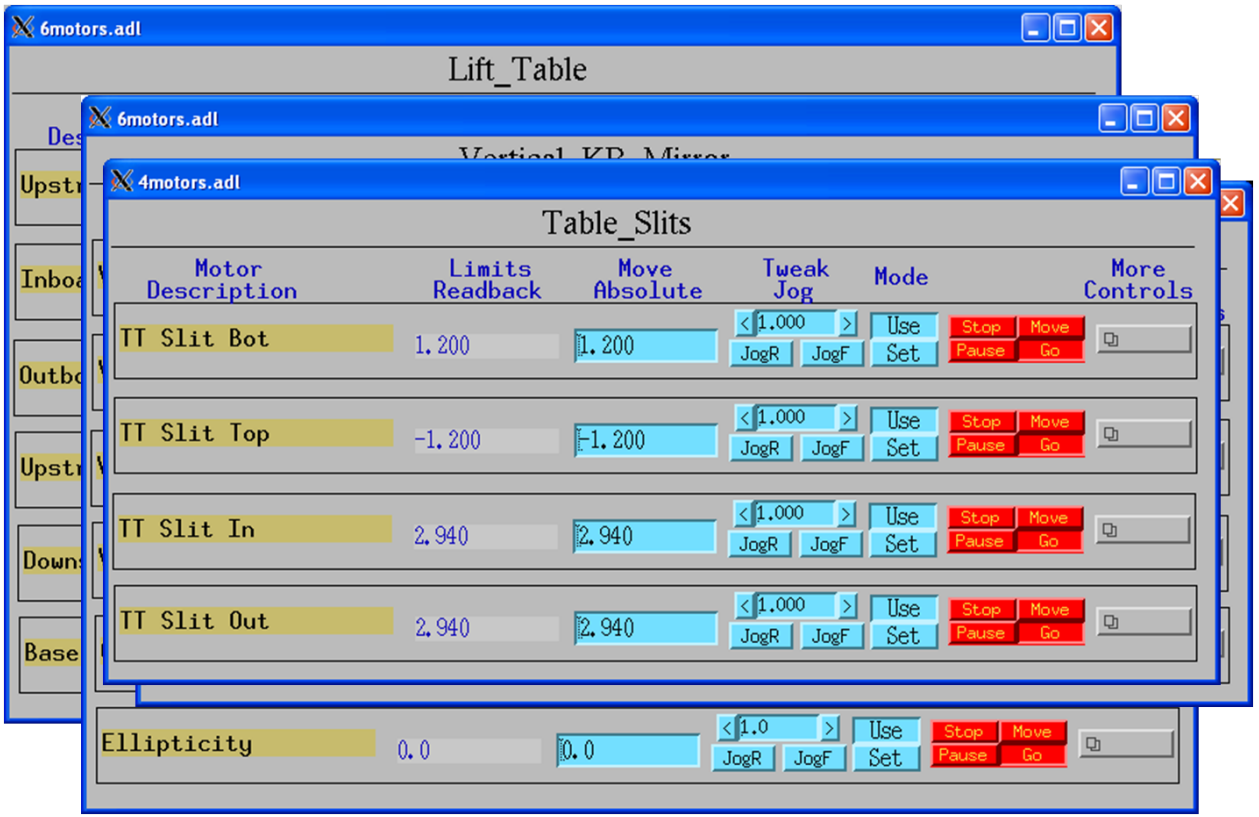
Ideally, map will be auto-converted to HDF5 for better tools ....

XRF Mapping is handled by a Server Process watching for "Execute".
Collection fully separated from Client and from Visualization.
A Map can be triggered via Epics or Command to SQL Table.

# Epics Instruments: Managing Sets of Epics PVs

EPICS has many wonderful features.

Beautiful, Easy-to-use GUIs is not one of them:



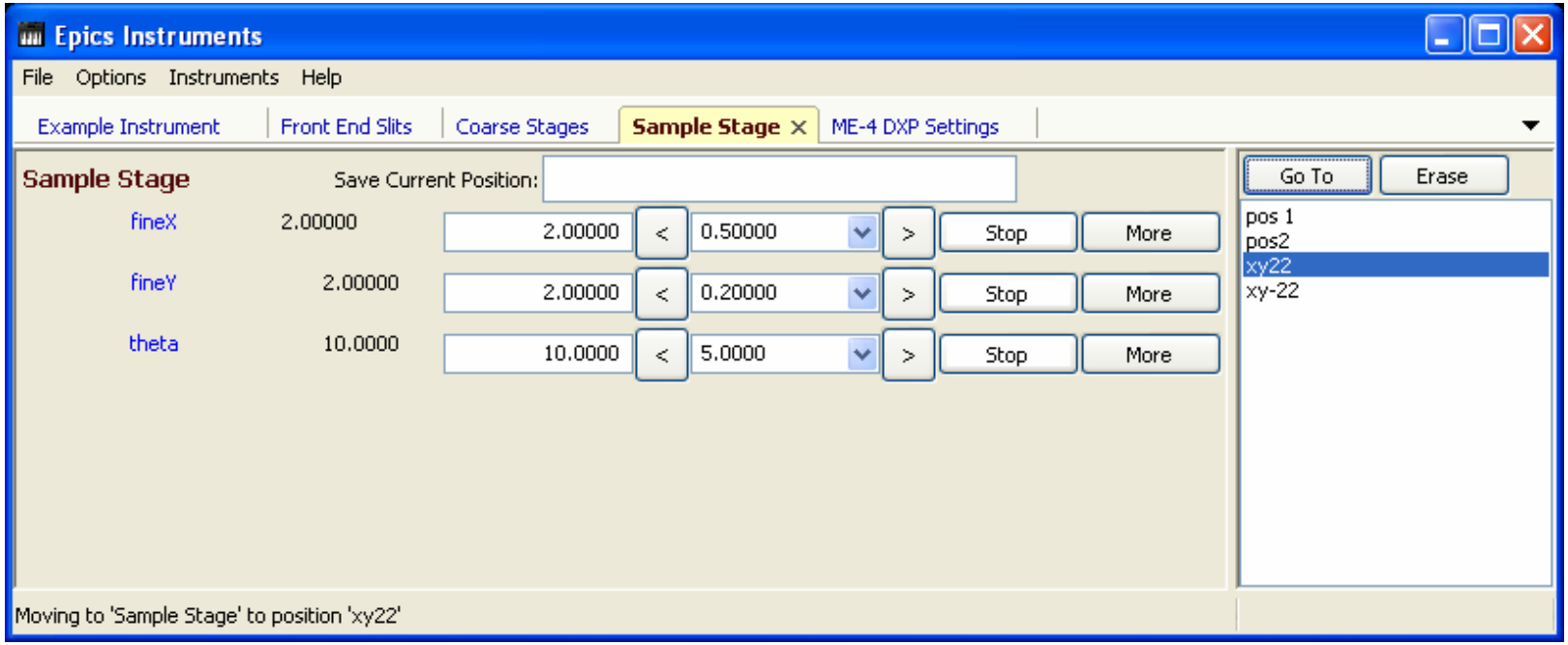

Not only do we have many ugly, hard-to-use MEDM screens for Motor and PV Displays, they cannot even save state information for a set of PVs.

# Epics Instruments: Sets of PVs with Saved Positions

Epics Instruments is a GUI application that lets any beamline scientist or user:

- Organize PVs into *Instruments* – a named collection of PVs
- Manage Many Instruments with tab interface – can hide/show instruments.
- Save Positions for any Instrument by name.
- Restore Positions for any Instrument by name.
- Smart Widgets sane cursor focus, know about PV limits, etc.

All definitions fit into a single file that can be loaded later.
Multiple Users can be using multiple instrument files at any one time.

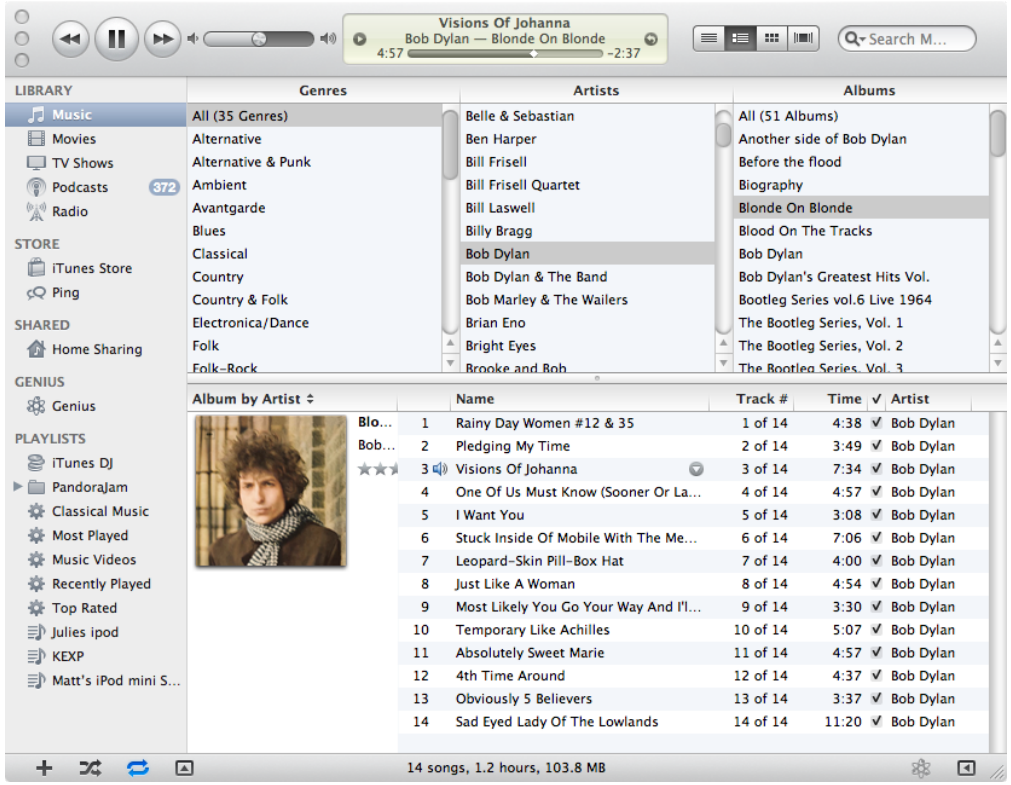



Egads! What sorcery is this?

# SQLite: a relational database with SQL access in a single file

SQLite puts a full relational database into a portable file, uses SQL for data manipulation, and is easy to use from many programming environments.

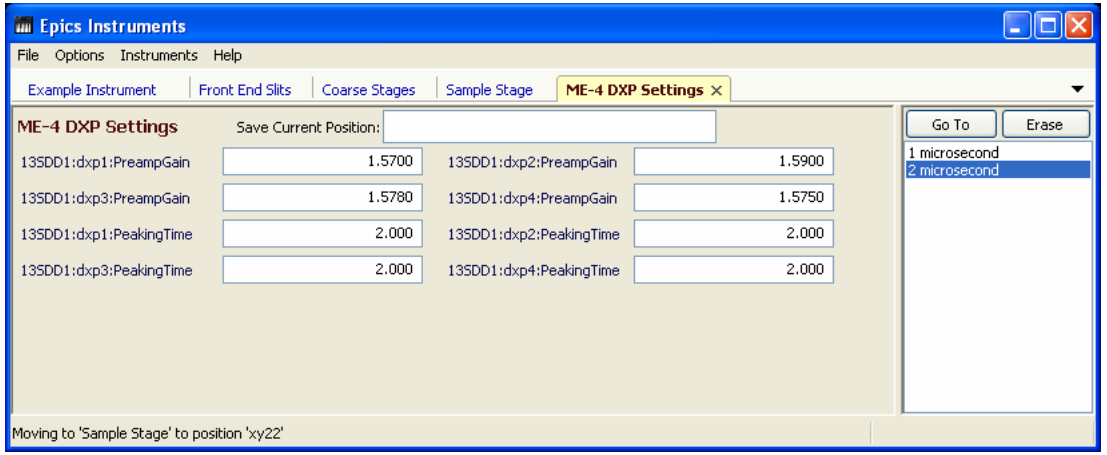
Solid reputation, widely used by applications that excel at coordinating and presenting "metadata":



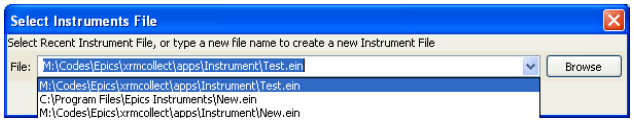
Also used by: Mozilla, Adobe, Google, Dropbox, Skype, . . . .

Completely Free.
Public Domain.

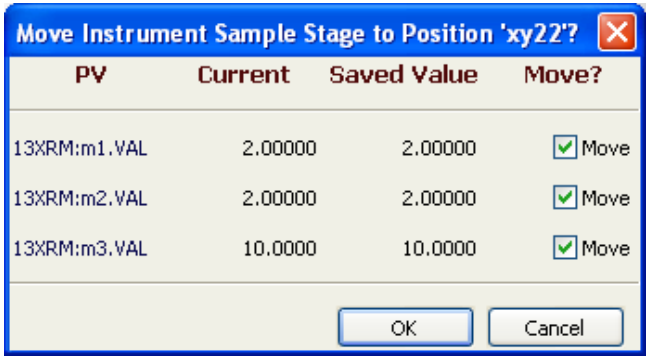
# Epics Instruments: A few more screenshots

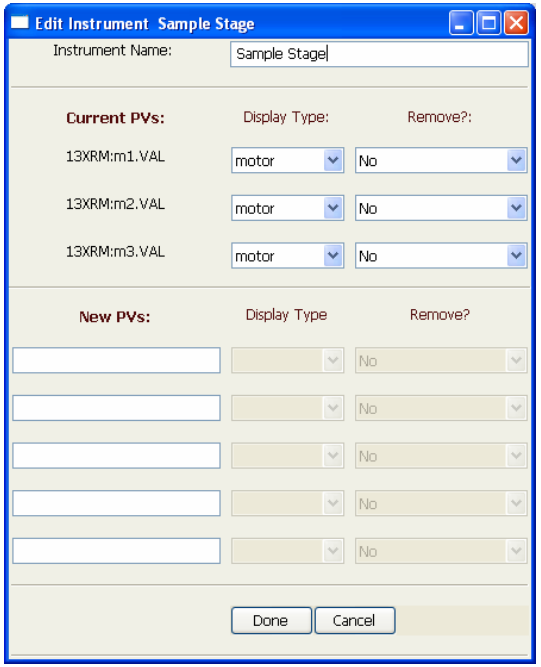



save / restore for DXP parameters (as generic PVs)




At startup, one of recent database files can be selected. On "Go To", settings can be selectively restored:




Edit Instrument Definition.

**Epics Access:**

An Epics Record can be used allowing other processes (IDL, Spec, Python) to restore positions through Epics.

At Initial Release:
Suggestions Welcome!

# Metadata and Relational Databases

Relational Databases are an excellent solution for metadata.

---

*A concrete example*: We want users to be able to:

1. make an XRF map of a sample.
2. take XAFS measurements at selected spots on the map.
3. take XRD patterns at selected spots on the map.

What are the options for doing this?

Obvious Solution store coordinates in the file headers of XAFS /XRD spectra. How do these go into those data files? How are they extracted? Can you figure if XAFS and XRD were taken at the same spot? Maybe....

Relational Database A table stores all *changes* in sample name, stage coordinates, Ring Current, etc as a function of time. This is independent of collection of Spectra or Scans and goes for the duration of the experiment.

Sample Coordinates (and other variables!) can be determined at any time via SQL.

The Relational Database solution is simpler, more robust and scalable.

# Keep External Metadata separate from Data

Don't try to save all metadata into the data files. You will fail.

**EpicsSpeak for Monitoring Metadata**

Have a separate process on the server monitor PVs holding metadata, save:

```
PVName, Timestamp, New Value
```

into a table every time a value changes. Monitor everything that might be related to the measurement. Disk space is cheap.

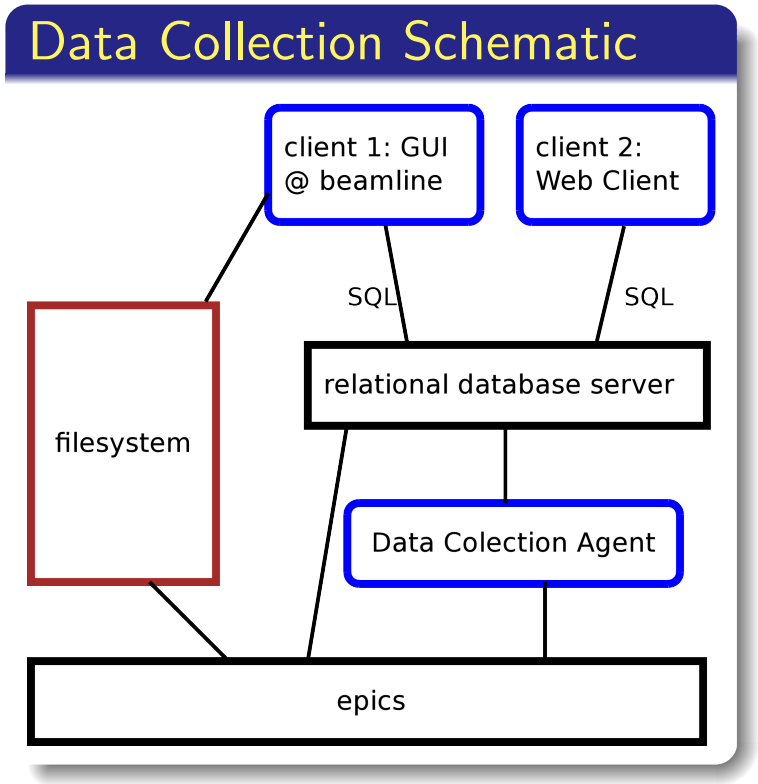
Can include changes to non-PVs (say, sample names and macro code).

Note: All changes are recorded, even during a scan.

This technology to archive changes in Epics PVs in Relational Databases already exists.

Yes, we want to bar-code samples.

# Client-Server Database as Heart of Data Collection




Relational Database Management Systems (MySQL, Postgres, . . . ) are designed for:

- multiple simultaneous clients.
- network connection.
- security.
- data integrity.

SQL is the communication protocol and language between Clients and Server.

Can implement Clients and Server in different languages, environments.

Commands sent into the Database completely separate Client and Server:

Clients send Commands via SQL to Command Table.

Server reads from Command Table what to do next.

# SQL Table for Commands in Client-Server Model

What does the Command Table look like? What do the commands look like?

**Command Table:**

| column | meaning | type |
|---|---|---|
| index | command sequence number | int |
| command | command name | string |
| arguments | command arguments/configuration | string |
| status | current status | string/enum |
| request_time | time command entered | timestamp |
| start_time | time command began | timestamp |
| end_time | time command ended | timestamp |
| datafile | name for output datafile | string |
| host_name | id of client making request | string |

Client can queue up many commands, then edit them between submission and execution.

The Command Table becomes the log file.

# SQL Commands, Macro Language?

What do the commands and arguments look like?

Commands are one of a set of Primitive commands, or defined Macros built from these.

**Example Commands:**

| command | arguments |
|---|---|
| MoveEnergy | 12000.0 |
| MoveInstrument | 'Sample Stage', 'area 1' |
| SetSampleName | 'sample1, area1' |
| XYMap | xstart=-1, xstop=1, ... |

Version 1: (current, $\alpha$)
The server decides what task to do based on command name.

Commands are defined in the server.

Version 2: (soon) Use a real scripting / command language.

- flow control, looping for maximal flexibility,
- client-defined functions held in database, interpreted (safely) by server.
- server-defined functions allow beamline scientists to configure beamline and collection particulars through scripts.
- powerful enough to include data processing.

# Larch: A Macro Language based on Python

Using an existing language (Python, Perl, . . . ) has some appeal, but some drawbacks as well.

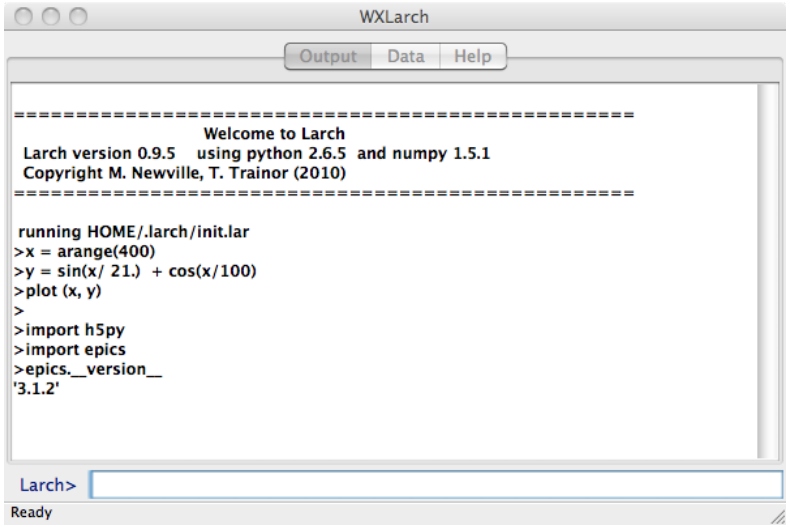
A domain-specific language (ie, that knows about scientific needs) would be better.

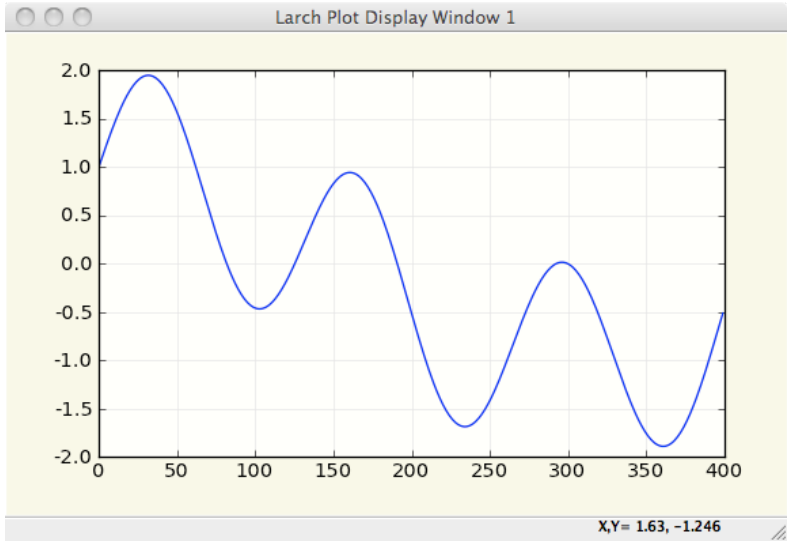
---

Warning: Python-specific, Possibly insane, Open for Suggestion.

Larch is a home-built language based on Python. This will become IFEFFIT 2.

- **Implemented in Python** Any Python code can be easily included (imported).
- **Uses Python Compiler** Code is translated to Python, compiled to Python Abstract-Syntax-Tree. Some things (say, "eval") can be simply not allowed.
- **Bendable Namespaces** AST is then executed in custom namespace, preloaded math/numerical functions. All variables are Python objects, and a Group concept (namespace for data) similar to HDF5 is used.
- **Modifiable Python syntax** block indentation replaceable with "end" .



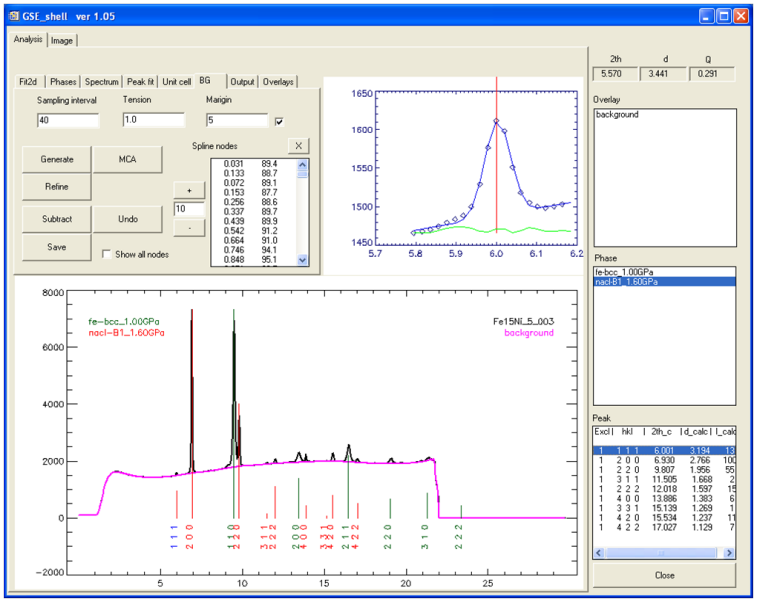



# A Macro Language to tie Analysis Codes Together!

Larch is **not** a general-purpose language.

But it could be allow a common macro language for data collection, processing, analysis.

The macro language is the framework




Przemek Dera (GSECARS) has this amazing GUI to read in XRD images and automatically subtract the background, look up and match to known structures.

This is what we want for $\mu$XRD.

It's in IDL. But a license is not $20,000.

We need to be able to include this and other such tools in our analysis streams.

If we don't invent a Framework, one might be thrust upon us.

# Conclusions

Use Client-Server Relational Databases for Data Collection.

Hold Metadata in a Relational Database.

A macro language can tie together data collection, visualization, and analysis.

There is much left to do, but all these steps are in progress.