Python Scripting Feature for NICE Software

MAX POOLE

What's the name of my project?

Integrating a Python Scripting Feature into NCNR Data Acquisition Software to allow Advanced/Customizable Experiments

What's the name of my project?

*Integrating a **Python** Scripting Feature into NCNR Data Acquisition Software to allow Advanced/Customizable Experiments*

Programming language which scientists want to write in

What's the name of my project?

*Integrating a Python **Script**ing Feature into NCNR Data Acquisition Software to allow Advanced/Customizable Experiments*

A small program

What's the name of my project?

Integrating a Python Scripting Feature into ***NCNR Data Acquisition Software*** *to allow Advanced/Customizable Experiments*

The NICE Software

What is the NICE Software?

Data acquisition:

- Software scientists use to run experiments

Universal:

- NICE is planned to work on all instruments maintained by RFO
- Currently works full time with five instruments (6th in near future)

What was my project?

This is what it is

Add the ability for scientists to write scripts in python that can be run through the NICE software. These scripts allow the scientists to create Advance/Customizable experiments

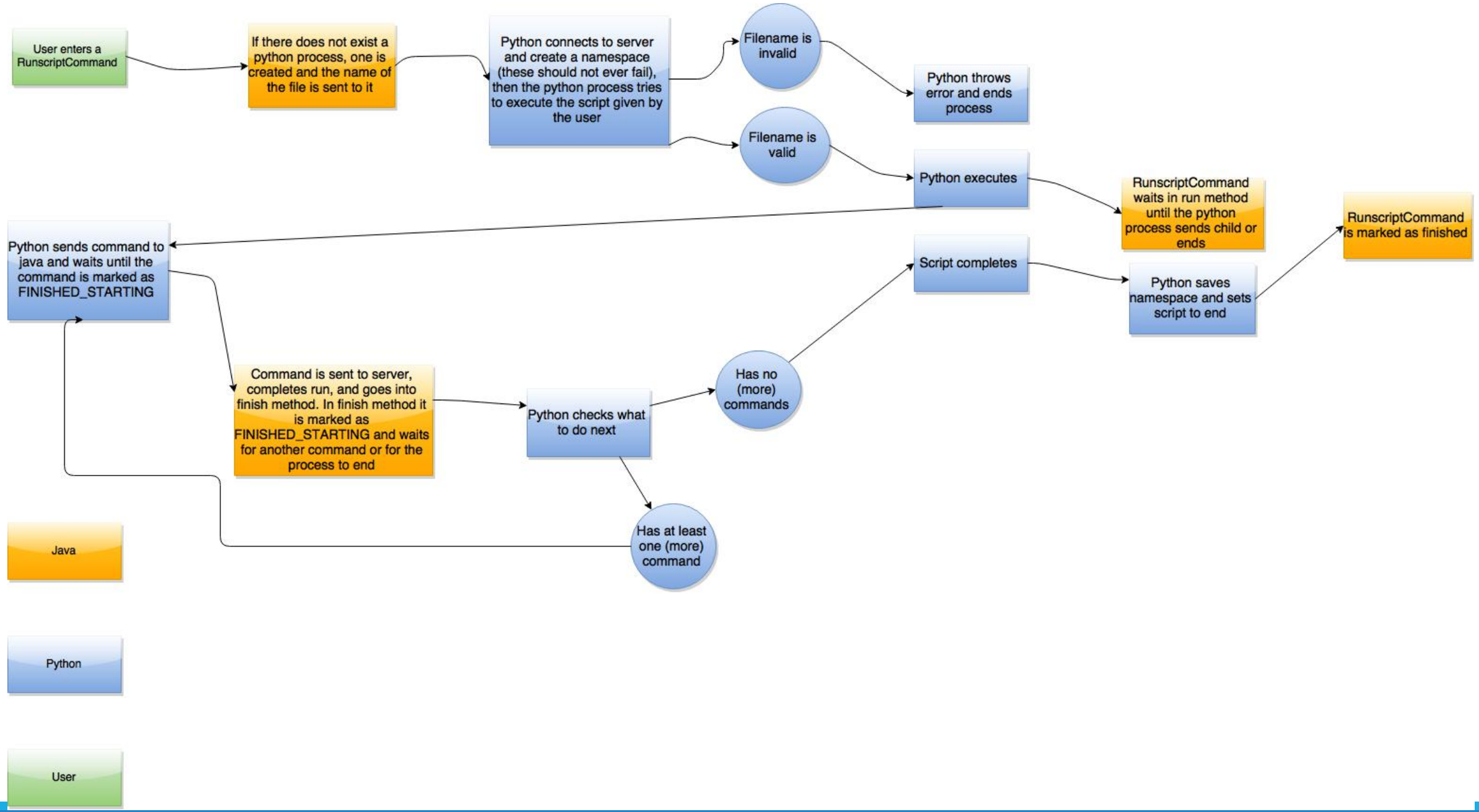
How was it done?

Python Scripting Process

- Singleton that runs the setup code
- Setup codes sets up connection with server, namespace, etc.

ScriptApi

- List of commands that the scripting feature can take advantage of


User enters a RunscriptCommand
If there does not exist a python process, one is created and the name of the file is sent to it
Python connects to server and create a namespace (these should not ever fail), then the python process tries to execute the script given by the user
Filename is invalid
Filename is valid
Python throws error and ends process
Python executes
RunscriptCommand waits in run method until the python process sends child or ends
RunscriptCommand is marked as finished
Python sends command to java and waits until the command is marked as FINISHED_STARTING
Command is sent to server, completes run, and goes into finish method. In finish method it is marked as FINISHED_STARTING and waits for another command or for the process to end
Python checks what to do next
Has no (more) commands
Has at least one (more) command
Script completes
Python saves namespace and sets script to end
Java
Python
User


User Writes Script
Python Scripting Process reads Script
Communicates with NICE
NICE Sends commands to the instruments

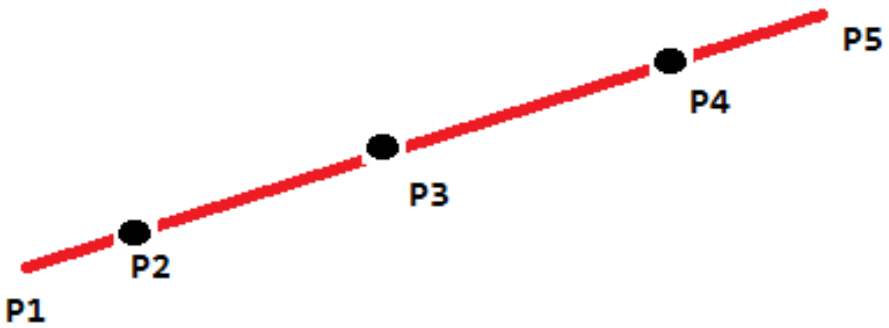
Benefits From my Project

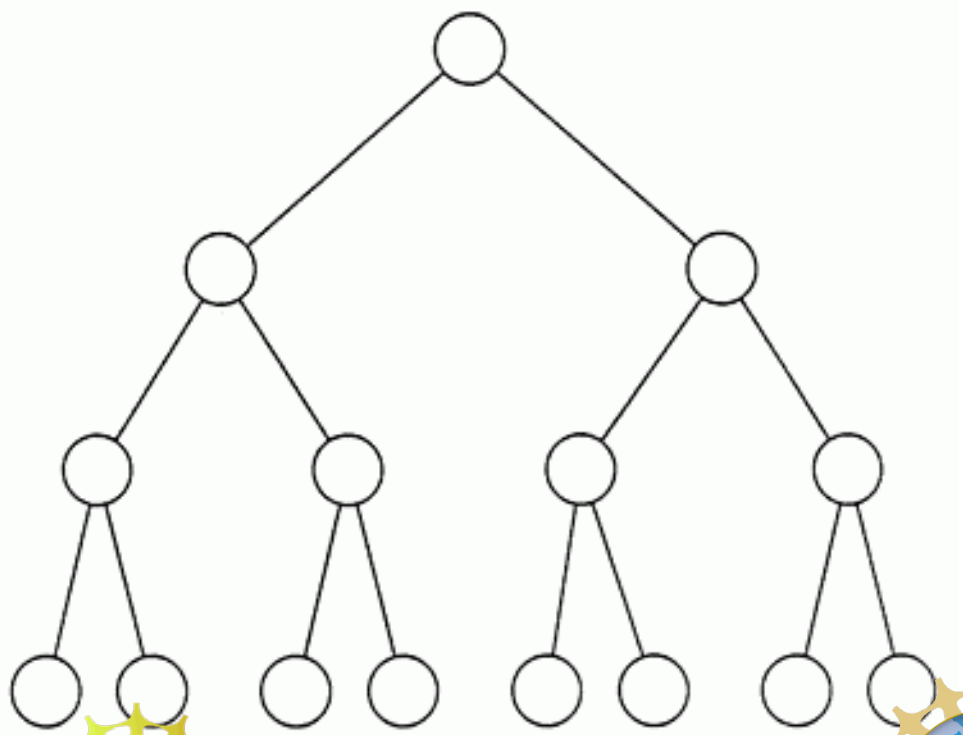
- Scientists can create flexible scripts
  - Dynamically react to conditions to what is measured
  - Tailor-made solutions
  - Can be made quickly (by one's self)
- Scientists can take advantage of NICE
  - Many already written features
  - Standardized across instruments

Dynamic Experiments

Scripts can react to data they measure and use this information to decide what to measure next

Creating Dynamic Experiments





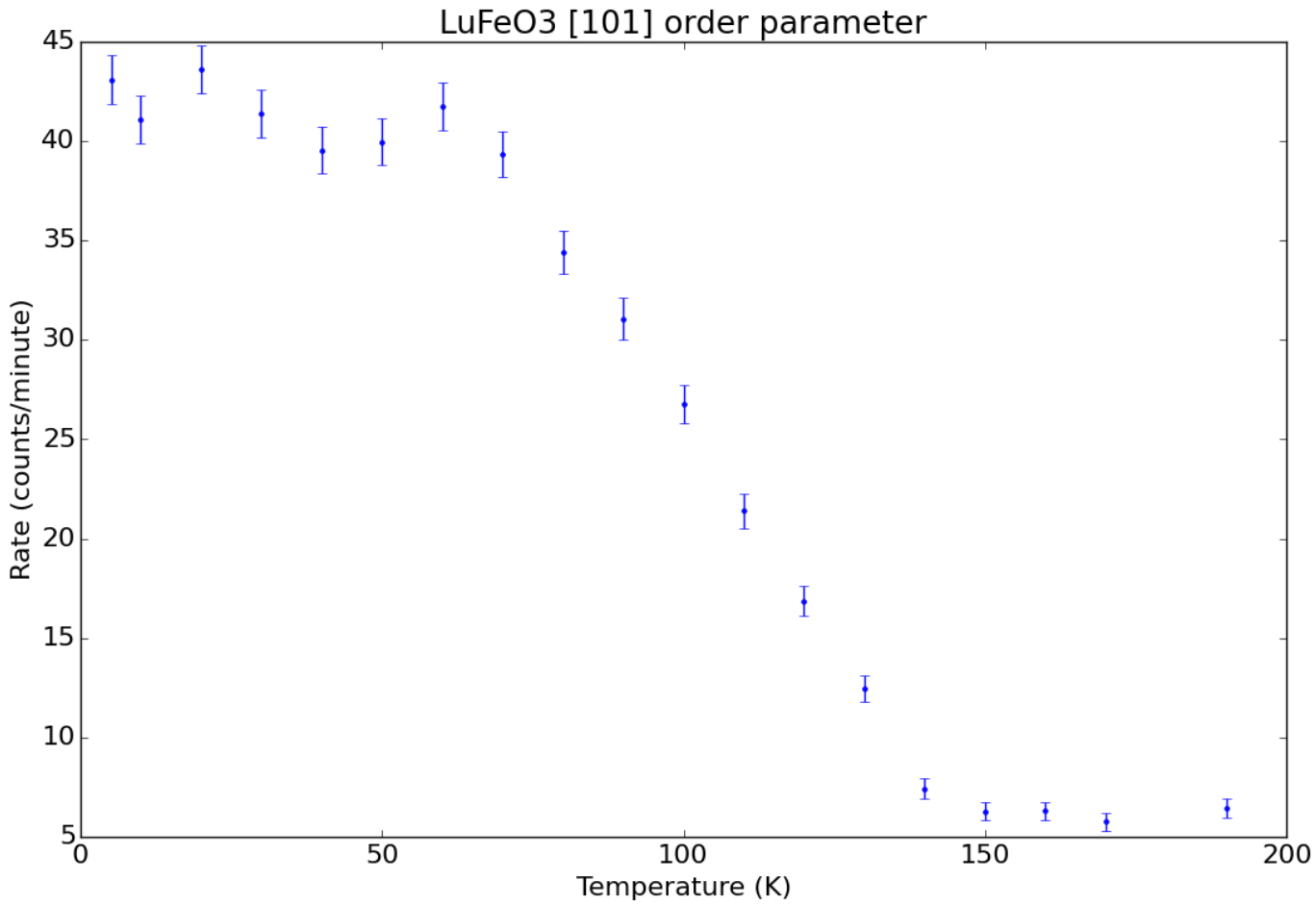
CHRNS
CENTER FOR HIGH RESOLUTION NEUTRON SCATTERING
NIST
NSF
NSF

Example phase change

With python scripting process users can have their program read data based on some condition

Phase change: Read information rapidly only when the phase is changing

Closer look


LuFeO3 [101] order parameter
Rate (counts/minute)
Temperature (K)
45
40
35
30
25
20
15
10
5
0
50
100
150
200

Tailor-made scripts

Scripts can be tailor-made for a user's specific problem

Scripts are created by the users themselves

You can do it all without asking the programmers

Make scripts quickly

- Useful for trying out new ideas
- Creating solutions for problems specific to only what you are doing right now

Integrated into the new and exciting software

NICE!

Advantages of using NICE

Scientists can take advantage of many features already written by the programmers

- Talking to motors
- Writing data in known formats
- And so much more!

```
    MoveCommand cmd = new MoveCommand();
    cmd.args = convertMapToList(positions);
    MoveCommand.validateNodeIds(cmd, positions.keySet(), cmd.status);
    return cmd;
}

@Override
public void validate()
{
    if ((args == null) || args.isEmpty())
    {
        // could happen if command is instantiated directly, not parsed with console parser
        String message = "Command has no arguments";
        CommandUtils.registerValidationError(this, EventLevel.WARNING, message);
    }

    if ((args.size() % 2) != 0)
    {
        String message = "Uneven number of arguments.";
        CommandUtils.registerValidationError(this, EventLevel.WARNING, message);
    }

    final Collection<String> nodeIds = new ArrayList<>();
    for (int i = 0; i < args.size(); ++i)
    {
        if ((i % 2) == 0)
        {
            nodeIds.add(args.get(i).toString());
        }
    }
    MoveCommand.validateNodeIds(this, nodeIds, status);
}

private ParsedControlCommand parseArgumentsIntoCommand(Command sourceCommand, DeviceModelInterface deviceModel)
        throws MissingNodeException, StatusValueNotGoodException
{
    final ReprMap<String, String> nodeToValueMap = new ReprMap<>();
    Iterator<Object> iterator = args.iterator();
    while (iterator.hasNext())
    {
        final String key = (String) iterator.next();
        String value = iterator.next().toString();
        // If a relative move is resuming, we should convert it to absolute move instead of
        // re-submitting the relative move.
        // This is done so that we can avoid following bug:
        // Motor m is at 10.
        // move m -r 5 (should set desired value of m to 15). while it is moving there, if a
        // pause is issued at 12 and later the user resumes it, the new desired value of m would
        // be 12 + 5 = 17.
        if (relative && isResuming())
        {
            NodeHandle<?> node = deviceModel.getNode(key);
            Object internalValue = node.getDesiredValueStatusOutput().getGoodValue();
            value = UnitConversion.convertInternalToUserUnit(internalValue, node).toString();
        }
        nodeToValueMap.put(key, value);
    }
    // Now that we've converted the relative move to absolute move for resuming relative moves,
```

```
1 script_api.move(["slitTrans1", "50"])
```

What can it do?

- Findpeak
- Count
- Read from instruments
- Move motors
- Start trajectories
- Monitor temperature

And More

- Run sequences
- Get findpeak data
- Save namespaces
- Email members of experiment
- Start and Stop writers

Expandable

Written as a part of NICE

- API can easily have new features added to it as NICE itself expands

A part of a whole

The Python Scripting feature can be used as a runscript command inside and alongside other commands

NICE Client

File Window Experiment Help

Magik (Simulated) **Experiment - Blank initial experiment** (nonims0) Submit Bug << Sim Rate >> Sim Step: >| BUSY Pause Stop

Console Event Trajectory progress

2015-08-03 14:33:29 > **IMPORTANT:** Now starting experiment "Blank initial experiment" (nonims0)
2015-08-03 14:33:34 > **IMPORTANT:** Server Started Successfully
2015-08-03 14:34:07 > add "fakeTemperature" -s true *(not typed at console)*
2015-08-03 14:34:07 > Queued. (id: e29a40be-419e-4694-8a41-f1c11464a9b4, position: 0)
2015-08-03 14:34:07 > Added simulated device "temp".
2015-08-03 14:34:10 > runScript tempmonitorexample.py
2015-08-03 14:34:10 > Queued. (id: 661a9aec-016c-4ec1-849c-2777af5d4ac9, position: 1)
2015-08-03 14:34:11 > Starting scan id: "Generic script name1".
2015-08-03 14:34:14 > **Peak fit results**

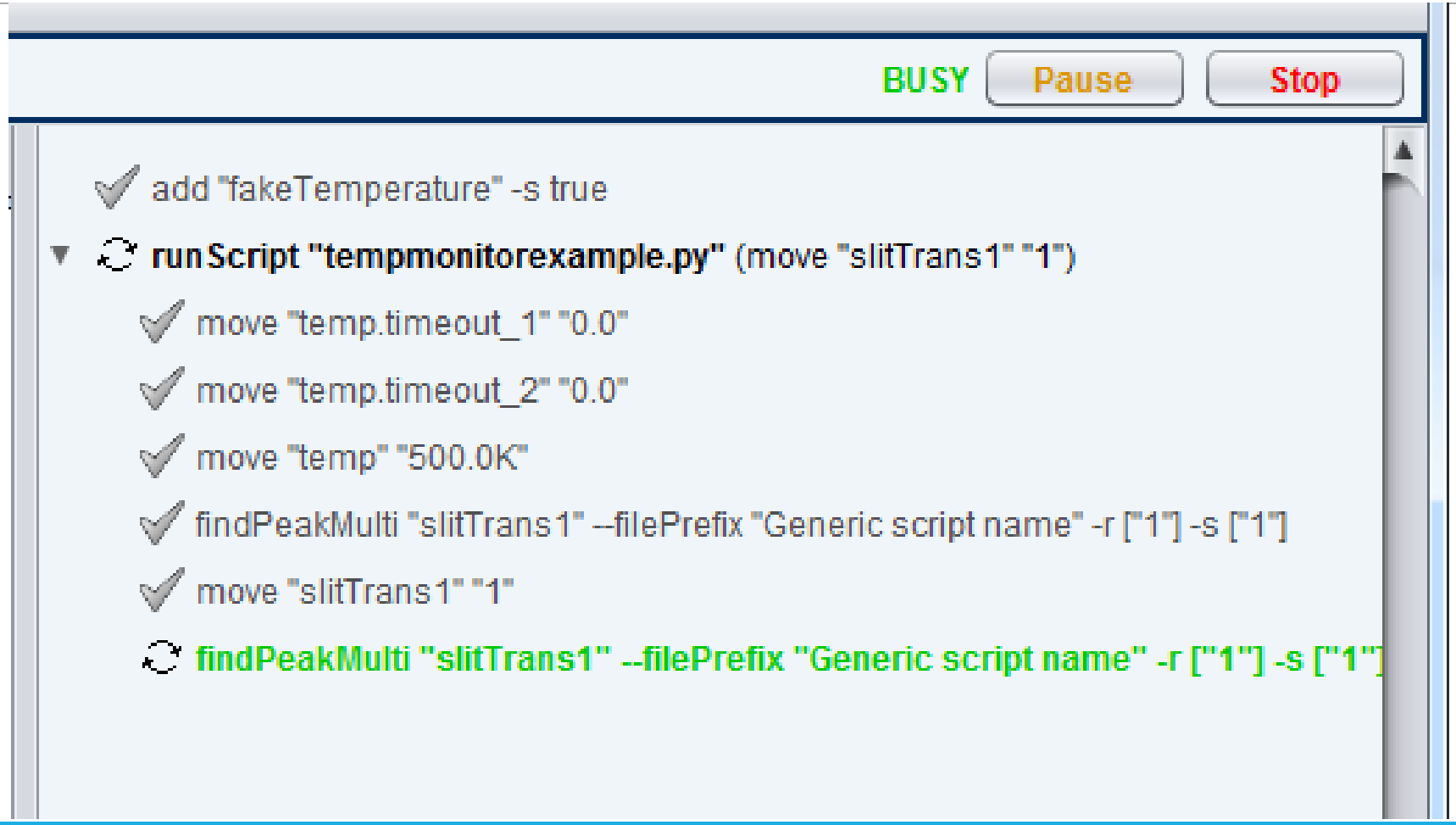
Scanned node:	slitTrans1.softPosition
Domain node:	slitTrans1.softPosition
Data of interest node:	counter.liveMonitor
Fit type:	GaussianPlusBackground
dev:	172.268 mm
background:	96.0376
center:	-0.499723 mm
height:	-0.0375917
FWHM:	405.659 mm
Reduced chi squared:	NaN

2015-08-03 14:34:14 > Moving slitTrans1.softPosition back to 0.0000 mm.
2015-08-03 14:34:14 > Finished writing file "C:\workspace\nice\server_data\experiments\nonims0\data\Generic script name_001.cgd" for trajectory id 1 experiment "nonims0"
2015-08-03 14:34:14 > Finished writing file "C:\workspace\nice\server_data\experiments\nonims0\data\Generic script name1.nxs.cgd.zip" for trajectory id 1 experiment "nonims0"
2015-08-03 14:34:15 > User ended findpeak command early
2015-08-03 14:34:17 > Starting scan id: "Generic script name2".

add "fakeTemperature" -s true
runScript "tempmonitorexample.py" (move "slitTrans1" "1")
move "temp.timeout_1" "0.0"
move "temp.timeout_2" "0.0"
move "temp" "500.0K"
findPeakMulti "slitTrans1" --filePrefix "Generic script name" -r ["1"] -s ["1"]
move "slitTrans1" "1"
findPeakMulti "slitTrans1" --filePrefix "Generic script name" -r ["1"] -s ["1"]

Expand All Collapse All

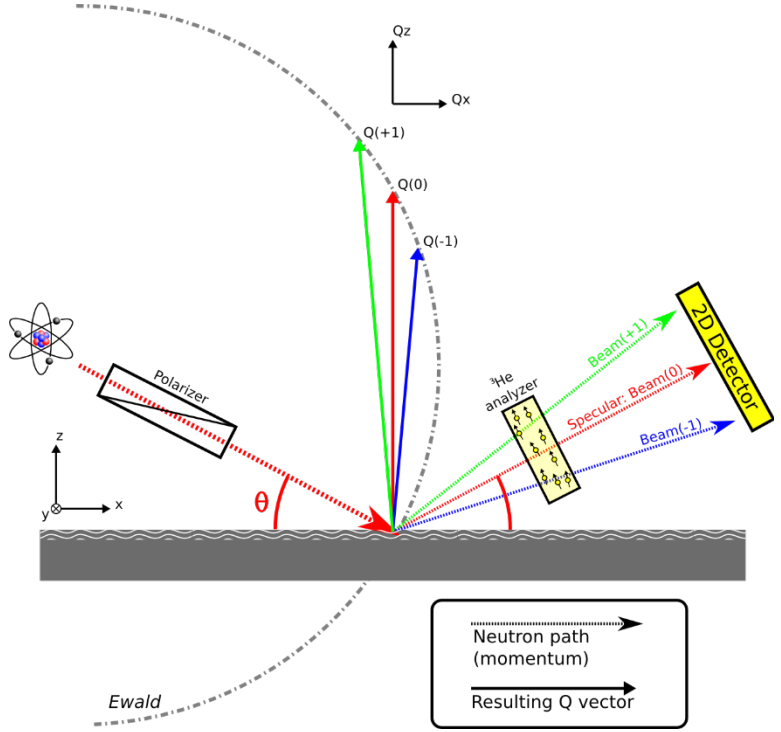
Calculate time left

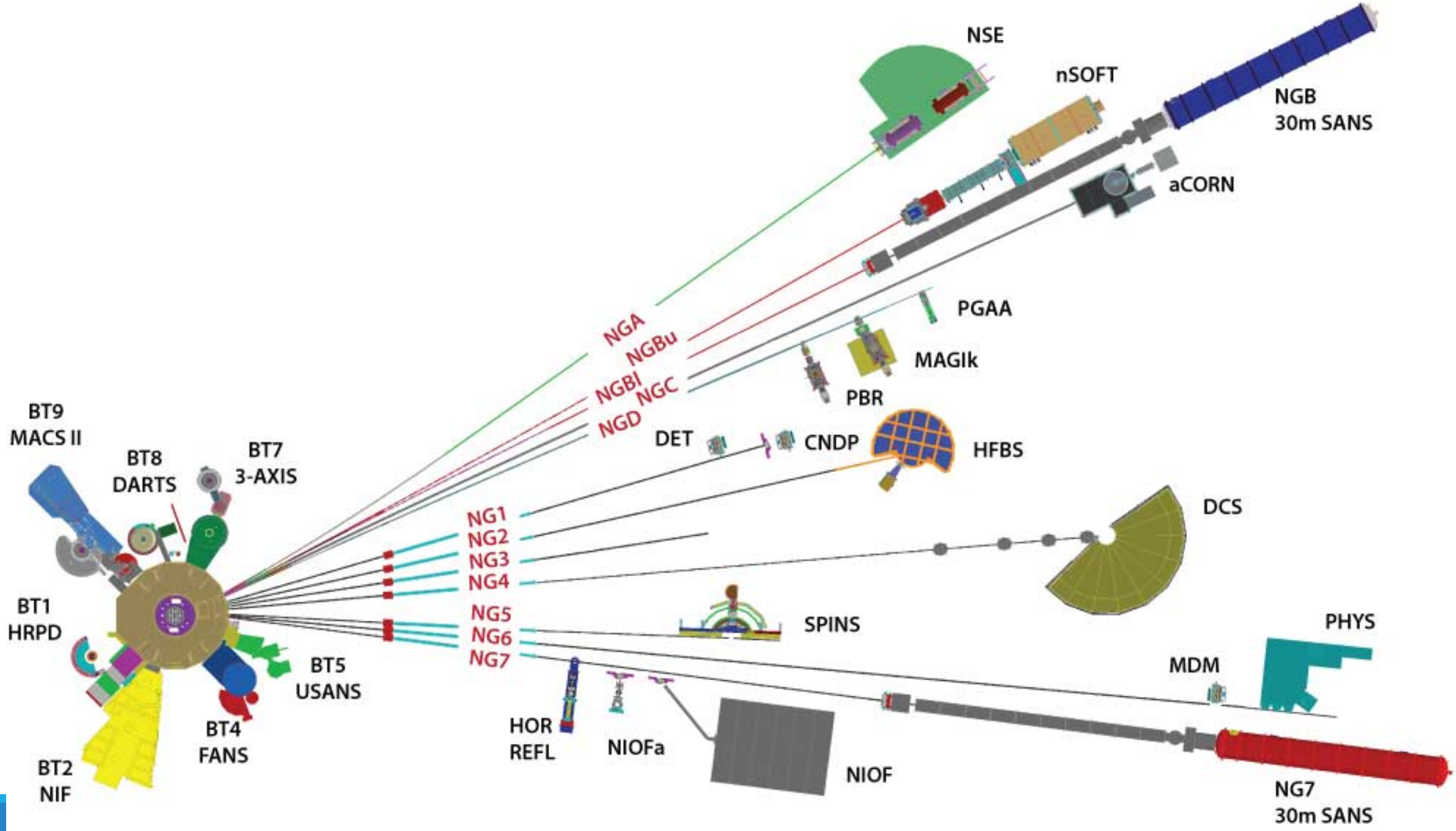

BUSY
Pause
Stop
add "fakeTemperature" -s true
runScript "tempmonitorexample.py" (move "slitTrans1" "1")
move "temp.timeout_1" "0.0"
move "temp.timeout_2" "0.0"
move "temp" "500.0K"
findPeakMulti "slitTrans1" --filePrefix "Generic script name" -r ["1"] -s ["1"]
move "slitTrans1" "1"
findPeakMulti "slitTrans1" --filePrefix "Generic script name" -r ["1"] -s ["1"]

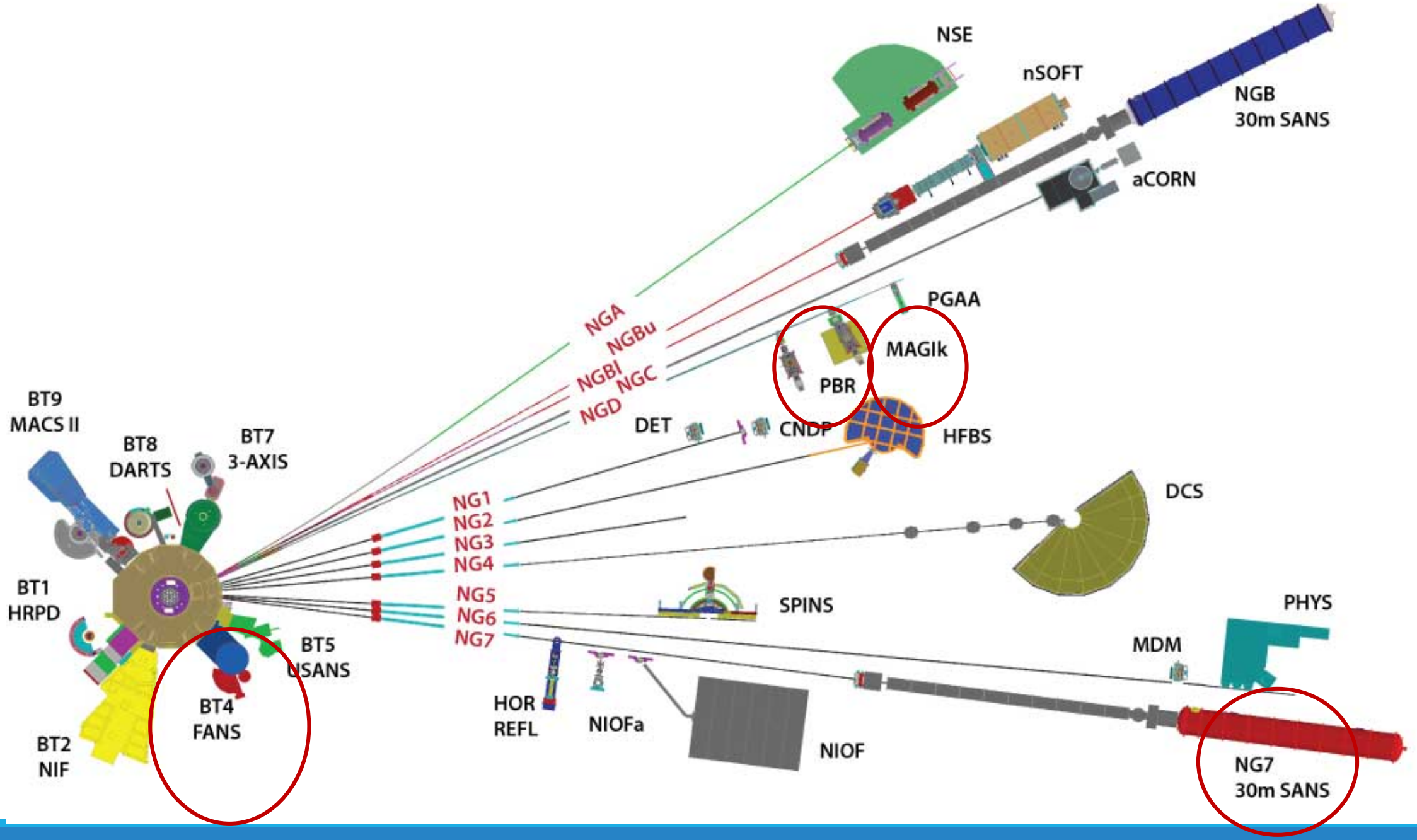
Advantages of using NICE

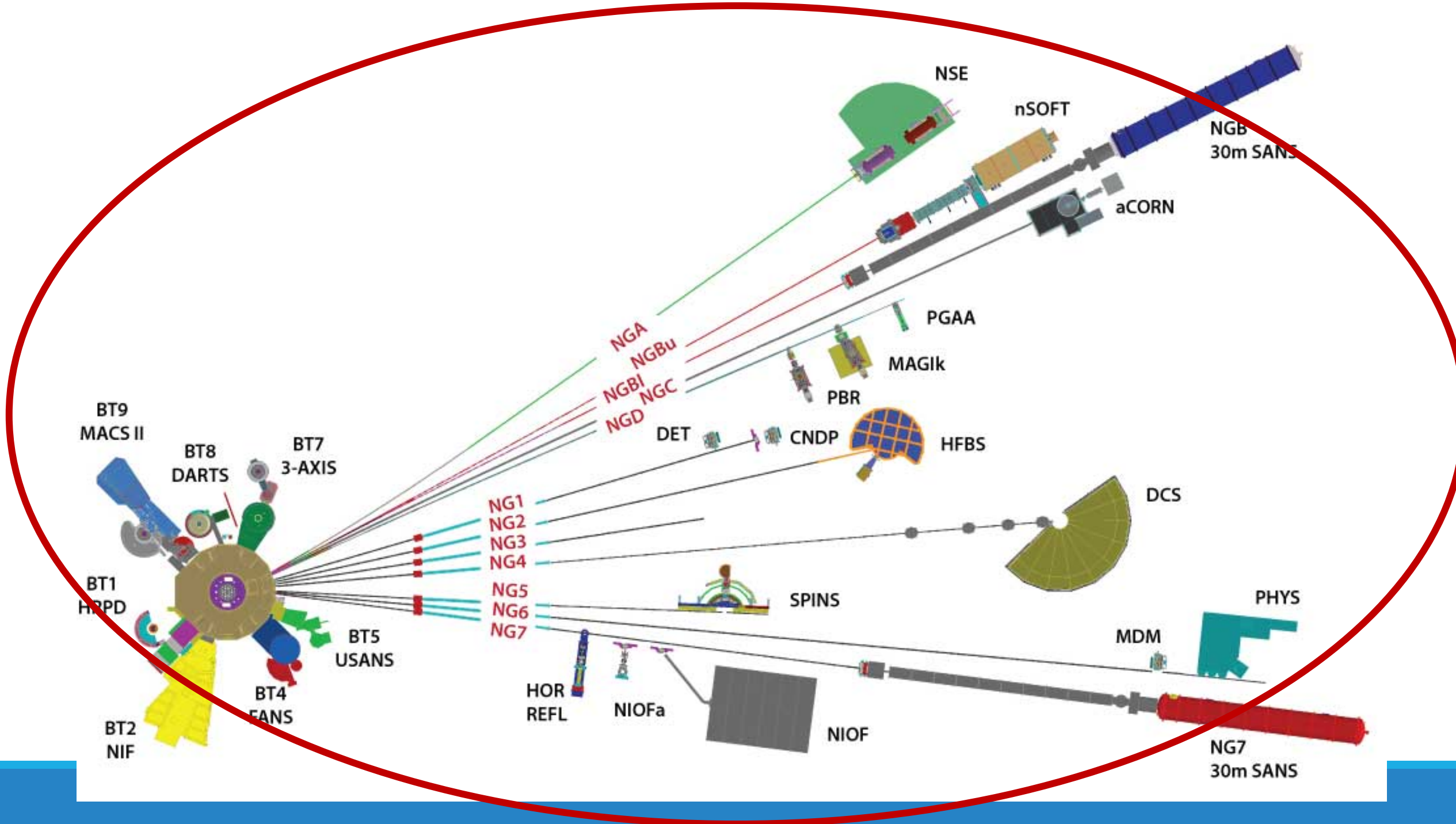
Instrument usage is standardized

- One piece of software can be used on various instruments




NSE
nSOFT
NGB
30m SANS
aCORN
PGAA
MAGIk
PBR
NGA
NGBu
NGBl
NGC
NGD
DET
CNDP
HFBS
DCS
BT9
MACS II
BT8
DARTS
BT7
3-AXIS
NG1
NG2
NG3
NG4
NG5
NG6
NG7
BT1
HRPD
SPINS
PHYS
MDM
BT5
USANS
BT4
FANS
BT2
NIF
HOR
REFL
NIOFa
NIOF
NG7
30m SANS


NSE
nSOFT
NGB
30m SANS
aCORN
PGAA
MAGIk
PBR
NGA
NGBu
NGBl
NGC
NGD
DET
CNDP
HFBS
DCS
BT9
MACS II
BT8
DARTS
BT7
3-AXIS
NG1
NG2
NG3
NG4
NG5
NG6
NG7
BT1
HRPD
BT5
USANS
SPINS
PHYS
MDM
BT4
FANS
HOR
REFL
NIOFa
NIOF
BT2
NIF
NG7
30m SANS


NSE
nSOFT
NGB
30m SANS
aCORN
NGA
NGBu
NGBl
NGC
NGD
PGAA
MAGIk
PBR
DET
CNDP
HFBS
BT9
MACS II
BT8
DARTS
BT7
3-AXIS
NG1
NG2
NG3
NG4
DCS
BT1
HRPD
NG5
NG6
NG7
SPINS
PHYS
MDM
BT5
USANS
BT4
FANS
HOR
REFL
NIOFa
NIOF
BT2
NIF
NG7
30m SANS

Conclusion

My project allows scientists to create simple programs

- Can be whatever they want them to be
- Can be flexible, can respond to the instrument and control the instruments

End and everything

Mentor: Stephen Pheiffer

Team Member: Natasha Shmunis

Paul Kienzle

Questions?


UMBC


NIST Center for
Neutron Research


NSF
CHRNS
CENTER FOR HIGH RESOLUTION NEUTRON SCATTERING
NIST


NSF