MLZ
Heinz Maier-Leibnitz Zentrum

NICOS
Networked Instrument COntrol System

Principle and practice of instrument control in neutron facilities - the NICOS system

Jens Krüger

MLZ is a cooperation between:
JÜLICH Forschungszentrum
Helmholtz-Zentrum hereon
Technical University of Munich

Challenges for Instrument Control

- Completely different types of instruments

Challenges for Instrument Control

- Completely different types of instruments
  - Triple axes

Challenges for Instrument Control

- Completely different types of instruments
  - Triple axes
  - Diffractometer

Challenges for Instrument Control

- Completely different types of instruments
  - Triple axes
  - Diffractometer
  - Reflectometer

Challenges for Instrument Control

- Completely different types of instruments
  - Triple axes
  - Diffractometer
  - Reflectometer
  - Small angle scattering

Challenges for Instrument Control

- Completely different types of instruments
  - Triple axes
  - Diffractometer
  - Reflectometer
  - Small angle scattering
  - Tomography

Challenges for Instrument Control

- Completely different types of instruments
  - Triple axes
  - Diffractometer
  - Reflectometer
  - Small angle scattering
  - Tomography
  - …

Challenges for Instrument Control

- Completely different types of instruments
  - Triple axes
  - Diffractometer
  - Reflectometer
  - Small angle scattering
  - Tomography
  - …
- Every instrument is unique!

Challenges for Instrument Control

- Completely different types of instruments
  - Triple axes
  - Diffractometer
  - Reflectometer
  - Small angle scattering
  - Tomography
  - …
- Every instrument is unique!
- Unified (or standardized) instrument control ?

Challenges for Instrument Control

- Completely different types of instruments
  - Triple axes
  - Diffractometer
  - Reflectometer
  - Small angle scattering
  - Tomography
  - …
- Every instrument is unique!
- Unified (or standardized) instrument control ?

  Typical answer: **No way**!

Challenges for Instrument Control

- Completely different types of instruments
  - Triple axes
  - Diffractometer
  - Reflectometer
  - Small angle scattering
  - Tomography
  - …
- Every instrument is unique!
- Unified (or standardized) instrument control ?

  Typical answer: **No way**!

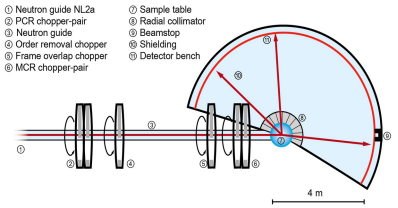
  My answer: Possible

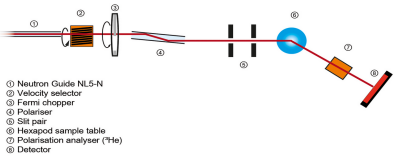
Challenges for Instrument Control

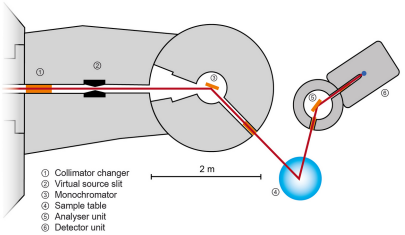
- Completely different types of instruments

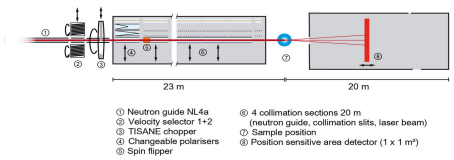
Challenges for Instrument Control

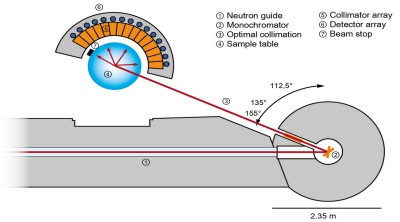
- Completely different types of instruments











Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
  - Move/Drive a motor/axis

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
  - Move/Drive a motor/axis
  - Move/Switch a switch

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
  - Move/Drive a motor/axis
  - Move/Switch a switch
  - Move/Drive a voltage/current

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
  - Move/Drive a motor/axis
  - Move/Switch a switch
  - Move/Drive a voltage/current
  - Move/Drive a temperature/field (magnetic, electric)

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
  - Move/Drive a motor/axis
  - Move/Switch a switch
  - Move/Drive a voltage/current
  - Move/Drive a temperature/field (magnetic, electric)
  - Preset a monitor, counter, detector, ...

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
  - Move/Drive a motor/axis
  - Move/Switch a switch
  - Move/Drive a voltage/current
  - Move/Drive a temperature/field (magnetic, electric)
  - Preset a monitor, counter, detector, ...
  - Start/Stop a monitor, counter, detector, ...

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
  - Move/Drive a motor/axis
  - Move/Switch a switch
  - Move/Drive a voltage/current
  - Move/Drive a temperature/field (magnetic, electric)
  - Preset a monitor, counter, detector, ...
  - Start/Stop a monitor, counter, detector, ...
  - Wait for finishing of a movement and/or counting

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
  - Move/Drive a motor/axis
  - Move/Switch a switch
  - Move/Drive a voltage/current
  - Move/Drive a temperature/field (magnetic, electric)
  - Preset a monitor, counter, detector, ...
  - Start/Stop a monitor, counter, detector, ...
  - Wait for finishing of a movement and/or counting
  - Store signal(s), additional (meta) data (positions, fields, temperature, …), logs

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
  - Move/Drive a motor/axis
  - Move/Switch a switch
  - Move/Drive a voltage/current
  - Move/Drive a temperature/field (magnetic, electric)
  - Preset a monitor, counter, detector, ...
  - Start/Stop a monitor, counter, detector, ...
  - Wait for finishing of a movement and/or counting
  - Store signal(s), additional (meta) data (positions, fields, temperature, …), logs

→ Group steps in sequences or more complex devices

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
  - Move/Drive a motor/axis
  - Move/Switch a switch
  - Move/Drive a voltage/current
  - Move/Drive a temperature/field (magnetic, electric)
  - Preset a monitor, counter, detector, ...
  - Start/Stop a monitor, counter, detector, ...
  - Wait for finishing of a movement and/or counting
  - Store signal(s), additional (meta) data (positions, fields, temperature, …), logs

→ Group steps in sequences or more complex devices

Higher level actions: Counts, Scans, …

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
- Huge number of devices to handle

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
- Huge number of devices to handle
  - Motors: dozens of axes are involved and some have to move „synchronised“

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
- Huge number of devices to handle
  - Motors: dozens of axes are involved and some have to move „synchronised“
  - Detectors for neutrons and other signals

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
- Huge number of devices to handle
  - Motors: dozens of axes are involved and some have to move „synchronised“
  - Detectors for neutrons and other signals
  - Many additional (complex) devices:

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
- Huge number of devices to handle
  - Motors: dozens of axes are involved and some have to move „synchronised“
  - Detectors for neutrons and other signals
  - Many additional (complex) devices:

    cryostats, magnets, furnaces, frequency generators, sample changer, …

Challenges for Instrument Control

- Completely different types of instruments
- Many (in some cases configurable) steps for a single measurement
- Huge number of devices to handle
  - Motors: dozens of axes are involved and some have to move „synchronised“
  - Detectors for neutrons and other signals
  - Many additional (complex) devices:

    cryostats, magnets, furnaces, frequency generators, sample changer, …
- Operation with (external) users!

Requirements to Instrument control

Requirements to Instrument control

- Highly automated
  - Goal: achieve a smooth operation

Requirements to Instrument control

- Highly automated
  - Goal: achieve a smooth operation
- All hardware components have to be monitored
  - Goal: reproduce the state

Requirements to Instrument control

- Highly automated
  - Goal: achieve a smooth operation
- All hardware components have to be monitored
  - Goal: reproduce the state
- External users must be able to operate the system within a short time
  - Goal: Effective use of measurement time

Requirements to Instrument control

- Highly automated
  - Goal: achieve a smooth operation
- All hardware components have to be monitored
  - Goal: reproduce the state
- External users must be able to operate the system within a short time
  - Goal: Effective use of measurement time
- Integrate new devices very quickly
  - Goal: Short reaction time to new requests

Requirements to Instrument control

- Highly automated
  - Goal: achieve a smooth operation
- All hardware components have to be monitored
  - Goal: reproduce the state
- External users must be able to operate the system within a short time
  - Goal: Effective use of measurement time
- Integrate new devices very quickly
  - Goal: Short reaction time to new requests
- Common, but flexible
  - Goal: Users need an interface with ability to program

Instrument Control solutions

- Different solutions on market at the Neutron facilities

Instrument Control solutions

- Different solutions on market at the Neutron facilities
- One of them is

NICOS
Networked Instrument COntrol System

NICOS solutions

- Python is implementation and user language

NICOS solutions

- Python is implementation and user language
- NICOS knows
  - Devices (implemented as Python objects)
  - Commands (for routine tasks with full freedom of use)

NICOS solutions

- Python is implementation and user language
- NICOS knows
  - Devices (implemented as Python objects)
  - Commands (for routine tasks with full freedom of use)
- Reconfiguration at runtime with different setups

NICOS solutions

- Python is implementation and user language
- NICOS knows
  - Devices (implemented as Python objects)
  - Commands (for routine tasks with full freedom of use)
- Reconfiguration at runtime with different setups
- Full „dry run“ mode for error checking and timing
  - Syntax checks
  - Limit checks
  - Runtime estimation

NICOS solutions

- Python is implementation and user language
- NICOS knows
  - Devices (implemented as Python objects)
  - Commands (for routine tasks with full freedom of use)
- Reconfiguration at runtime with different setups
- Full „dry run“ mode for error checking and timing
  - Syntax checks
  - Limit checks
  - Runtime estimation
- State collection and automatic archiving

NICOS solutions

- Python is implementation and user language
- NICOS knows
  - Devices (implemented as Python objects)
  - Commands (for routine tasks with full freedom of use)
- Reconfiguration at runtime with different setups
- Full „dry run“ mode for error checking and timing
  - Syntax checks
  - Limit checks
  - Runtime estimation
- State collection and automatic archiving
- Different user interfaces: configurable GUI, console

NICOS solutions

- Python is implementation and user language
- NICOS knows
  - Devices (implemented as Python objects)
  - Commands (for routine tasks with full freedom of use)
- Reconfiguration at runtime with different setups
- Full „dry run“ mode for error checking and timing
  - Syntax checks
  - Limit checks
  - Runtime estimation
- State collection and automatic archiving
- Different user interfaces: configurable GUI, console
- Hardware „independent“ (TANGO, EPICS, SECoP, CARESS, ...)

History of NICOS

- NICOS (Networked Instrument COntrol System) started in 2002
- Ran on 5 instruments
- Very simple, not a client/server system
- Diverged lines of development and code states (CVS, not code review)

History of NICOS

- NICOS (Networked Instrument COntrol System) started in 2002
- Ran on 5 instruments
- Very simple, not a client/server system
- Diverged lines of development and code states (CVS, not code review)
- 2009 discussions about modernizing between user and instrument control groups, start rewriting of NICOS (client/server and GUI), first commit 2009/11/10

History of NICOS

- NICOS (Networked Instrument COntrol System) started in 2002
- Ran on 5 instruments
- Very simple, not a client/server system
- Diverged lines of development and code states (CVS, not code review)
- 2009 discussions about modernizing between user and instrument control groups, start rewriting of NICOS (client/server and GUI), first commit 2009/11/10
- 2011 switch to git/gerrit/Jenkins/Redmine in the development

History of NICOS

- NICOS (Networked Instrument COntrol System) started in 2002
- Ran on 5 instruments
- Very simple, not a client/server system
- Diverged lines of development and code states (CVS, not code review)
- 2009 discussions about modernizing between user and instrument control groups, start rewriting of NICOS (client/server and GUI), first commit 2009/11/10
- 2011 switch to git/gerrit/Jenkins/Redmine in the development
- 2012 „NICOS 2“ was stable and fully featured

History of NICOS

- NICOS (Networked Instrument COntrol System) started in 2002
- Ran on 5 instruments
- Very simple, not a client/server system
- Diverged lines of development and code states (CVS, not code review)
- 2009 discussions about modernizing between user and instrument control groups, start rewriting of NICOS (client/server and GUI), first commit 2009/11/10
- 2011 switch to git/gerrit/Jenkins/Redmine in the development
- 2012 „NICOS 2“ was stable and fully featured
- 2013 MLZ decision, NICOS is standard control system

History of NICOS

- NICOS (Networked Instrument COntrol System) started in 2002
- Ran on 5 instruments
- Very simple, not a client/server system
- Diverged lines of development and code states (CVS, not code review)
- 2009 discussions about modernizing between user and instrument control groups, start rewriting of NICOS (client/server and GUI), first commit 2009/11/10
- 2011 switch to git/gerrit/Jenkins/Redmine in the development
- 2012 „NICOS 2“ was stable and fully featured
- 2013 MLZ decision, NICOS is standard control system
- 2015 collaboration with ESS started

History of NICOS

- NICOS (Networked Instrument COntrol System) started in 2002
- Ran on 5 instruments
- Very simple, not a client/server system
- Diverged lines of development and code states (CVS, not code review)
- 2009 discussions about modernizing between user and instrument control groups, start rewriting of NICOS (client/server and GUI), first commit 2009/11/10
- 2011 switch to git/gerrit/Jenkins/Redmine in the development
- 2012 „NICOS 2“ was stable and fully featured
- 2013 MLZ decision, NICOS is standard control system
- 2015 collaboration with ESS started
- 2018 PSI joined the NICOS developer group

History of NICOS

- NICOS (Networked Instrument COntrol System) started in 2002
- Ran on 5 instruments
- Very simple, not a client/server system
- Diverged lines of development and code states (CVS, not code review)
- 2009 discussions about modernizing between user and instrument control groups, start rewriting of NICOS (client/server and GUI), first commit 2009/11/10
- 2011 switch to git/gerrit/Jenkins/Redmine in the development
- 2012 „NICOS 2“ was stable and fully featured
- 2013 MLZ decision, NICOS is standard control system
- 2015 collaboration with ESS started
- 2018 PSI joined the NICOS developer group
- 2018 INL (US, Aaron Craft) joined the NICOS users group

History of NICOS

- NICOS (Networked Instrument COntrol System) started in 2002
- Ran on 5 instruments
- Very simple, not a client/server system
- Diverged lines of development and code states (CVS, not code review)
- 2009 discussions about modernizing between user and instrument control groups, start rewriting of NICOS (client/server and GUI), first commit 2009/11/10
- 2011 switch to git/gerrit/Jenkins/Redmine in the development
- 2012 „NICOS 2“ was stable and fully featured
- 2013 MLZ decision, NICOS is standard control system
- 2015 collaboration with ESS started
- 2018 PSI joined the NICOS developer group
- 2018 INL (US, Aaron Craft) joined the NICOS users group
- 2019/2020 MGML (Charles University Prague), LAHN (Argentina) facility, and TU Vienna started to use NICOS

History of NICOS

- NICOS (Networked Instrument COntrol System) started in 2002
- Ran on 5 instruments
- Very simple, not a client/server system
- Diverged lines of development and code states (CVS, not code review)
- 2009 discussions about modernizing between user and instrument control groups, start rewriting of NICOS (client/server and GUI), first commit 2009/11/10
- 2011 switch to git/gerrit/Jenkins/Redmine in the development
- 2012 „NICOS 2“ was stable and fully featured
- 2013 MLZ decision, NICOS is standard control system
- 2015 collaboration with ESS started
- 2018 PSI joined the NICOS developer group
- 2018 INL (US, Aaron Craft) joined the NICOS users group
- 2019/2020 MGML (Charles University Prague), LAHN (Argentina) facility, and TU Vienna started to use NICOS
- 2025 BNC ?

Design principles of NICOS

Design principles of NICOS

- Accessibility:
  - Powerful to „power users“
  - Easy to learn for new users

Design principles of NICOS

- Accessibility:
  - Powerful to „power users“
  - Easy to learn for new users
- Flexibility:
  - Integration of devices by instrument scientists
  - Quick reaction to new experiment conditions

Design principles of NICOS

- Accessibility:
  - Powerful to „power users“
  - Easy to learn for new users
- Flexibility:
  - Integration of devices by instrument scientists
  - Quick reaction to new experiment conditions
- Reliability:
  - Permanent storing of experiment state
  - Recovering previous state

Design principles of NICOS

- Accessibility:
  - Powerful to „power users“
  - Easy to learn for new users
- Flexibility:
  - Integration of devices by instrument scientists
  - Quick reaction to new experiment conditions
- Reliability:
  - Permanent storing of experiment state
  - Recovering previous state
- Lightweight:
  - Less code is better
  - Reuse existing code as much as possible
  - Better to configure than to code

Design principles of NICOS

- Network based:
  - Multi client
  - Multi user

Design principles of NICOS

- Network based:
  - Multi client
  - Multi user
- Science driven
  - Close cooperation with instrument scientists and/or users
  - No feature without any need

Design principles of NICOS

- Network based:
  - Multi client
  - Multi user
- Science driven
  - Close cooperation with instrument scientists and/or users
  - No feature without any need
- Open source
  - Everybody is welcome to use and contribute

Supported instrument types

- Triple axes

Supported instrument types

- Triple axes
- Reflectometer (Neutrons and X-Ray)

Supported instrument types

- Triple axes
- Reflectometer (Neutrons and X-Ray)
- Tomography

Supported instrument types

- Triple axes
- Reflectometer (Neutrons and X-Ray)
- Tomography
- Powder diffractometer

Supported instrument types

- Triple axes
- Reflectometer (Neutrons and X-Ray)
- Tomography
- Powder diffractometer
- Residual stress (Neutron and X-Ray)

Supported instrument types

- Triple axes
- Reflectometer (Neutrons and X-Ray)
- Tomography
- Powder diffractometer
- Residual stress (Neutron and X-Ray)
- Single crystal spectrometer

Supported instrument types

- Triple axes
- Reflectometer (Neutrons and X-Ray)
- Tomography
- Powder diffractometer
- Residual stress (Neutron and X-Ray)
- Single crystal spectrometer
- Small angle scattering

Supported instrument types

- Triple axes
- Reflectometer (Neutrons and X-Ray)
- Tomography
- Powder diffractometer
- Residual stress (Neutron and X-Ray)
- Single crystal spectrometer
- Small angle scattering
- ...

Setups

- Configuration of an instrument is divided into several „setups“

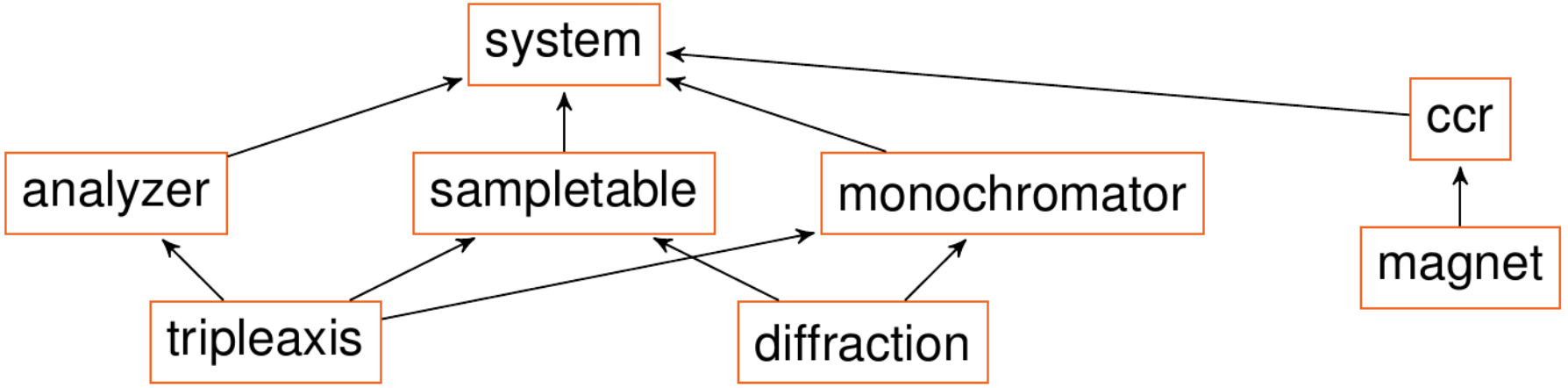
Setups

- Configuration of an instrument is divided into several „setups“
- A setup is a simple Python file defining devices and their parameters

```
description = 'MIRA1 monochromator'
group = 'lowlevel'

includes = ['base', 'sample', 'alias_mono']

tango_base = 'tango://miractrl.mira.frm2.tum.de:10000/mira/'

devices = dict(
    co_m1tt = device('nicos.devices.entangle.Sensor',
        visibility = (),
        tangodevice = tango_base + 'mono1/mtt_enc',
        unit = 'deg',
        precision = 0.05,
    ),
    mo_m1tt = device('nicos.devices.entangle.Motor',
        tangodevice = tango_base + 'mono1/mtt_mot',
        visibility = (),
        precision = 0.05,
    ),
    m1tt = device('nicos_mlz.mira.devices.axis.HoveringAxis',
        description = 'monochromator two-theta angle',
        abslimits = (-50.0, 0),
        motor = 'mo_m1tt',
        coder = 'co_m1tt',
        startdelay = 1,
        stopdelay = 4,
        switch = 'air_mono',
        switchvalues = (0, 1),
        fmtstr = '%.3f',
        precision = 0.05,
    ),
```

Setups

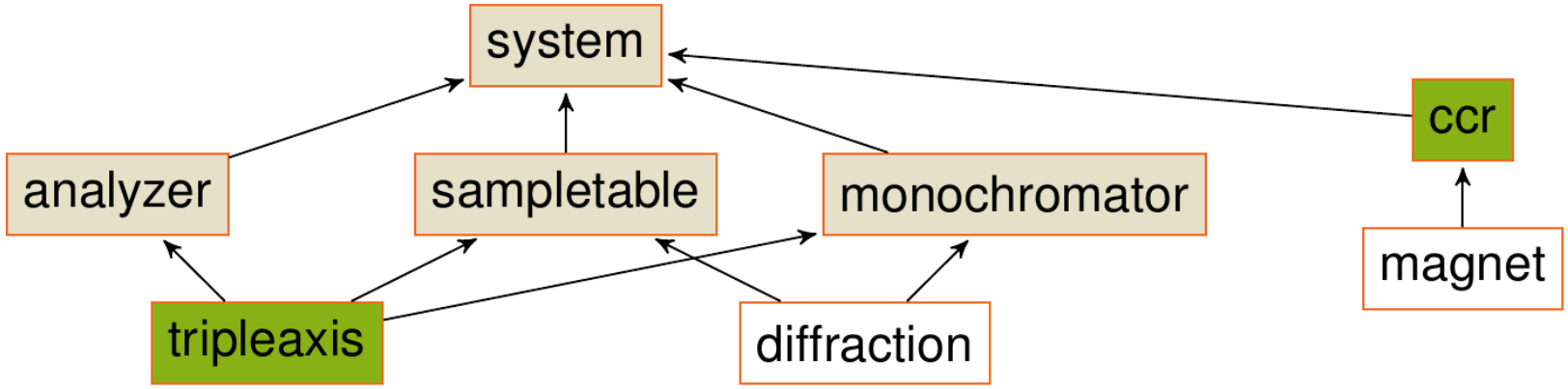
- Configuration of an instrument is divided into several „setups“
- A setup is a simple Python file defining devices and their parameters
- A setup may depend on other setups (used or use)



Setups

- Configuration of an instrument is divided into several „setups“
- A setup is a simple Python file defining devices and their parameters
- A setup may depend on other setups (used or use)

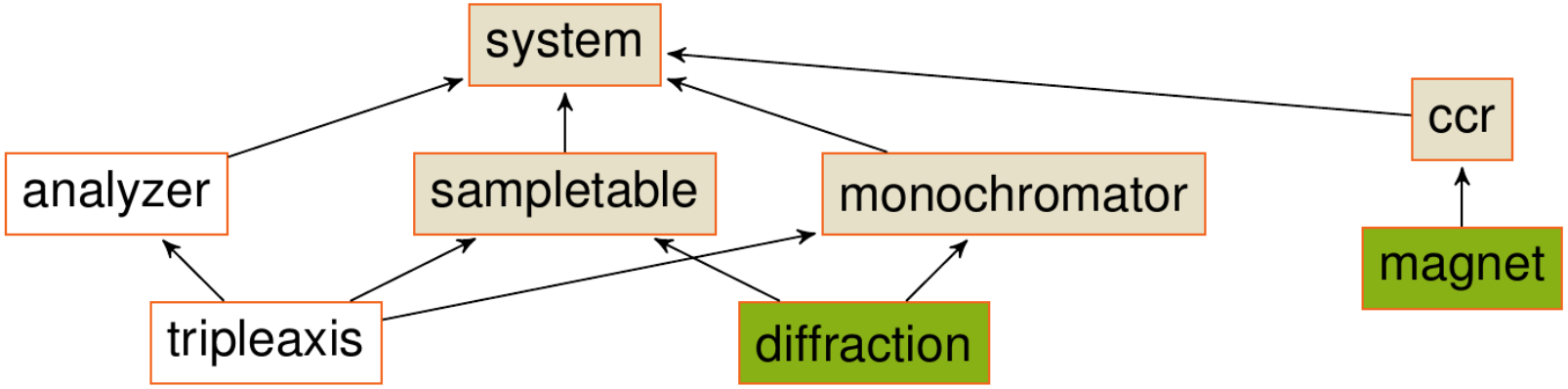
```
NewSetup('tripleaxis', 'ccr')
```



Setups

- Configuration of an instrument is divided into several „setups“
- A setup is a simple Python file defining devices and their parameters
- A setup may depend on other setups (used or use)

```
NewSetup('diffraction', 'magnet')
```



Devices in NICOS

Devices in NICOS

- NICOS encapsulates access to hardware in „Devices“

Devices in NICOS

- NICOS encapsulates access to hardware in „Devices“
- Each device will be represented by a configurable Python class

Devices in NICOS

- NICOS encapsulates access to hardware in „Devices“
- Each device will be represented by a configurable Python class
- NICOS has a hierarchy of devices with a set of base classes
  - define the interface, which has to implemented in derived classes

Devices in NICOS

- NICOS encapsulates access to hardware in „Devices“
- Each device will be represented by a configurable Python class
- NICOS has a hierarchy of devices with a set of base classes
  - define the interface, which has to implemented in derived classes
- A number of generic device and helper classes

Devices in NICOS

- NICOS encapsulates access to hardware in „Devices“
- Each device will be represented by a configurable Python class
- NICOS has a hierarchy of devices with a set of base classes
  - define the interface, which has to implemented in derived classes
- A number of generic device and helper classes
- Device classes may inherite other classes or simply use them

Devices in NICOS

- NICOS encapsulates access to hardware in „Devices“
- Each device will be represented by a configurable Python class
- NICOS has a hierarchy of devices with a set of base classes
  - define the interface, which has to implemented in derived classes
- A number of generic device and helper classes
- Device classes may inherite other classes or simply use them
- Configuration parameters:
  - Device parameters (read/write access during runtime)
  - Values and their changes will be archived

Devices in NICOS

- NICOS encapsulates access to hardware in „Devices“
- Each device will be represented by a configurable Python class
- NICOS has a hierarchy of devices with a set of base classes
  - define the interface, which has to implemented in derived classes
- A number of generic device and helper classes
- Device classes may inherite other classes or simply use them
- Configuration parameters:
  - Device parameters (read/write access during runtime)
  - Values and their changes will be archived
- Device value and status will be automatically archived

User interface

User interface

- Most basic user interface: Python **command** prompt

User interface

- Most basic user interface: Python **command** prompt
- Simple commands:
  - `read(device), move(device, target), status(device),…`

User interface

- Most basic user interface: Python **command** prompt
- Simple commands:
  - `read(device), move(device, target), status(device),…`
- Scanning commands:
  - `scan(device, start, step, num, preset), ...`

User interface

- Most basic user interface: Python **command** prompt
- Simple commands:
  - `read(device), move(device, target), status(device),…`
- Scanning commands:
  - `scan(device, start, step, num, preset), ...`
- Integrated help system:
  - `help(object), ListDevices(), ListSetups(),...`

User interface

- Most basic user interface: Python **command** prompt
- Simple commands:
  - `read(device), move(device, target), status(device),…`
- Scanning commands:
  - `scan(device, start, step, num, preset), ...`
- Integrated help system:
  - `help(object), ListDevices(), ListSetups(),...`
- Python control structures
  - **loops, conditions, variables, …**

User interface

- Most basic user interface: Python **command** prompt
- Simple commands:
  - `read(device), move(device, target), status(device),…`
- Scanning commands:
  - `scan(device, start, step, num, preset), ...`
- Integrated help system:
  - `help(object), ListDevices(), ListSetups(),...`
- Python control structures
  - **loops, conditions, variables, …**
- Execution environment is safe against user errors
  - **no overriding of devices and commands**

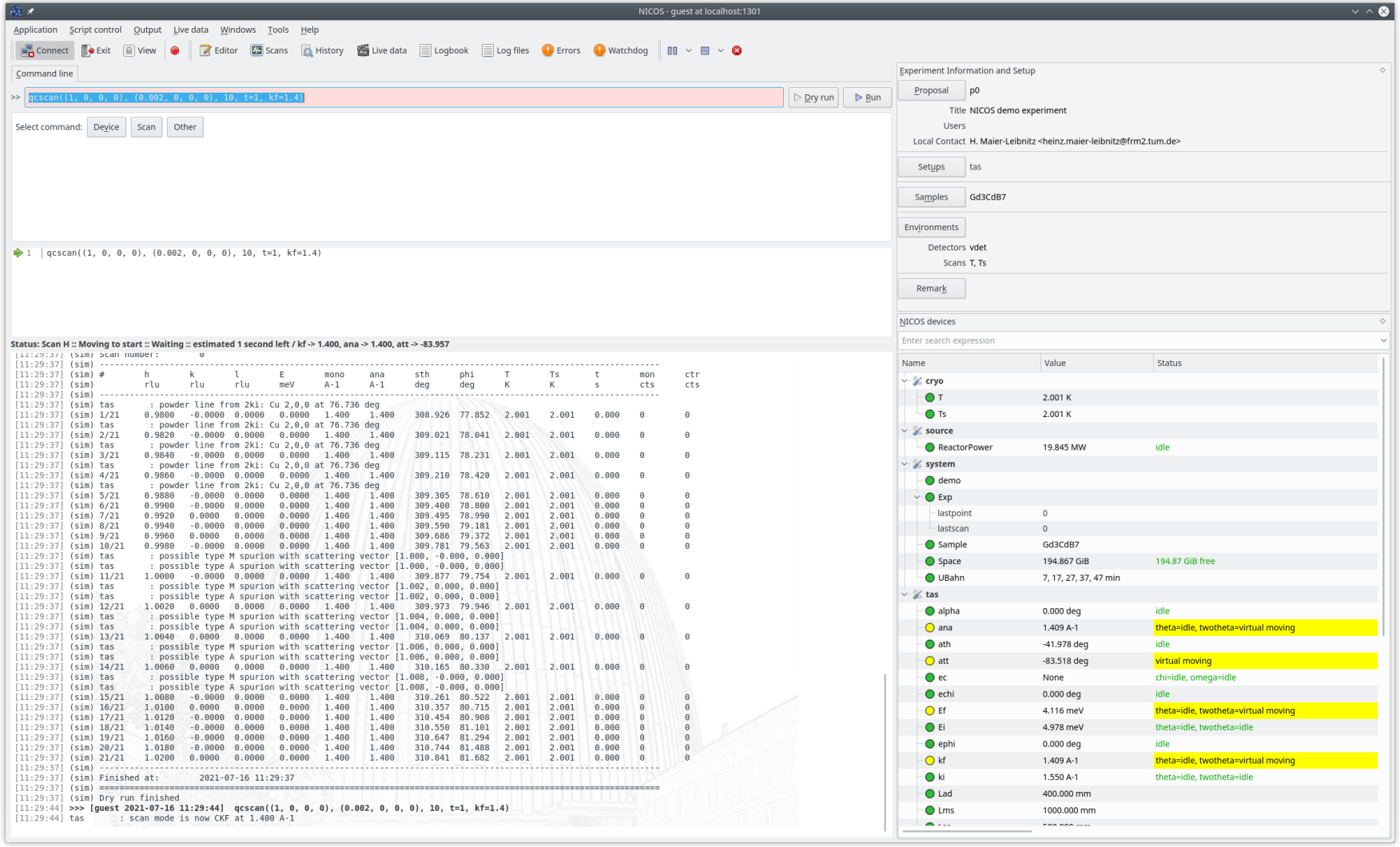
Graphical user interface

- Client-Server architecture: more than one connection possible

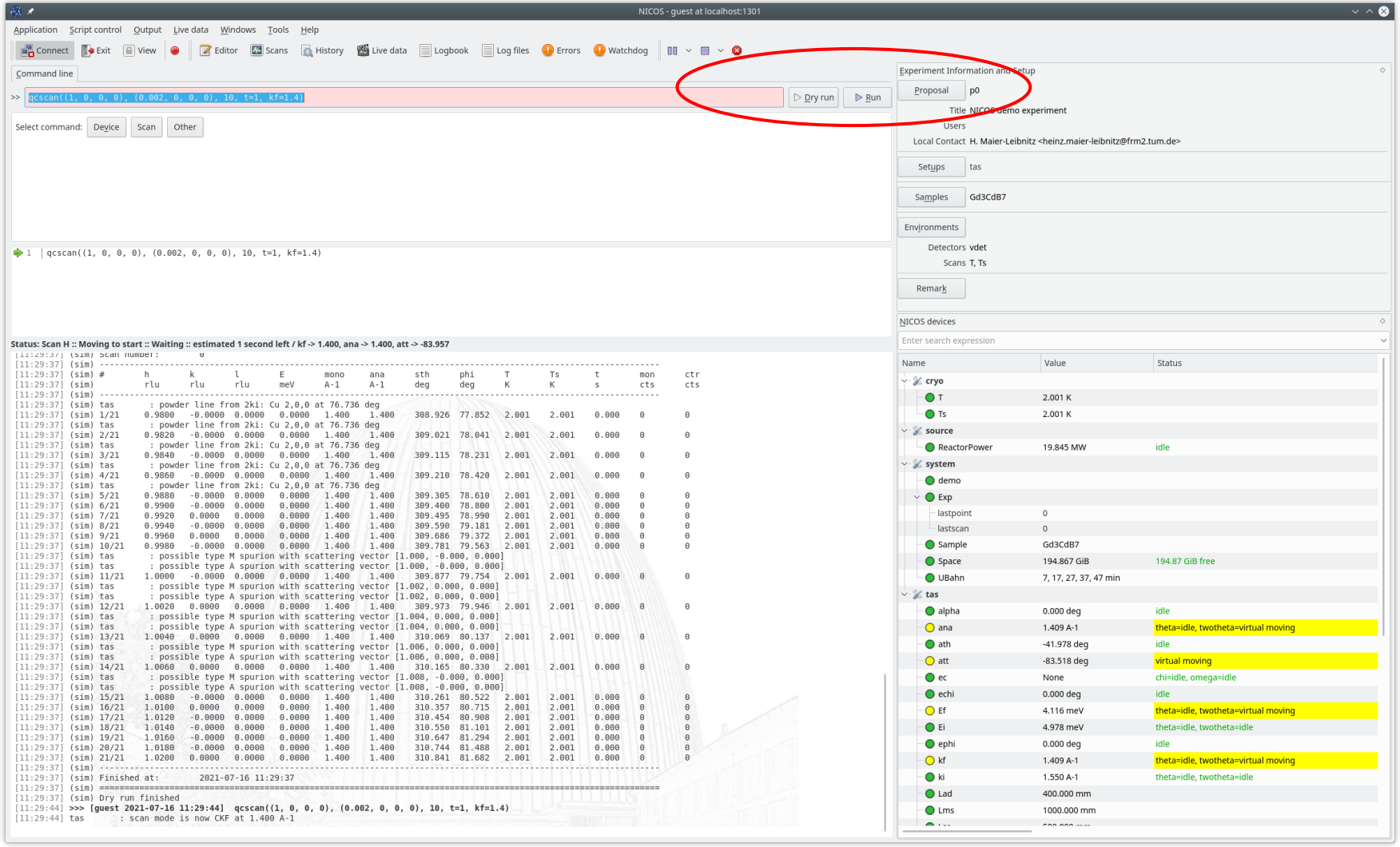
Graphical user interface

- Client-Server architecture: more than one connection possible
- Command line window

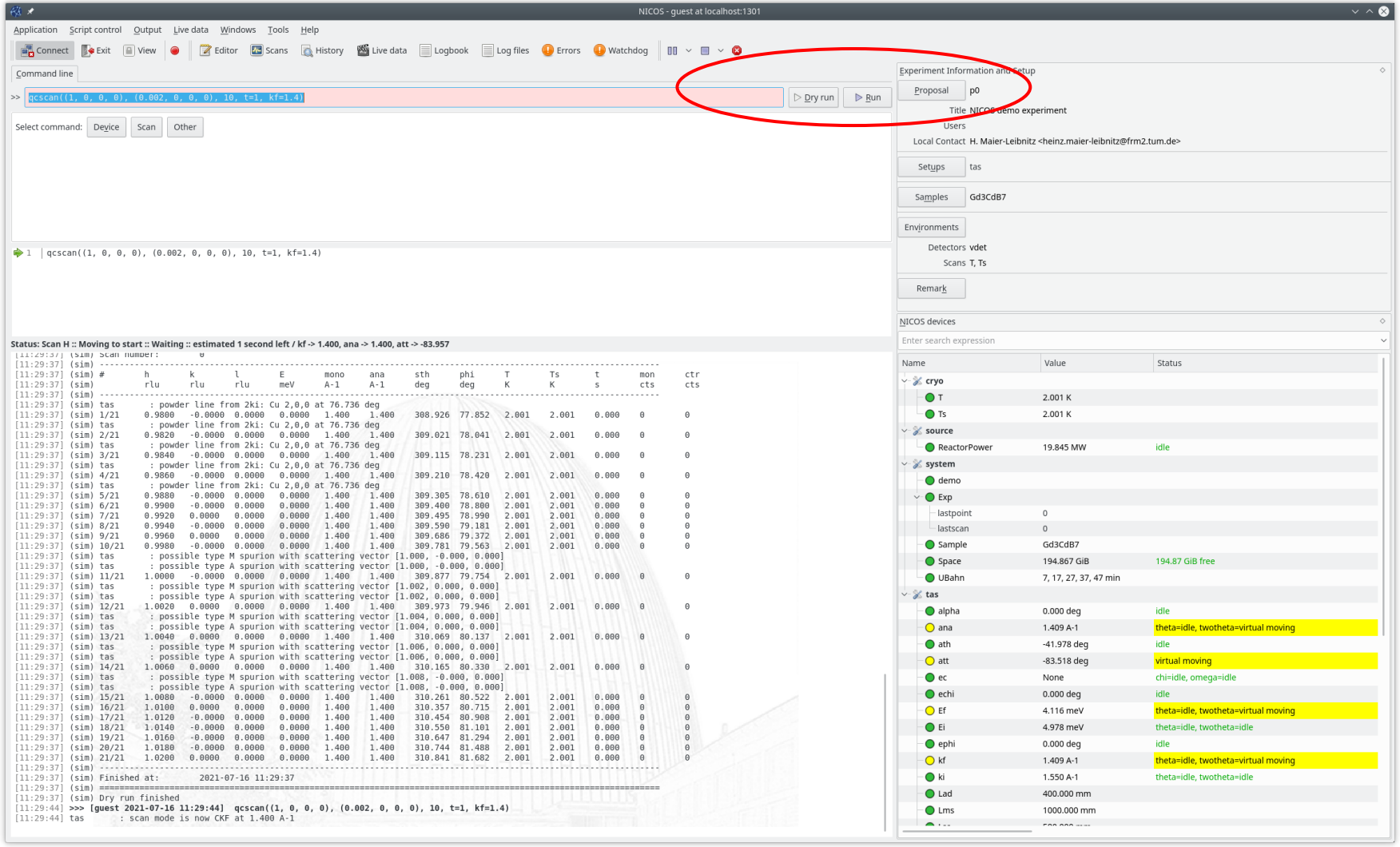
Graphical user interface



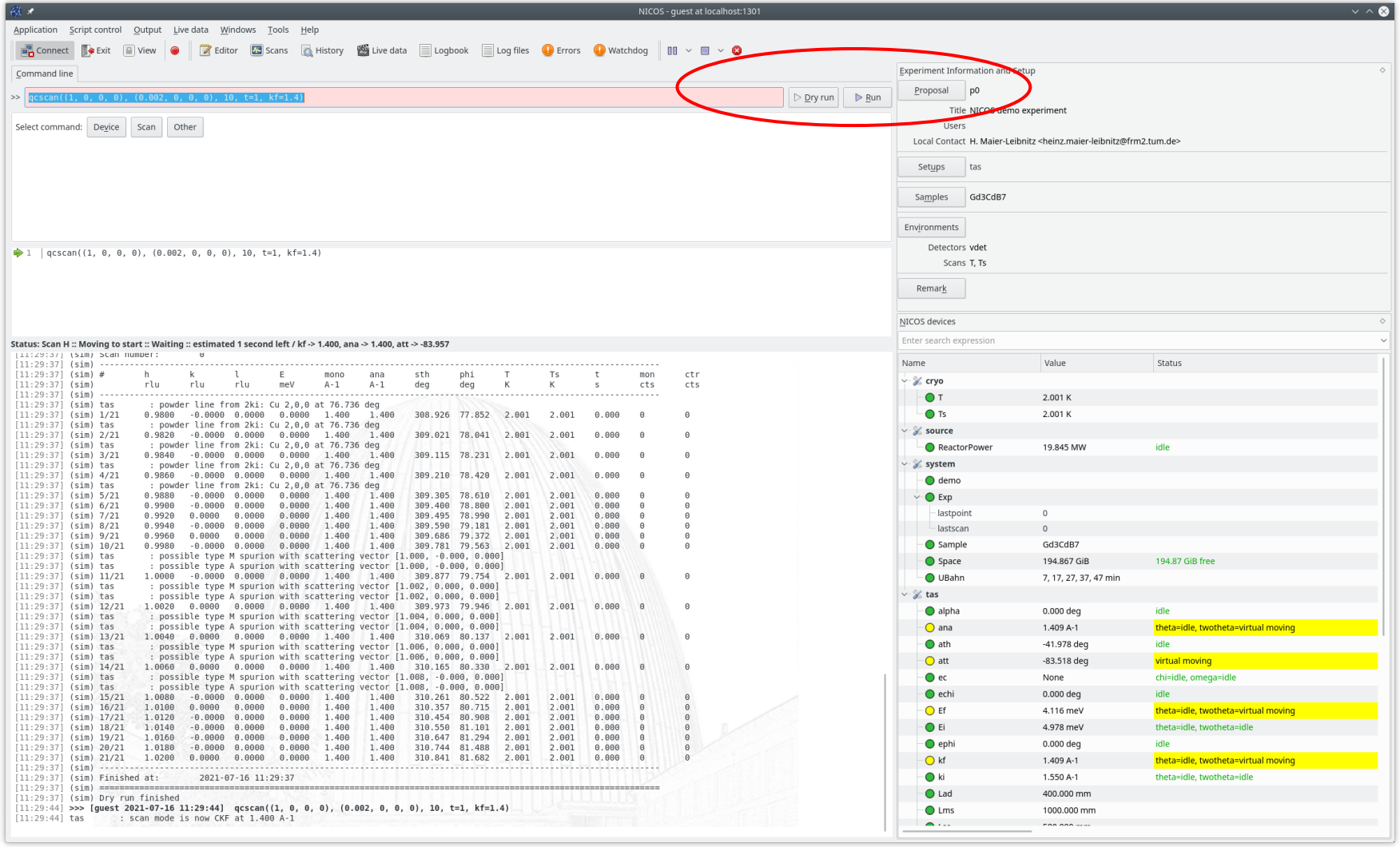
Graphical user interface



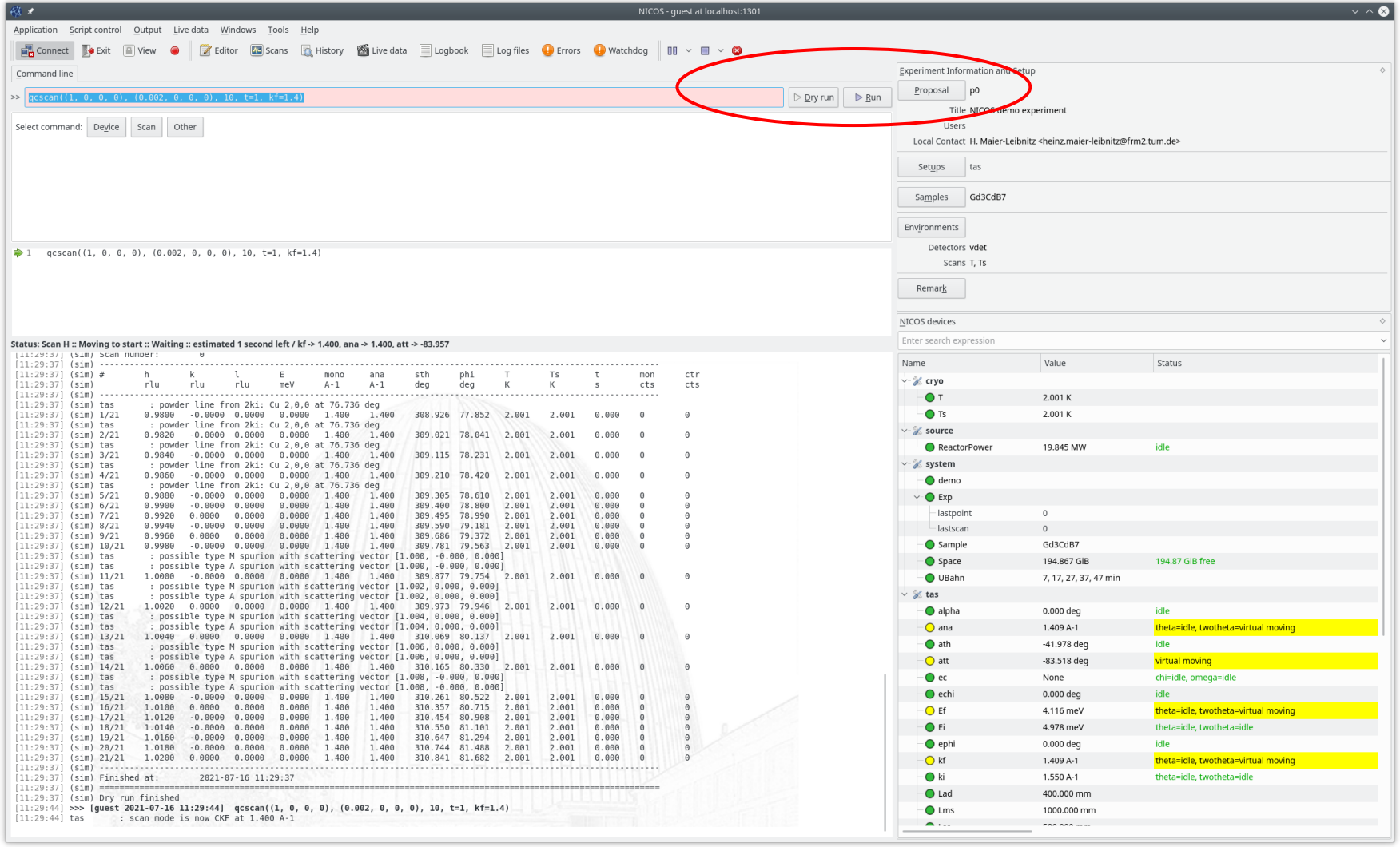
Graphical user interface



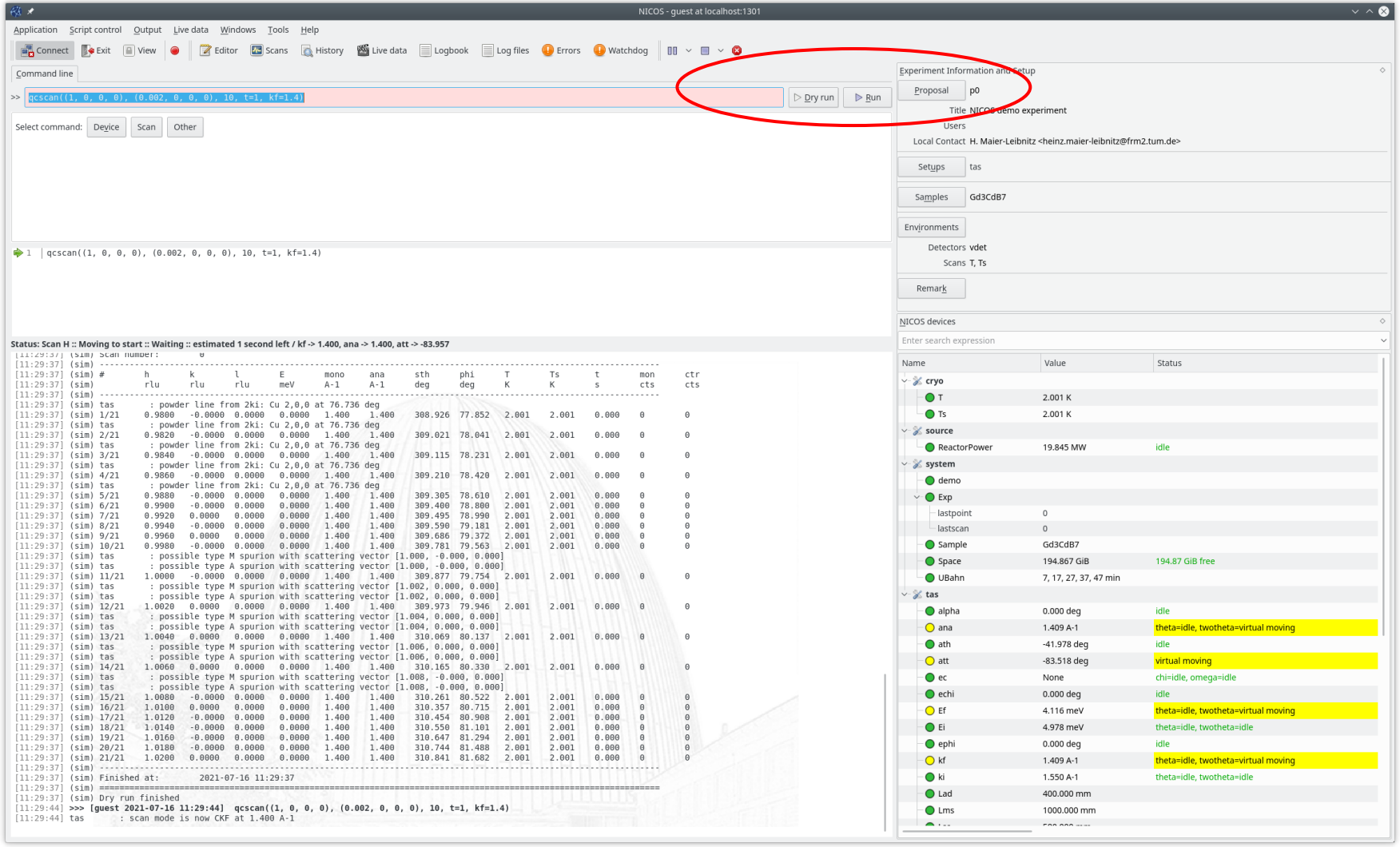
Graphical user interface



Graphical user interface



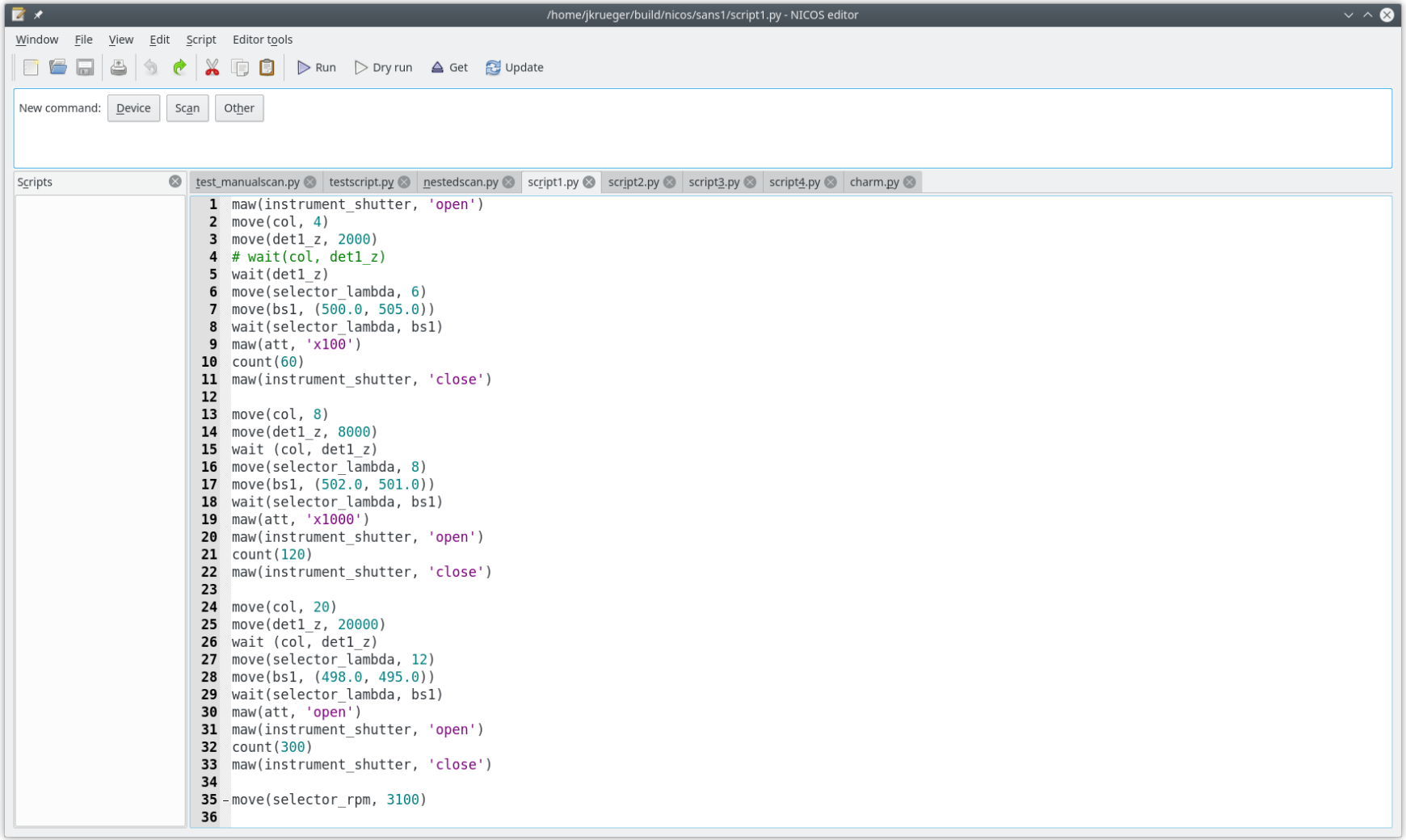
Graphical user interface



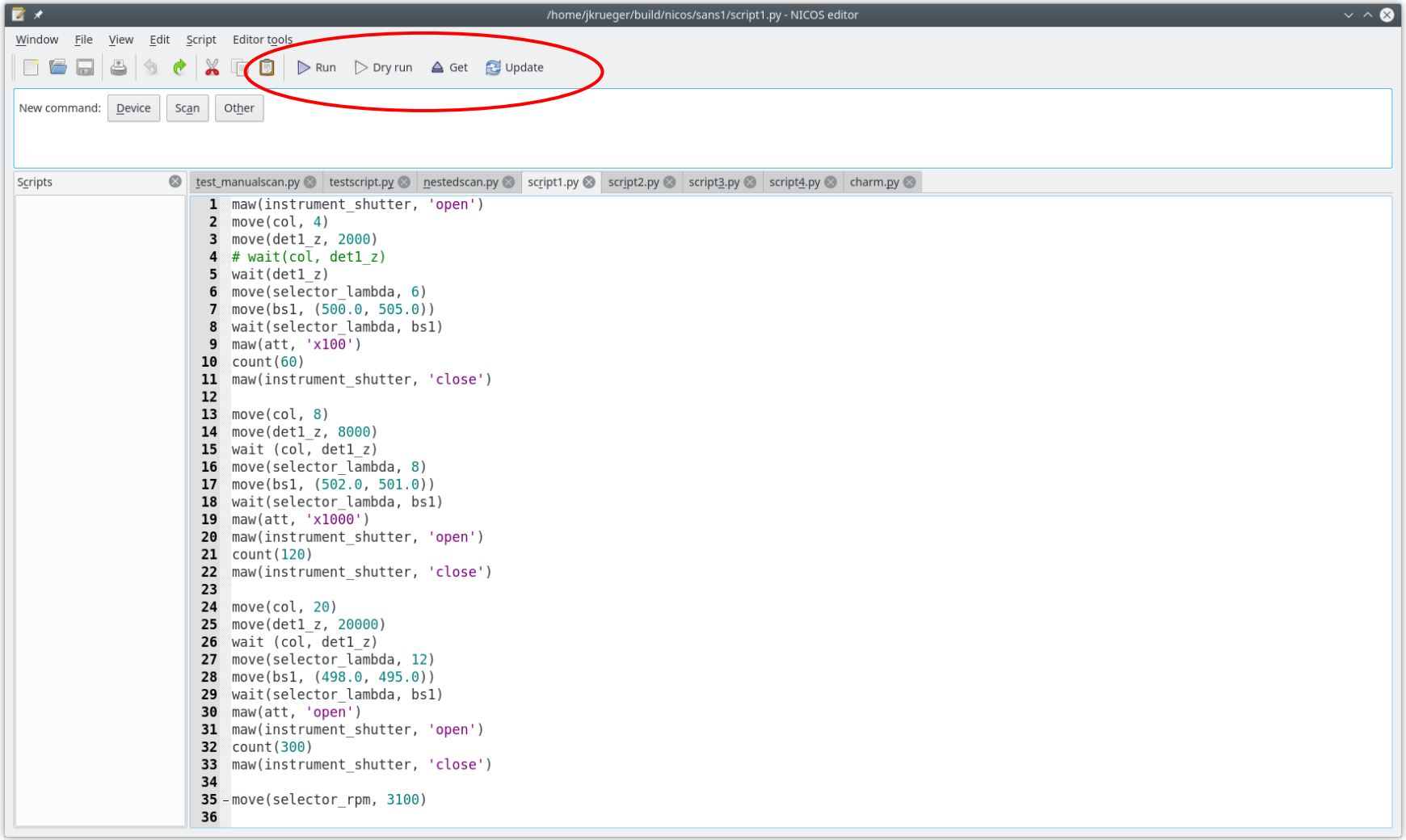
Graphical user interface

- Client-Server architecture: more than one connection possible
- Command line window
- User script editor

Graphical user interface



Graphical user interface

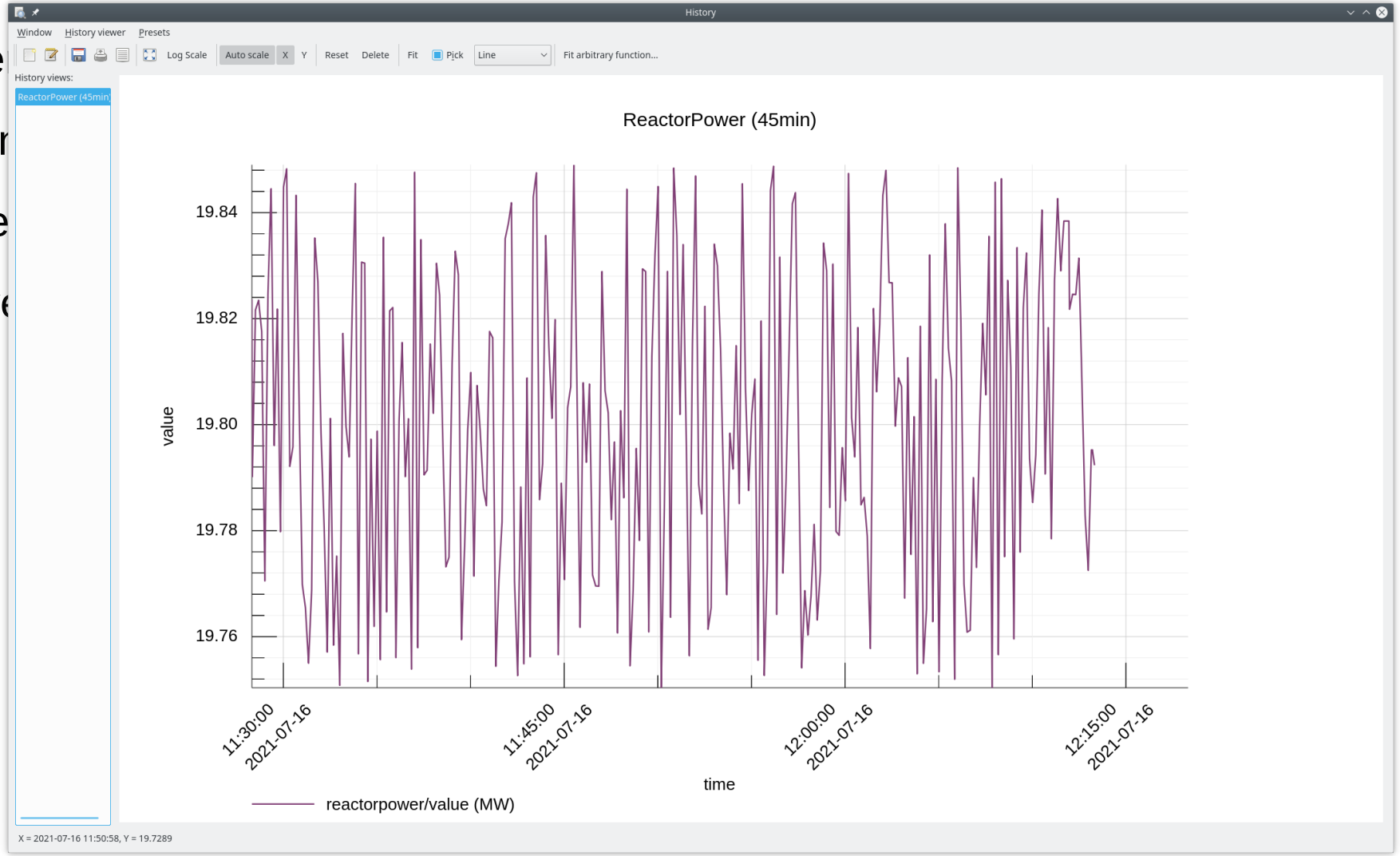


Graphical user interface

- Client-Server architecture: more than one connection possible
- Command line window
- User script editor
- History viewer: Display of archived data

Graphical user interface

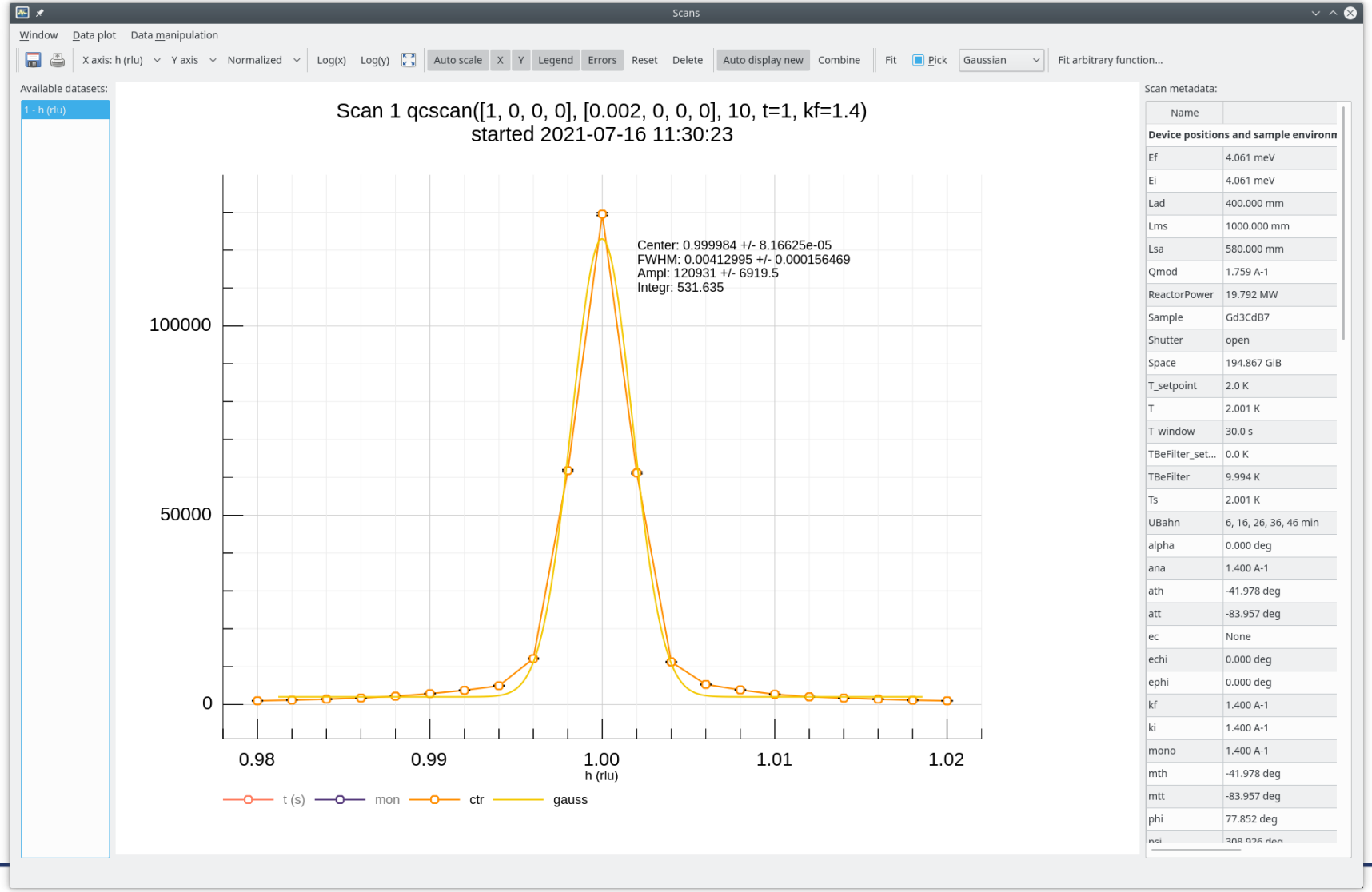
- Client-Serve
- Command lir
- User script e
- History viewe



Graphical user interface

- Client-Server architecture: more than one connection possible
- Command line window
- User script editor
- History viewer: Display of archived data
- On-line data display with preliminary fitting

Graphical user interface

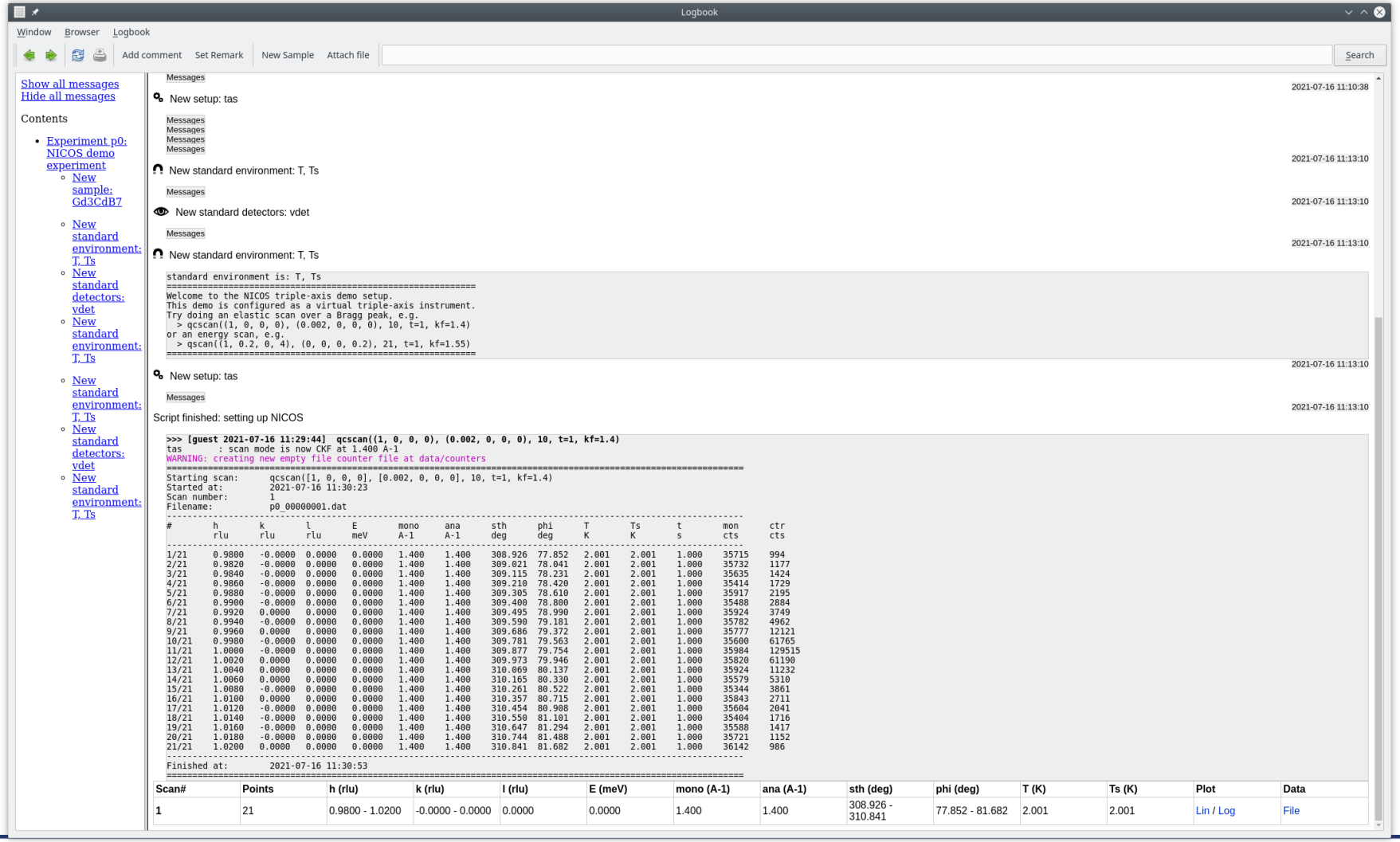


Graphical user interface

Graphical user interface

- Client-Server architecture: more than one connection possible
- Command line window
- User script editor
- History viewer: Display of archived data
- On-line data display with preliminary fitting
- Electronic logbook to show the experiment history

Graphical user interface



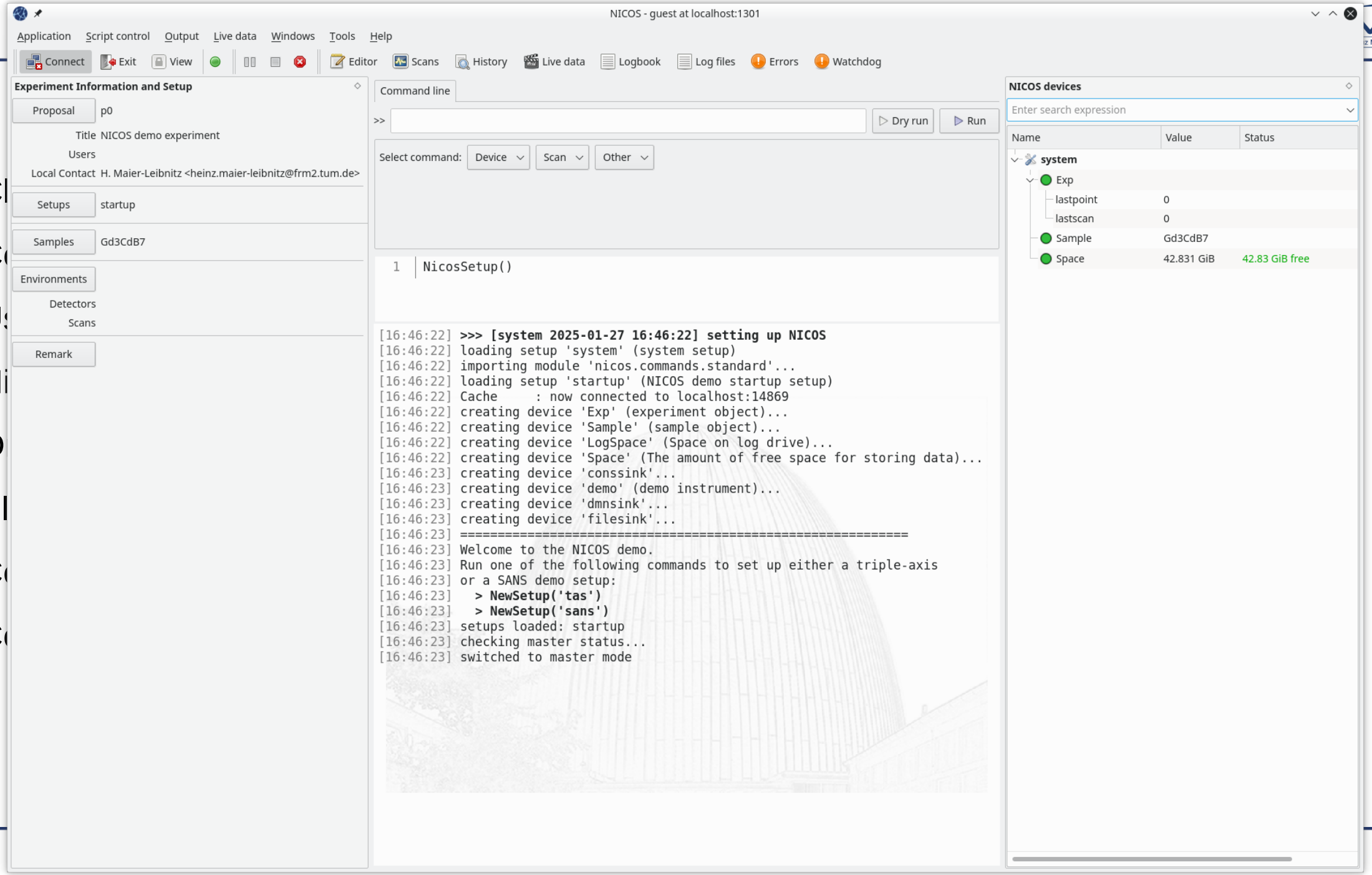
Graphical user interface

- Client-Server architecture: more than one connection possible
- Command line window
- User script editor
- History viewer: Display of archived data
- On-line data display with preliminary fitting
- Electronic logbook to show the experiment history
- Configurable and dynamic status monitor

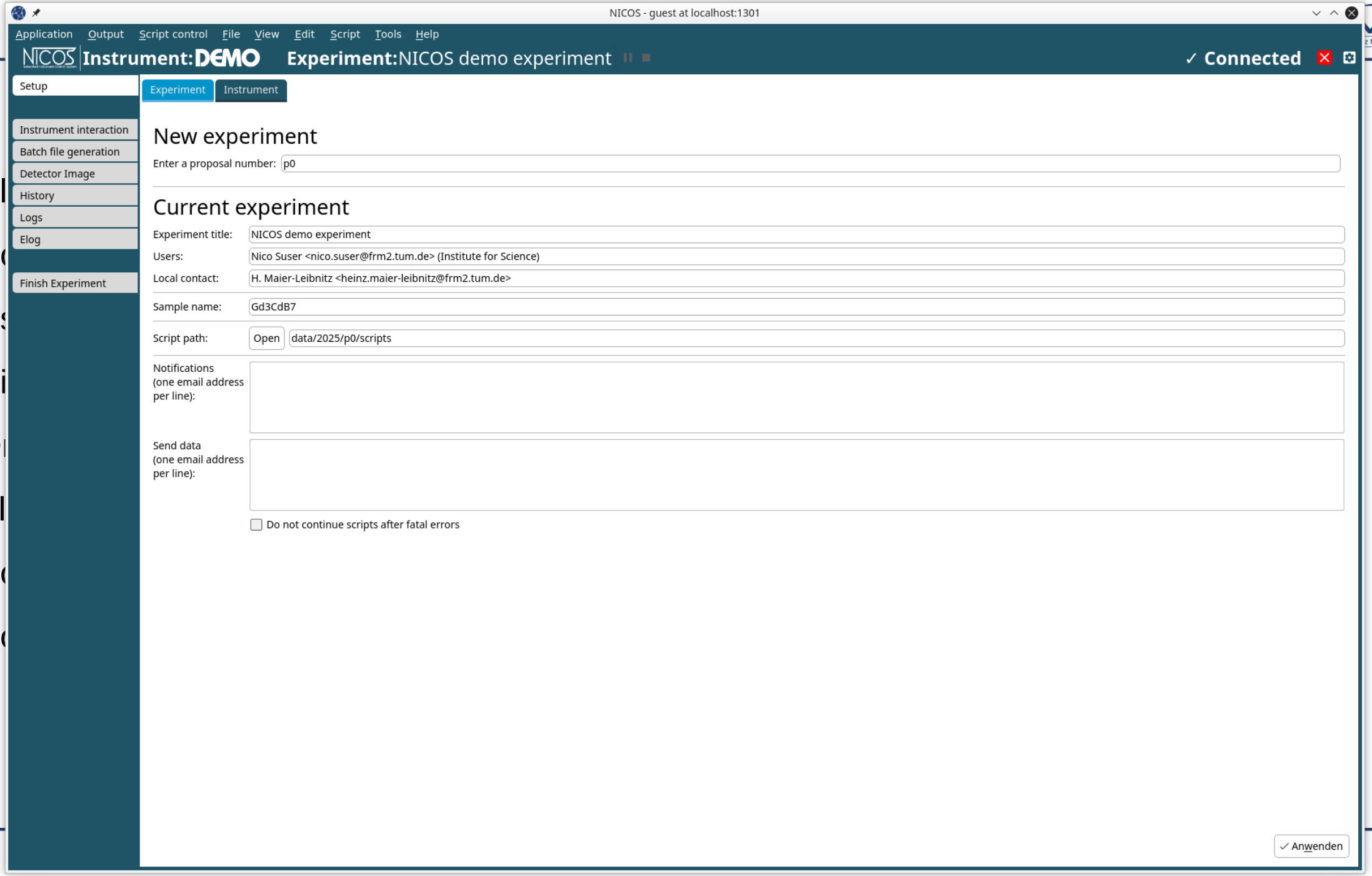
Graphical user interface

- Client-Server architecture: more than one connection possible
- Command line window
- User script editor
- History viewer: Display of archived data
- On-line data display with preliminary fitting
- Electronic logbook to show the experiment history
- Configurable and dynamic status monitor
- Configurable and dynamic

- Cl
- C
- Us
- Hi
- O
- El
- C
- C



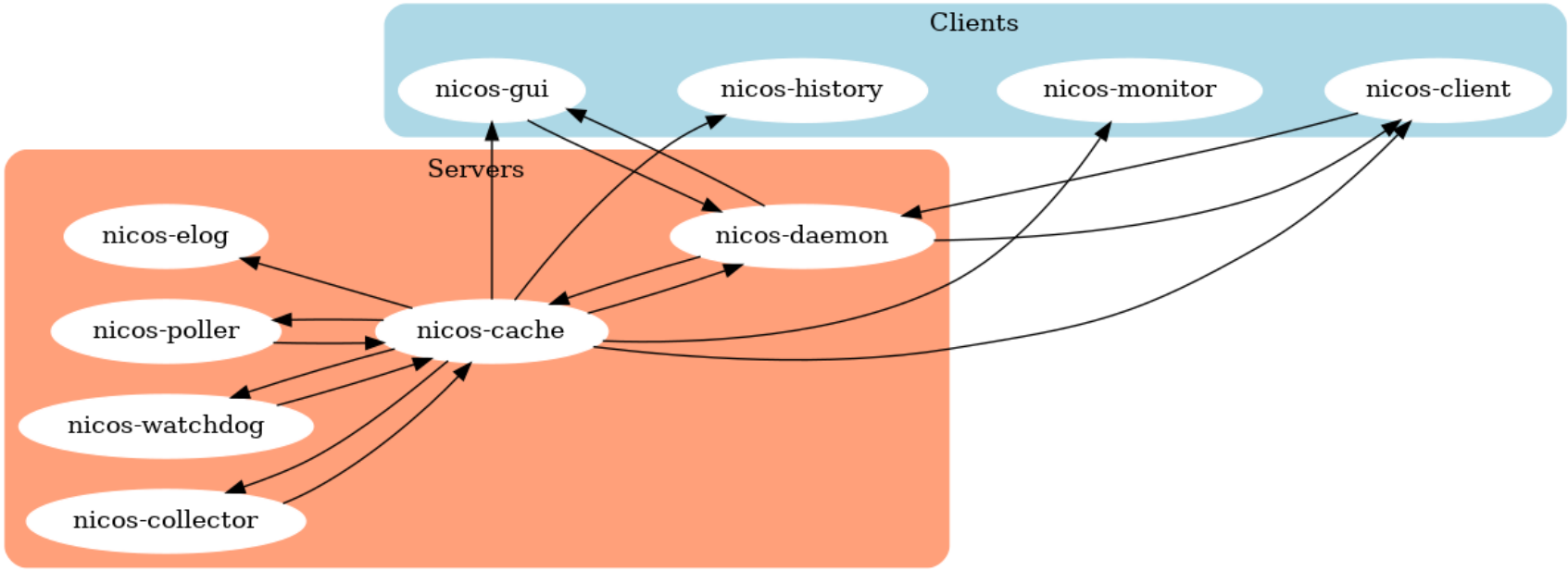
- Cl
- Cr
- Us
- Hi
- Op
- El
- Cr
- Cr



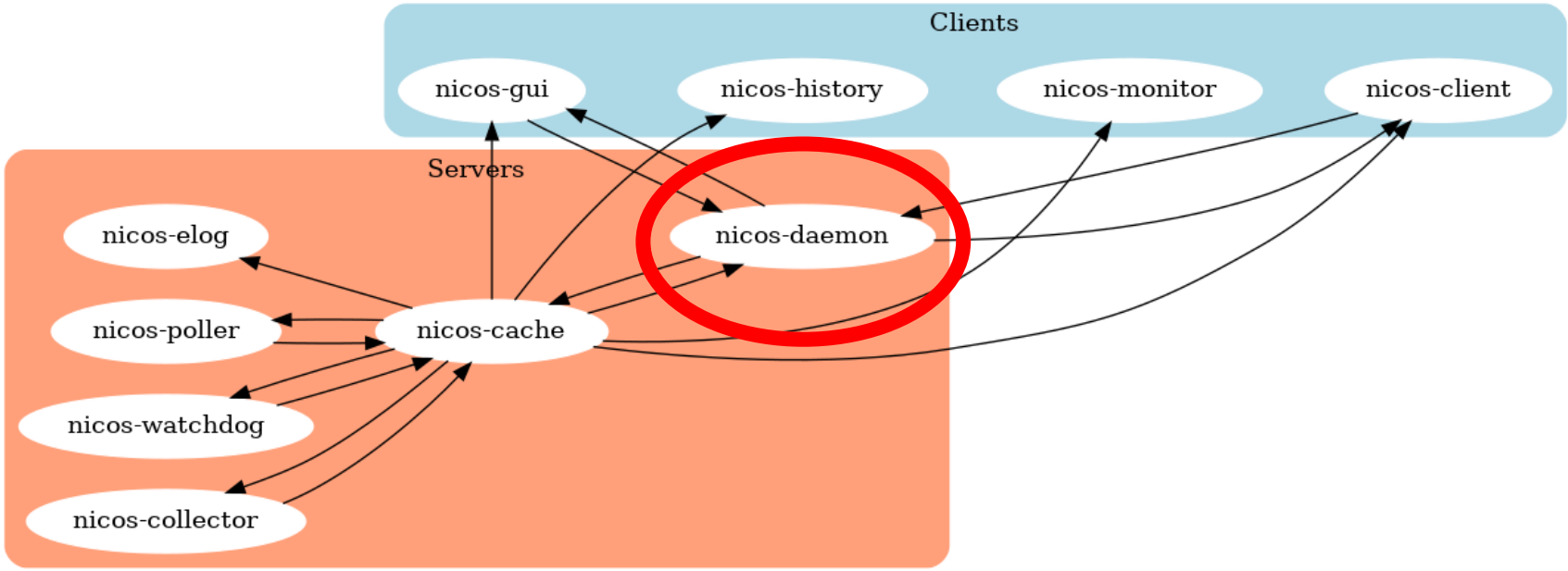
Graphical user interface

- Client-Server architecture: more than one connection possible
- Command line window
- User script editor
- History viewer: Display of archived data
- On-line data display with preliminary fitting
- Electronic logbook to show the experiment history
- Configurable and dynamic status monitor
- Configurable and dynamic
- Extendable by custom panels

Components

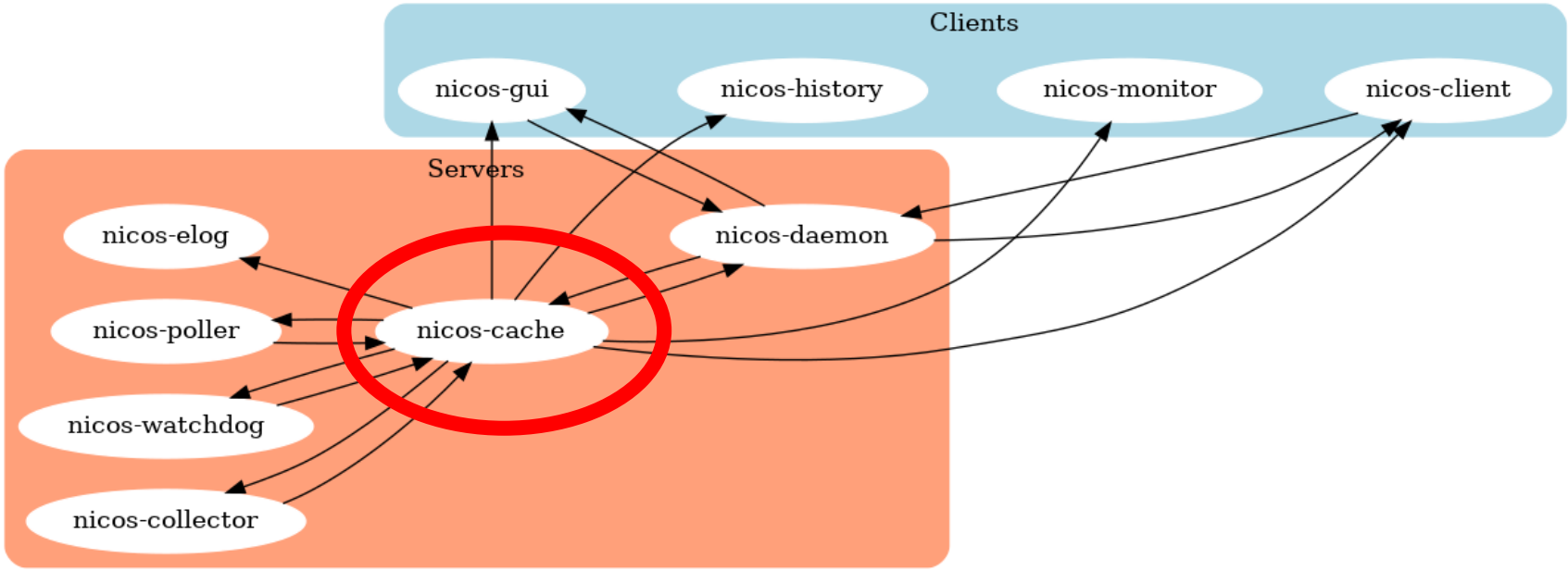

Clients
nicos-gui
nicos-history
nicos-monitor
nicos-client
Servers
nicos-elog
nicos-daemon
nicos-poller
nicos-cache
nicos-watchdog
nicos-collector

Components



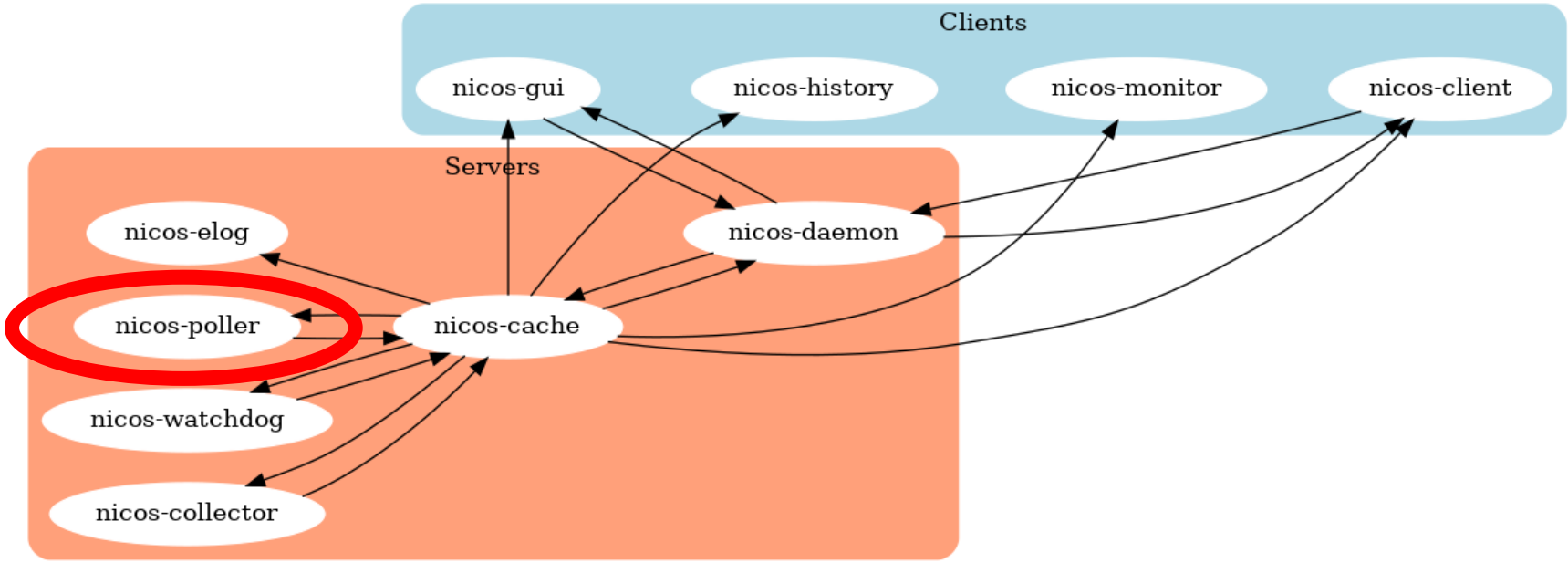
- Execution daemon: executes user input

Components



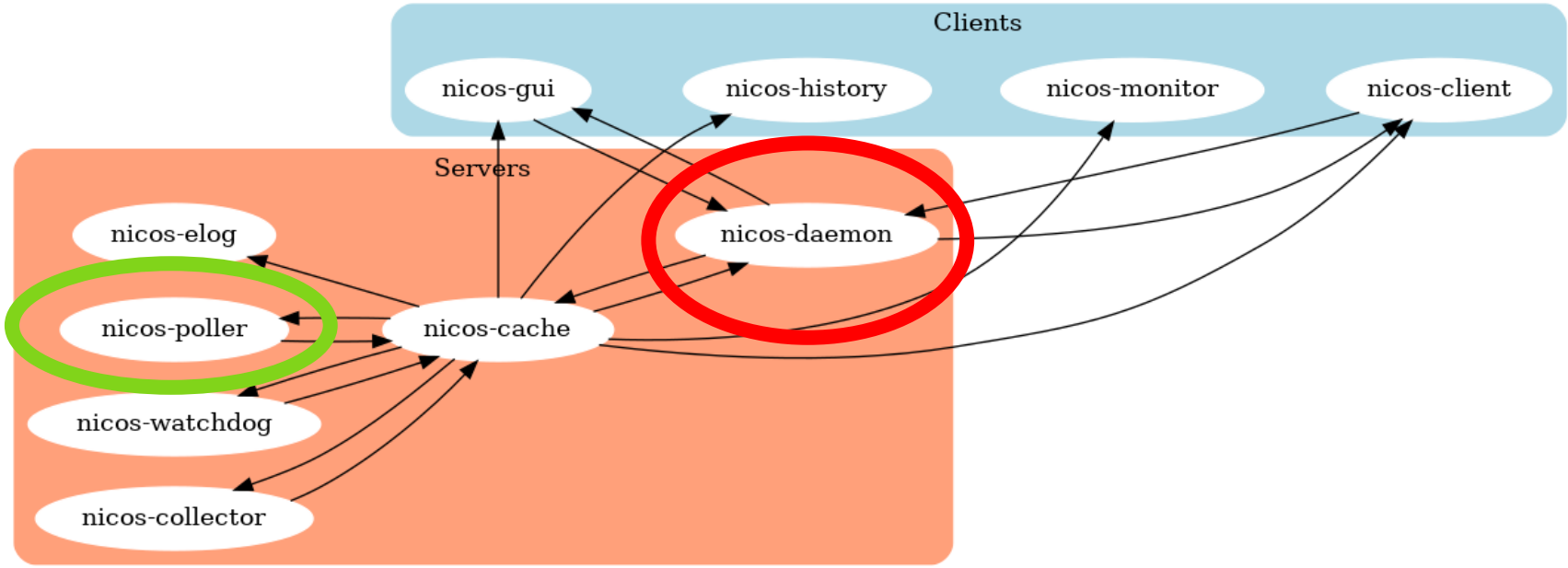
- Cache: save states/values/parameters, pushes value/state changes

Components



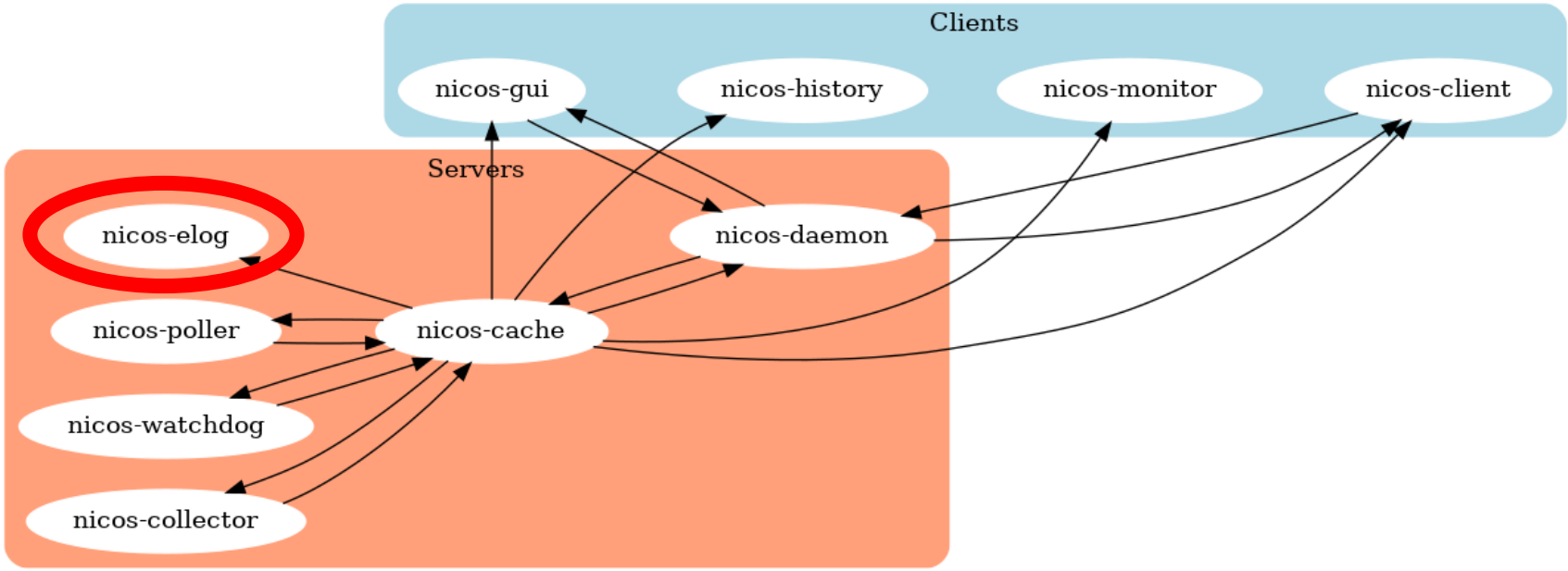
- Poller: fetches periodically values/parameter/states from devices

Components



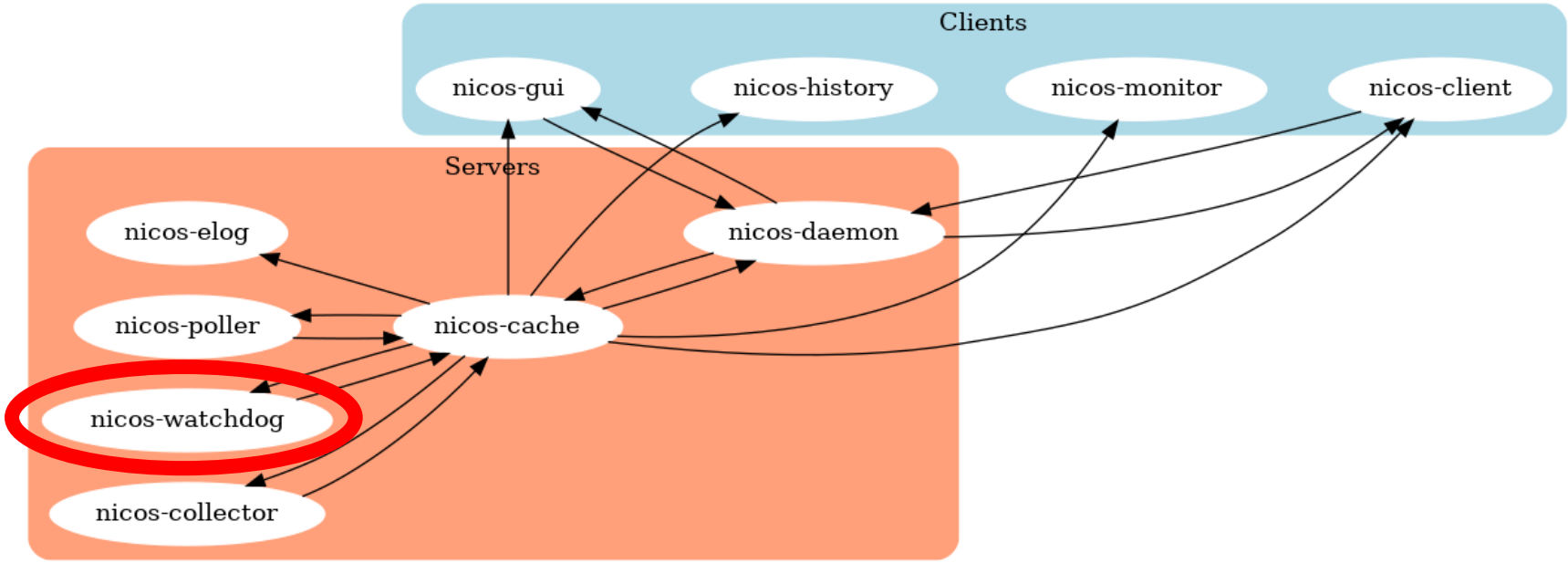
- Daemon and poller (read-only) interact with hardware

Components



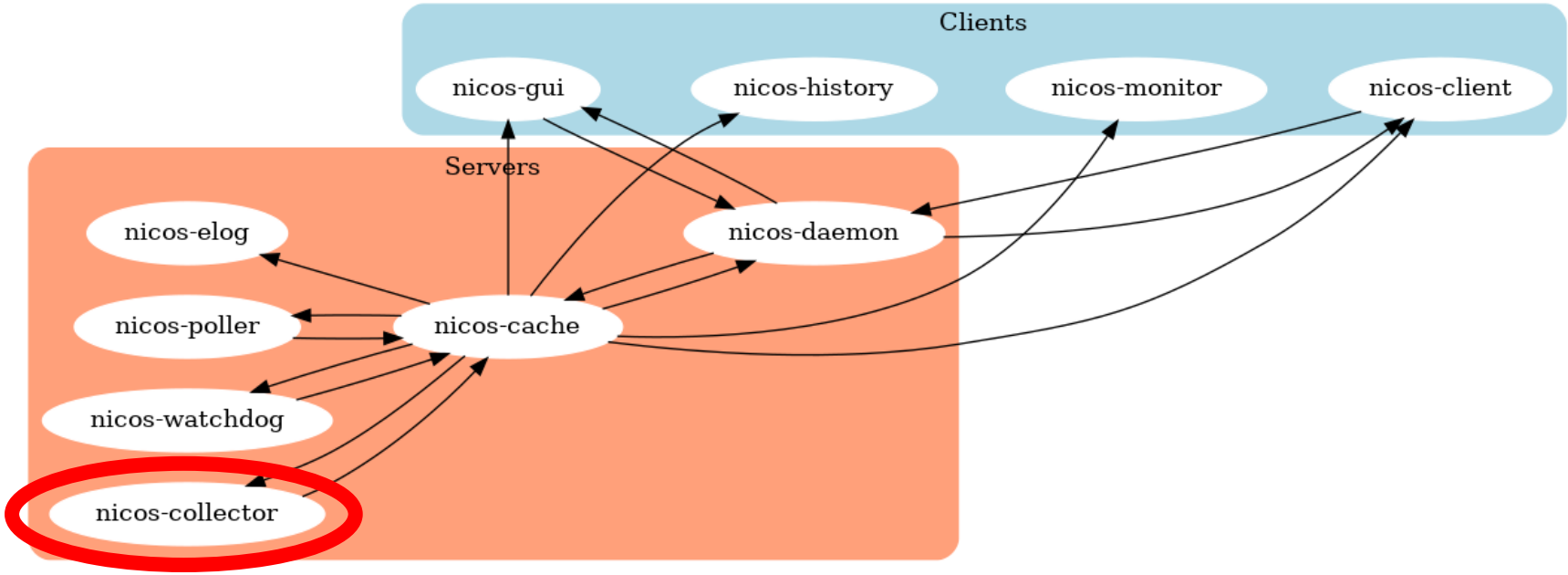
- Electronic logbook: experiment history in user friendly form (ASCII, HTML, external „ELN“, ...)

Components



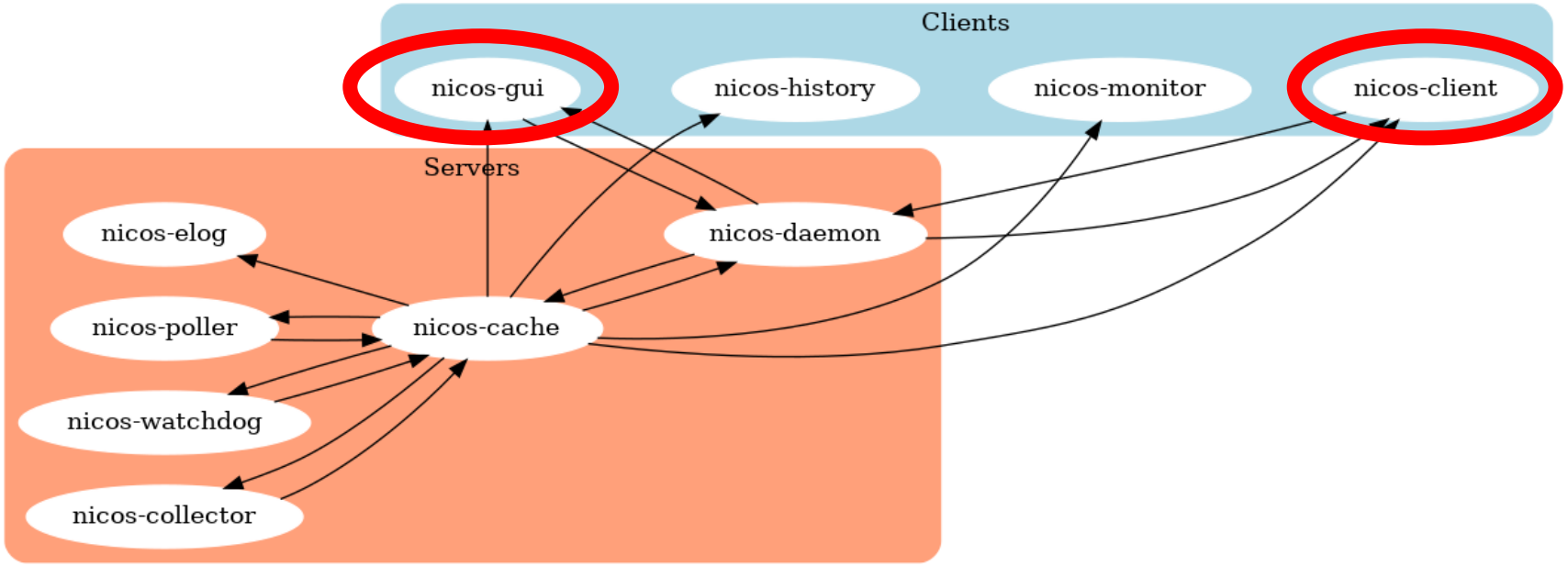
- Watchdog
  - Check for events and send out alerts (Email, SMS, Slack, …)
  - Configurable conditions

Components



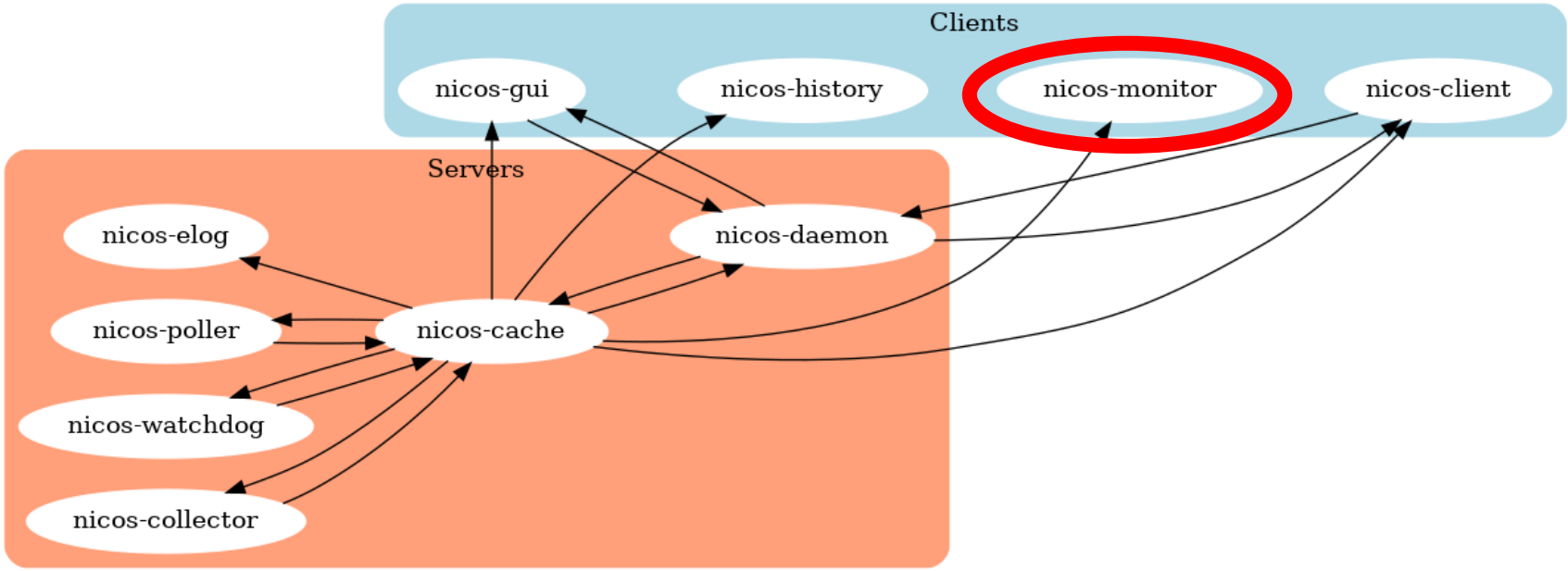
- Collector: transfer data to/from other cache(s)

Components



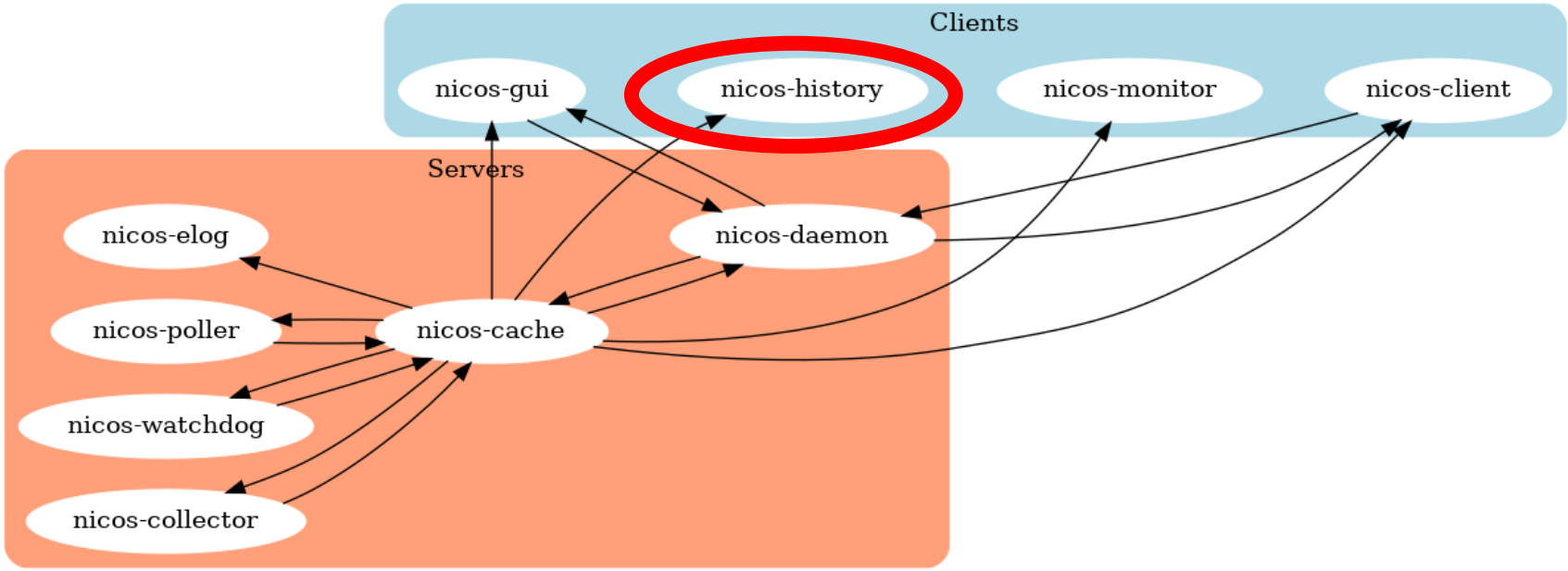
- GUI/client: user interaction

Components



- Monitor: Display status/value/parameter of devices
- Qt and HTML version

Components



- History: Display archived date depending on time

Development

Development

- Git as version control system

Development

- Git as version control system
- Test suite to catch regressions before integrate changes into source

Development

- Git as version control system
- Test suite to catch regressions before integrate changes into source
- CI with checking against test suite, static code checker, and setup check

Development

- Git as version control system
- Test suite to catch regressions before integrate changes into source
- CI with checking against test suite, static code checker, and setup check
- Code review system (Gerrit) to review, discuss, and improve changes

Development

- Git as version control system
- Test suite to catch regressions before integrate changes into source
- CI with checking against test suite, static code checker, and setup check
- Code review system (Gerrit) to review, discuss, and improve changes
- Code is freely available (GNU public license)

Development

- Git as version control system
- Test suite to catch regressions before integrate changes into source
- CI with checking against test suite, static code checker, and setup check
- Code review system (Gerrit) to review, discuss, and improve changes
- Code is freely available (GNU public license)
- Project page: https://www.nicos-controls.org with links to
  - Source repository
  - Issue tracker
  - Code review

Development

- Git as version control system
- Test suite to catch regressions before integrate changes into source
- CI with checking against test suite, static code checker, and setup check
- Code review system (Gerrit) to review, discuss, and improve changes
- Code is freely available (GNU public license)
- Project page: https://www.nicos-controls.org with links to
  - Source repository
  - Issue tracker
  - Code review
- Readonly copy on GitHub: https://github.com/mlz-ictrl/nicos

Some reasons to use NICOS at BNC

Some reasons to use NICOS at BNC

- NICOS is widely distributed and used at European Neutron Facilities
  - MLZ 25 instruments + 5 in preparation + several lab installations
  - PSI 16 running instruments
  - ESS ~10 ready to run installations (waiting for hardware and neutrons)

Some reasons to use NICOS at BNC

- NICOS is widely distributed and used at European Neutron Facilities
  - MLZ 25 instruments + 5 in preparation + several lab installations
  - PSI 16 running instruments
  - ESS ~10 ready to run installations (waiting for hardware and neutrons)
- NICOS is mature (more the 15 years in operation)

Some reasons to use NICOS at BNC

- NICOS is widely distributed and used at European Neutron Facilities
  - MLZ 25 instruments + 5 in preparation + several lab installations
  - PSI 16 running instruments
  - ESS ~10 ready to run installations (waiting for hardware and neutrons)
- NICOS is mature (more the 15 years in operation)
- User friendly
  - Science driven development
  - Developed from scientists for scientist
  - High user acceptance and satisfaction

Some reasons to use NICOS at BNC

- NICOS is widely distributed and used at European Neutron Facilities
  - MLZ 25 instruments + 5 in preparation + several lab installations
  - PSI 16 running instruments
  - ESS ~10 ready to run installations (waiting for hardware and neutrons)
- NICOS is mature (more the 15 years in operation)
- User friendly
  - Science driven development
  - Developed from scientists for scientist
  - High user acceptance and satisfaction
- Wide range of supported HW access layers
  - Tango, EPICS, SECoP, CARESS, …

Some reasons to use NICOS at BNC

- NICOS is widely distributed and used at European Neutron Facilities
  - MLZ 25 instruments + 5 in preparation + several lab installations
  - PSI 16 running instruments
  - ESS ~10 ready to run installations (waiting for hardware and neutrons)
- NICOS is mature (more the 15 years in operation)
- User friendly
  - Science driven development
  - Developed from scientists for scientist
  - High user acceptance and satisfaction
- Wide range of supported HW access layers
  - Tango, EPICS, SECoP, CARESS, …
- Open Source

NICOS in action