Research Data Management and Measurement Frameworks at KIT-ALFA (KARA & FLUTE)

Dr. Saeid Masoumi, Dr. Patrick Schreiber

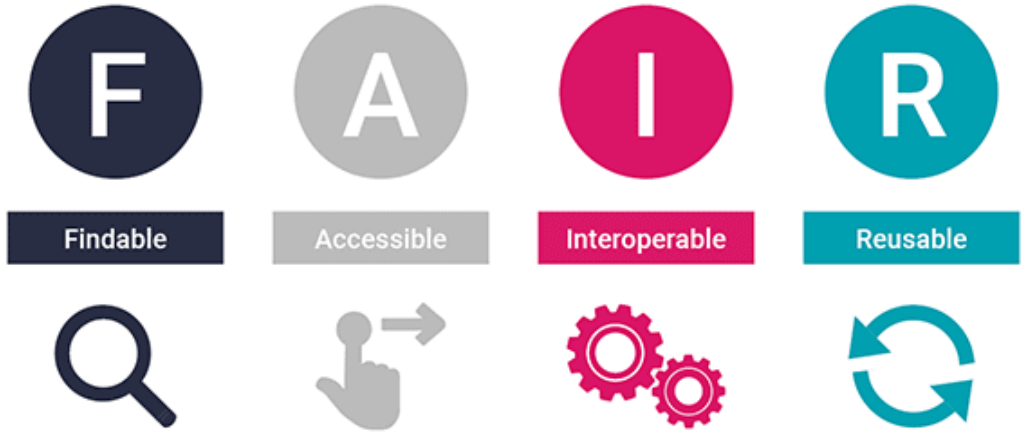
Research Data Management Framework

Data Management Framework at KIT-ALFA

- **FAIR** principles makes research data more useful and sustainable

- Findable → Data has rich metadata & unique identifiers
- Accessible → Data can be retrieved using standard and open protocols
- Interoperable → Data is stored standard formats & uses controlled vocabularies
- Reusable → Data is well-documented for future use

- Why FAIR matters:
  - Enhances reproducibility of research
  - Makes collaboration easier
  - Increases visibility and impact of data
  - Supports open science practices



Data Management Framework at KIT-ALFA

- **What we needed**
  - A customizable framework that facilitates good research data management (RDM) practices and collaboration between researchers
- **Why KADI4mat**
  - An open-source platform developed at KIT (Karlsruhe Institute of Technology).
  - Combines features of an Electronic Lab Notebook (ELN) and a data repository.
  - Originally designed for materials science but adaptable to various research disciplines.
- **Key Features:**
  - Data & Metadata Management: Organizes and describes research data comprehensively
  - Workflow Automation: Supports reproducible research through automated workflows
  - Collaboration Tools: Facilitates sharing and collaboration among researchers

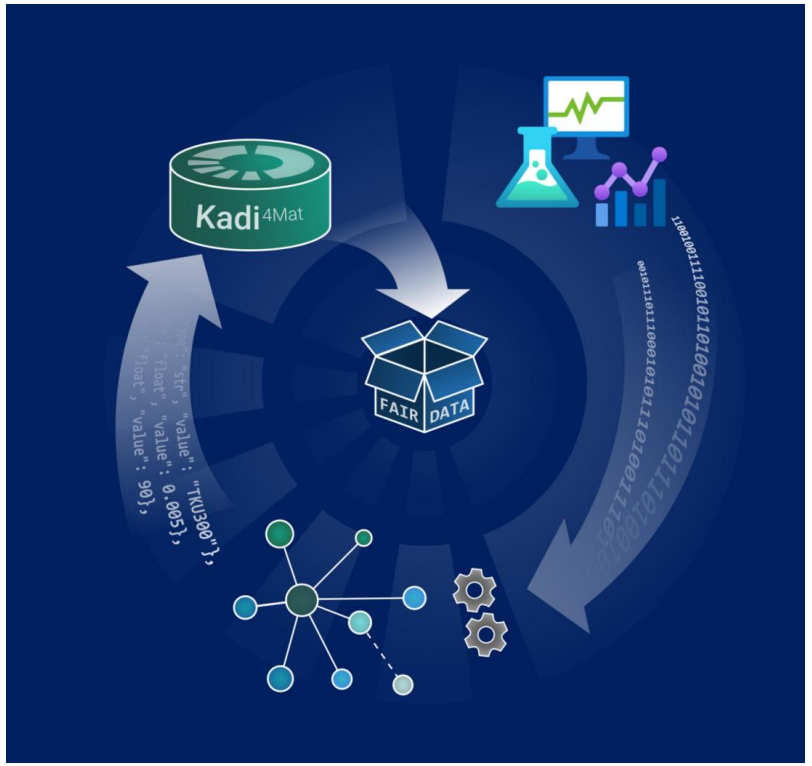
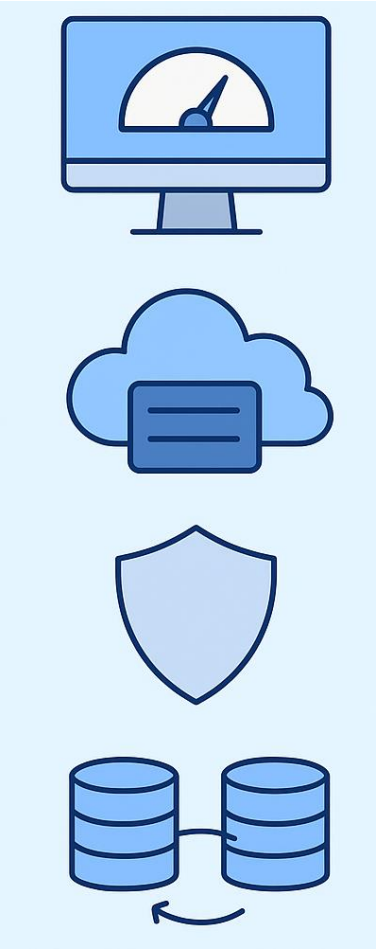


Image generated by Google Gemini, 2025

Key Improvements in Research Data Management

- **Comprehensive Maintenance Control**
  - Full oversight of system operations
- **Optimized Data Transfer**
  - Faster performance through fine-tuned KADI4mat configuration
- **Enhanced Security**
  - Stronger protection for data and users
- **Replicated Instances**
  - Dedicated KADI setups for partners and collaborators

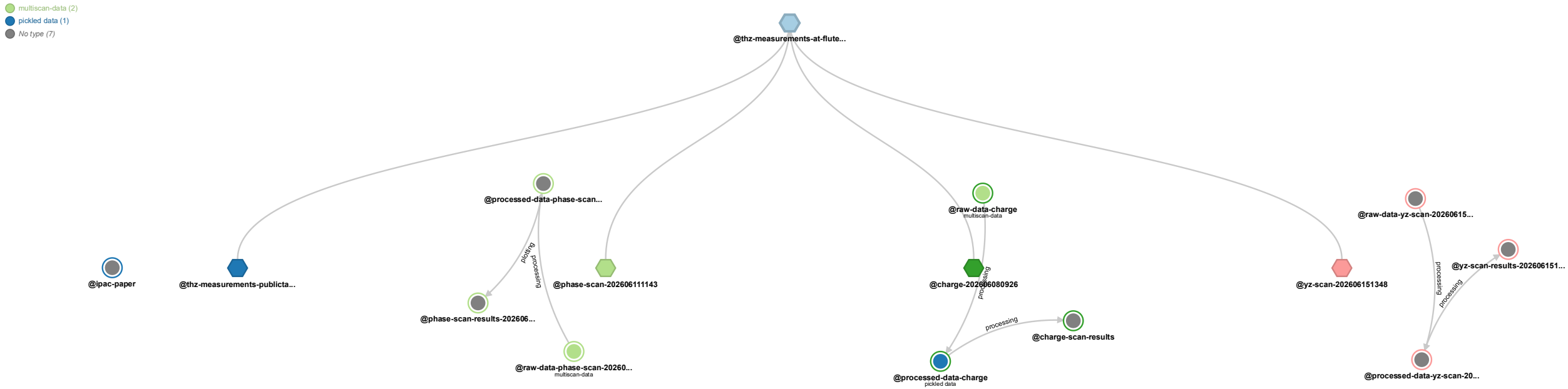


Core Concepts in KADI

- **Record**
  - Is a basic component of KADI4Mat
  - Represents any kind of digital or digitized object, such as research data, samples, or experimental devices
  - Consists of metadata
  - Can also be grouped into collections or linked to other records
- **Collection**
  - Represents a logical grouping of multiple records or even other collections
  - For example, projects, simulation studies, or experiments
- **Template**
  - Is used to define blueprints for creating different types of resources in a structured and consistent way.
- **User**
  - Represents an individual who works within the system
- **Group**
  - Is used to organize multiple users together,
  - Mainly to simplify access management and control permissions

Linking in KADI

- Record linking
  - Directly link a record with one or more records
  - Directly link a record with one or more collections
- Collection linking
  - Directly link a collection with one or more records.
  - Directly link this collection with a parent collection



Permissions

- Permissions are defined by assigning different roles to users or groups for records and collections
- **Roles in KADI:**
  - *Member role*
    - Allows a user/group to view a record/collection
  - *Collaborator role*
    - Read & Link
  - *Editor role*
    - Read & Link & Update
  - *Admin role (full access)*
    - Read & Link & Update & Permissions & Delete

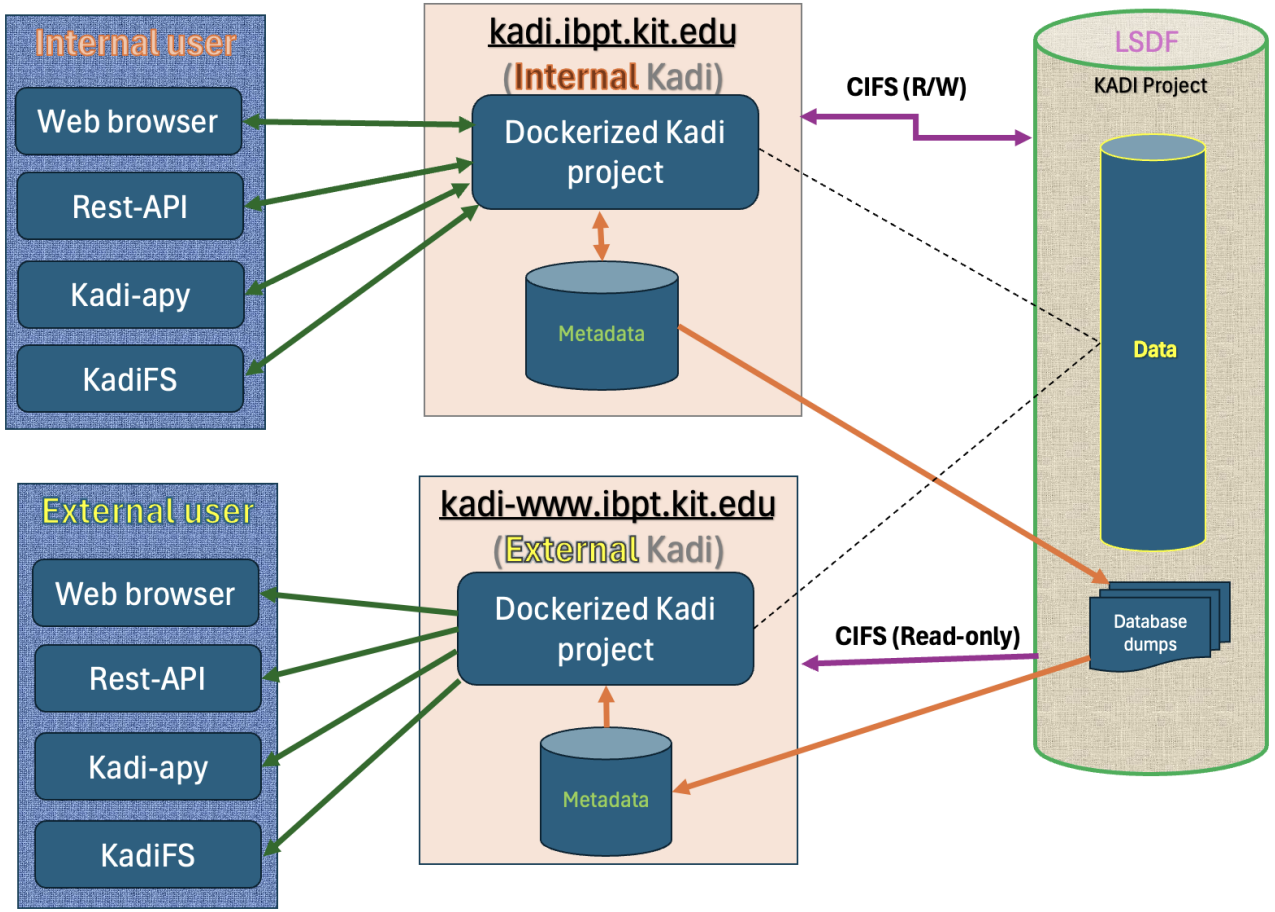
Our implementation

Kadi.ibpt.kit.edu (internal instance)

- Upload scientific data and metadata (R/W)
- connected to the **LSDF** (Large Scale Data Facility), for large datasets
- Internal login by KIT account
- Internal users see
  - Own data and public data
  - Grouping users is possible
- Private data can be open to a group of users
  - With different level of permission to users and groups

Kadi-www.ibpt.kit.edu (external instance)

- A read-only shadow copy of the internal one
- users cannot modify or delete data there
- External login by ORCID account
- External users see
  - Only public data
- User & group permission come from internal instance



KADI4MAT DEPLOYMENT WORKFLOW: IBPT / KIT

Secure and Automated Container Lifecycle

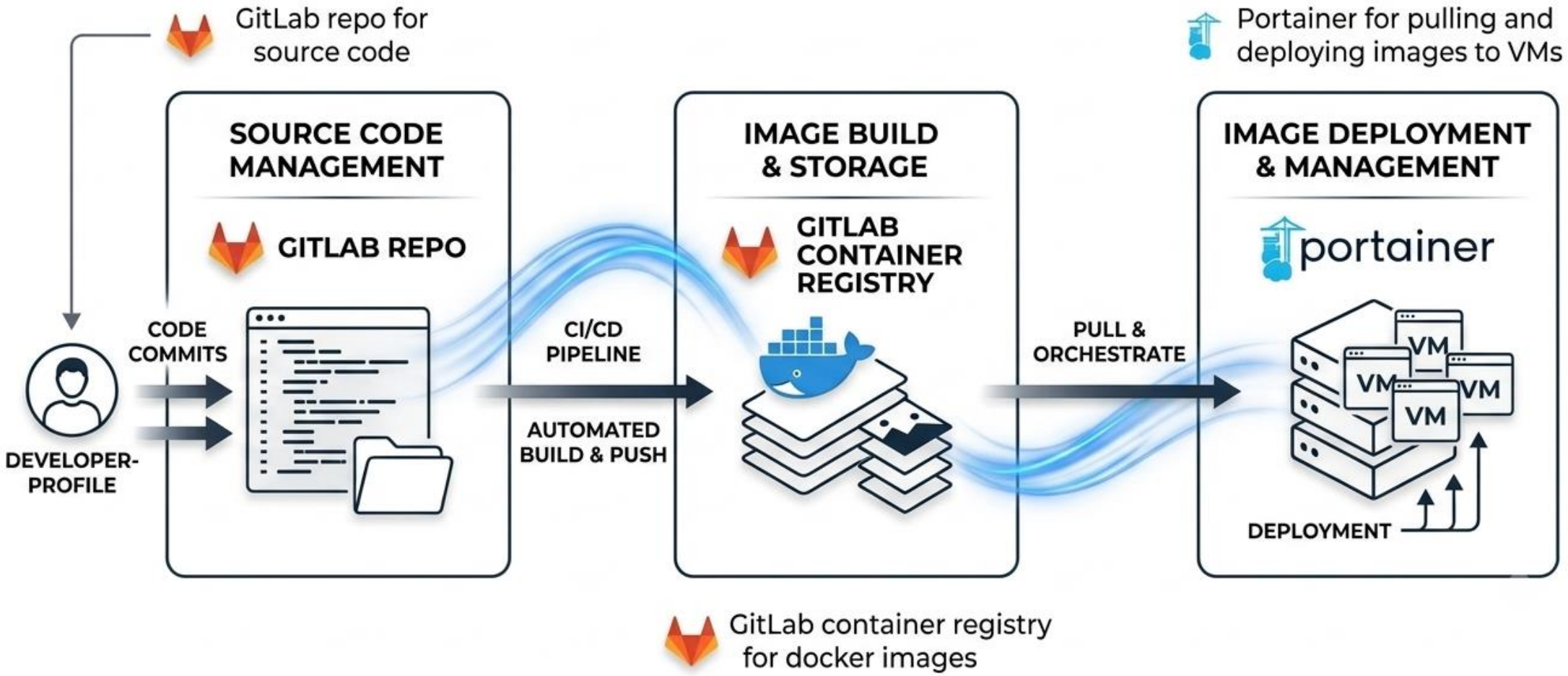


Image generated by Google Gemini, June 2025

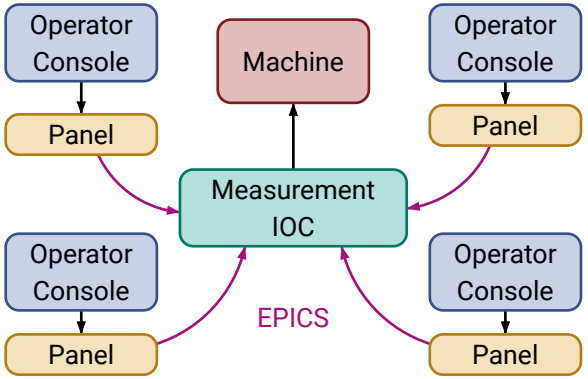
Measurement Framework

Measurement Framework

Motivation

- Measurements are important for operation and research at accelerator facilities
- Often dedicated scans
- Commonly implemented as scripts
  - By scientists, students or operators

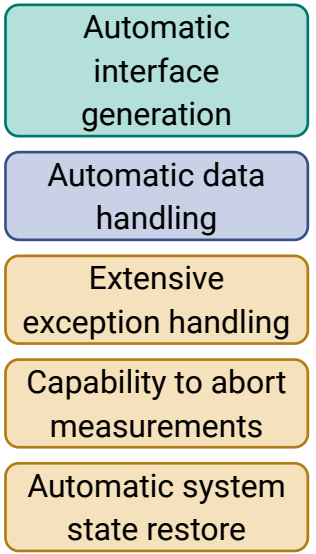
  ⇒ Sometimes hard to find and reuse
- Integration into control system would
  - Encourage reuse by integration in operator panels
  - Allow automatic result archiving
  - Provide a central access point from anywhere



Measurement Framework

Description

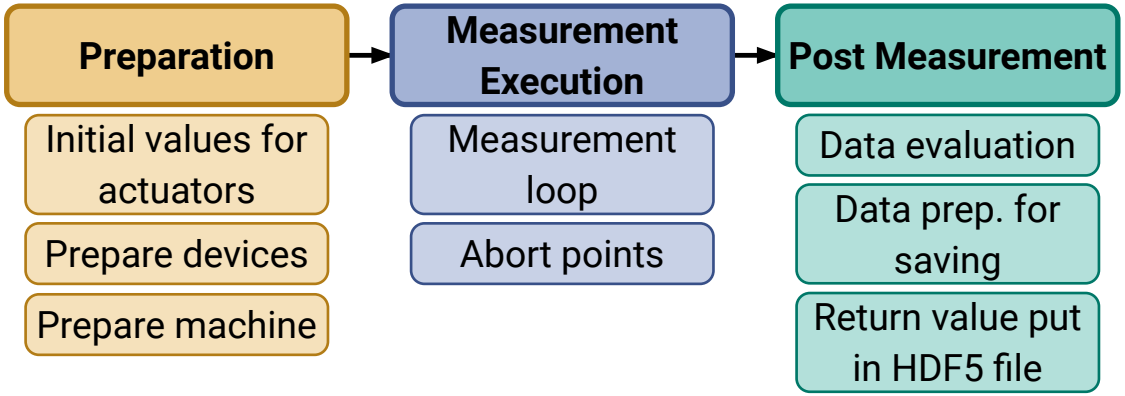
- Python framework
- Provides basic structure
- Automatically provides interfaces
  - GUI
  - Command line
  - Control system (EPICS) integration
- No expert knowledge required
- Automatic data and analysis result saving
- Extensive handling of exceptions
- Automatic reset of machine state after measurement or errors
- Capability to abort measurements



Measurement Framework

Structure

- Measurement divided in three parts
- Main loop freely implementable
- Dedicated data analysis routines foreseen
- Data is automatically saved including results of analysis



Interfaces

Configuration and Output Options

- Configuration parameters are specified with name, (Python) datatype, default value and help message
- Output values are specified the same
- Interfaces are built based on specified in- and outputs

Interfaces

Command Line

- Used as main entry-point
- Ability to run measurement including configuration
- Automatic parameter creation based on configured inputs
- Parameter description based on configured help text

```
$ python chromaticity.py -h
Usage: chromaticity.py [-h] [--simulation] {build-gui,gui,ioc,build-opi,execute,exec,run} ...

Options:
  -h, --help                show this help message and exit
  --simulation              Run the measurement in simulation mode, blocking Actuator Writes. (default: False)

Commands:
  The measurement can be controlled in various ways, use the subcommands

  {build-gui,gui,ioc,build-opi,execute,exec,run}
    build-gui               Build UI File for the GUI to modify. GUI can be used without building first
    gui                     Run a minimal gui for the measurement
    ioc                     Start a simple EPICS IOC for the measurement
    build-opi               Build a minimal CSS panel for the ioc
    execute (exec, run)     Configure and execute the measurement from commandline
```

Interfaces

Command Line

- Used as main entry-point
- Ability to run measurement including configuration
- Automatic parameter creation based on configured inputs
- Parameter description based on configured help text

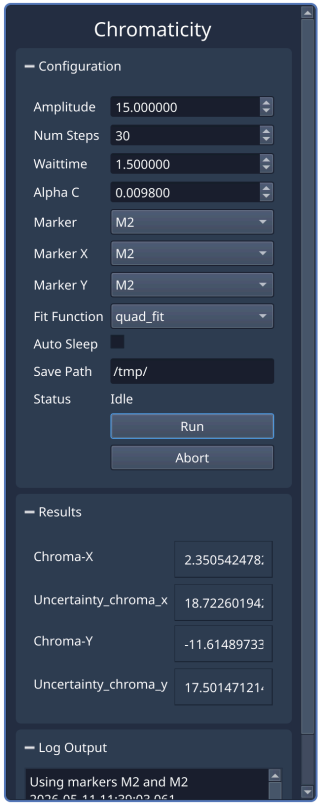
```
$ python chromaticity.py run -h
Usage: chromaticity.py execute [-h] [--amplitude <float>] [--num-steps <int>] [--waittime <float>] [--alpha-c <float>] [--marker {M1,M2,PhaseTracker,Individual}] [--marker-x {M1,M2,PhaseTracker}]
                               [--marker-y {M1,M2,PhaseTracker}] [--fit-function {quad_fit,lin_fit,cubic_fit}] [--auto-sleep] [--save-path <str>]

Options:
  -h, --help                  show this help message and exit
  --amplitude <float>         Maximum amplitude in kHz (unit: kHz) (default: 5)
  --num-steps <int>           Absolute number of steps to take (default: 15)
  --waittime <float>          Time to wait after setting RF frequency before reading data (unit: s) (default: 3)
  --alpha-c <float>           Momentum compaction factor. A value of 0 means 'get from machine pv' (default: 0.0098)
  --marker {M1,M2,PhaseTracker,Individual}
                              Which marker to use. Setting this will override x_marker and y_marker (default: M2)
  --marker-x {M1,M2,PhaseTracker}
                              Which marker to use for X (default: M2)
  --marker-y {M1,M2,PhaseTracker}
                              Which marker to use for Y (default: M2)
  --fit-function {quad_fit,lin_fit,cubic_fit}
                              Type of fit. (default: quad_fit)
  --auto-sleep                If set, ignore waittime, sleep based on num_steps, amplitude, speed of RF (default: False)
  --save-path <str>           Path to where the resulting hdf5 files will be saved (default: /tmp/)
```

Interfaces

GUI

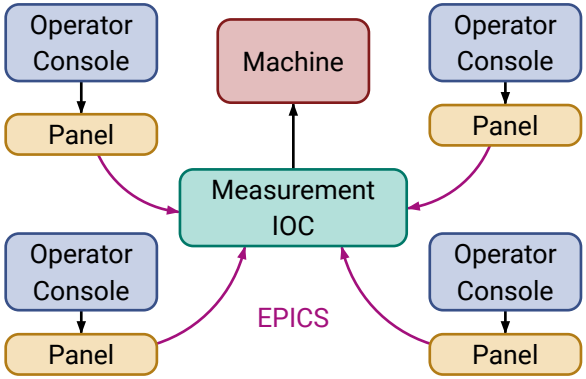
- Standalone GUI
- Single or multiple measurement executions
- Auto generated without additional code



Interfaces

Epics IOC

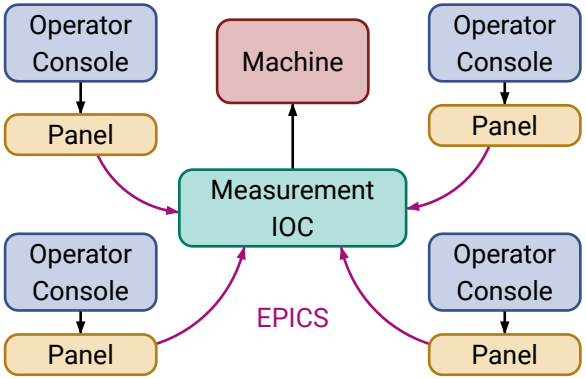
- Integration into the EPICS control system
- Centralised measurement execution
- Accessible from anywhere
- Findable
- Auto generated without the need for knowledge of EPICS and IOCs
- Translates configured in- and outputs to PVs
- Automatic panel generation

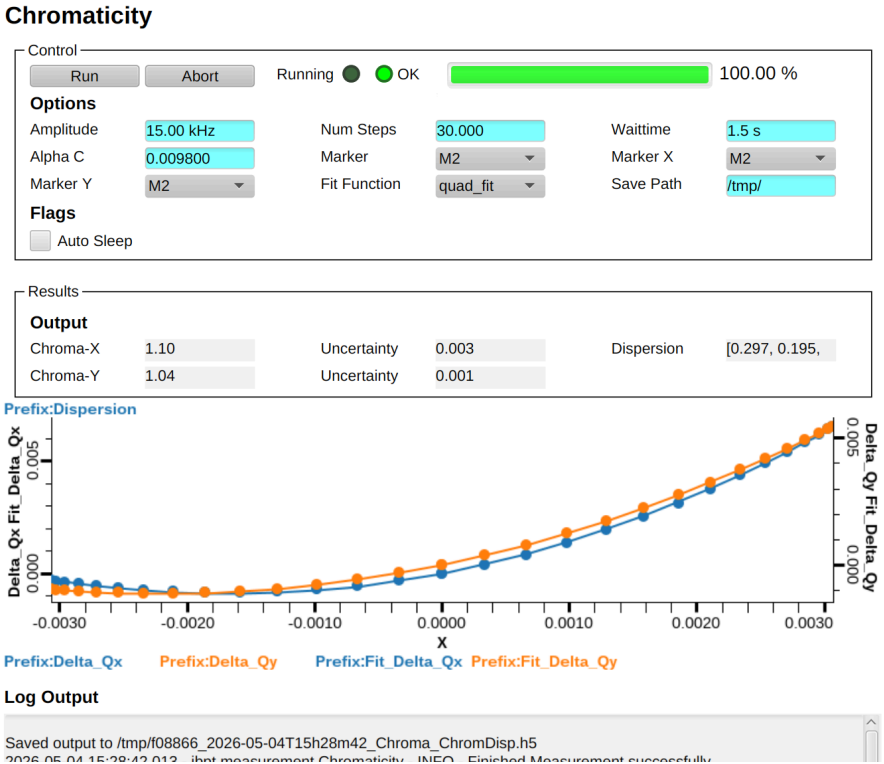


Interfaces

Panels

- Panels for CSS and CSS-Pheobus can be auto generated
- No additional code required





Summary

- Python framework targeted at scientists/students/operators
- No knowledge of interfaces and control system required
- Simple structure
- Automatic interface (Command line, GUI, EPICS IOC+Panel) generation
- Safeguards in case of exceptions
- Data handling included

Code Example

Minimal Measurement Implementation

```python
class Meas(Measurement, Configurable):
  def __init__(self):
    super().__init__("Meas")
    self.setup_config({
      "iterations":
       CO(type=int, default=5)
    })
    self.set_monitors([
      EpicsMonitor("Mon", "Some:PV")
    ])
    self.set_actuators([
      EpicsActuator("A:PV", name="act")
    ])
```

```python
def prepare(self):
  epics.caput("S:PV", 3)
def measure(self):
  num = self.config.iterations
  for i in range(num):
    self.put_actuator("act", 3)
    self.sleep(1, abortable=True)
    self.read_data()

def post_measure(self):
   f = self.data.get("Mon")**2
   return { "q": f**2 }
```