PAUL SCHERRER INSTITUT

PSI

Introduction to ooPye for soft IOC Implementation

Zheqiao Geng

07.12.2017

PAUL SCHERRER INSTITUT

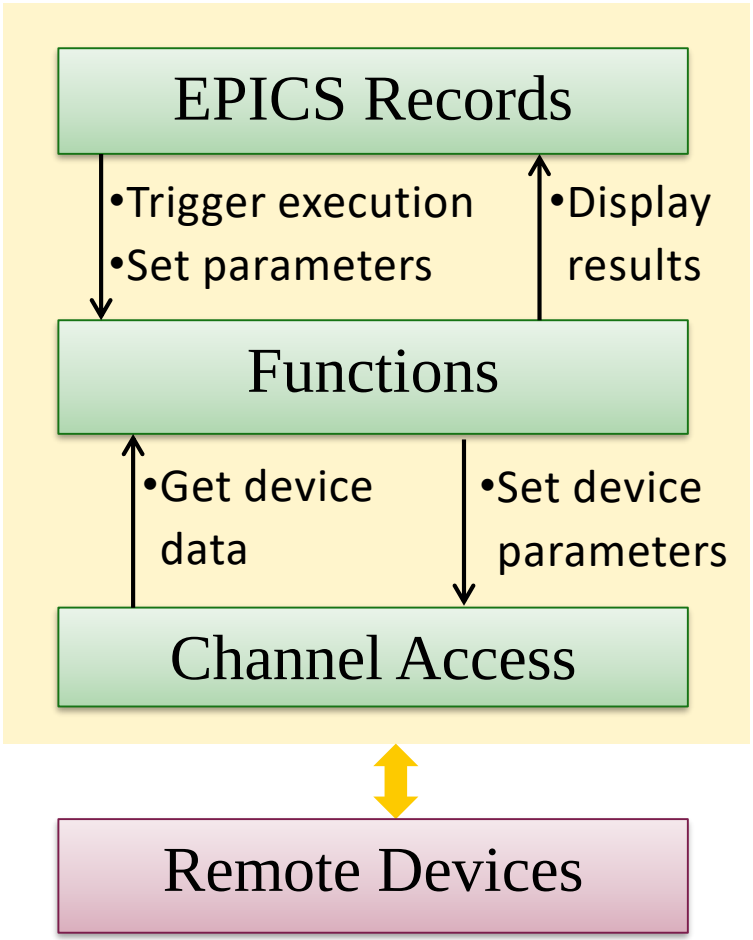
Outline

- Motivation
- Software architecture behind ooPye
- ooPye classes and Python
- Code example

PAUL SCHERRER INSTITUT

Motivation

A Soft IOC



- Soft IOC access EPICS channels of devices instead of directly accessing hardware
- Soft IOC defines its own EPICS records to accept commands, settings and display results

Functions of a soft IOC:

- Not real-time
- Usually interacts with multiple remote devices
- Domain algorithm intensive for setup, calibrations and optimizations
- Automation procedures

Tools to Implement Soft IOC

- Conventional EPICS IOC development procedure
  - Manually define EPICS database to hold the commands, parameters and results of the soft IOC functions
  - Implement EPICS device support in C language for soft records to realize the soft IOC functions
- Sequencer and the Sequencer Notation Language (SNL)
  - Manually define EPICS database
  - All functions of soft IOC are implemented as a Finite State Machine
  - SNL is a custom language similar as C
- Soft IOC fully implemented in Python (need compile with EPICS base and Python C library)
  - https://github.com/Araneidae/pythonIoc/tree/master/softIocApp

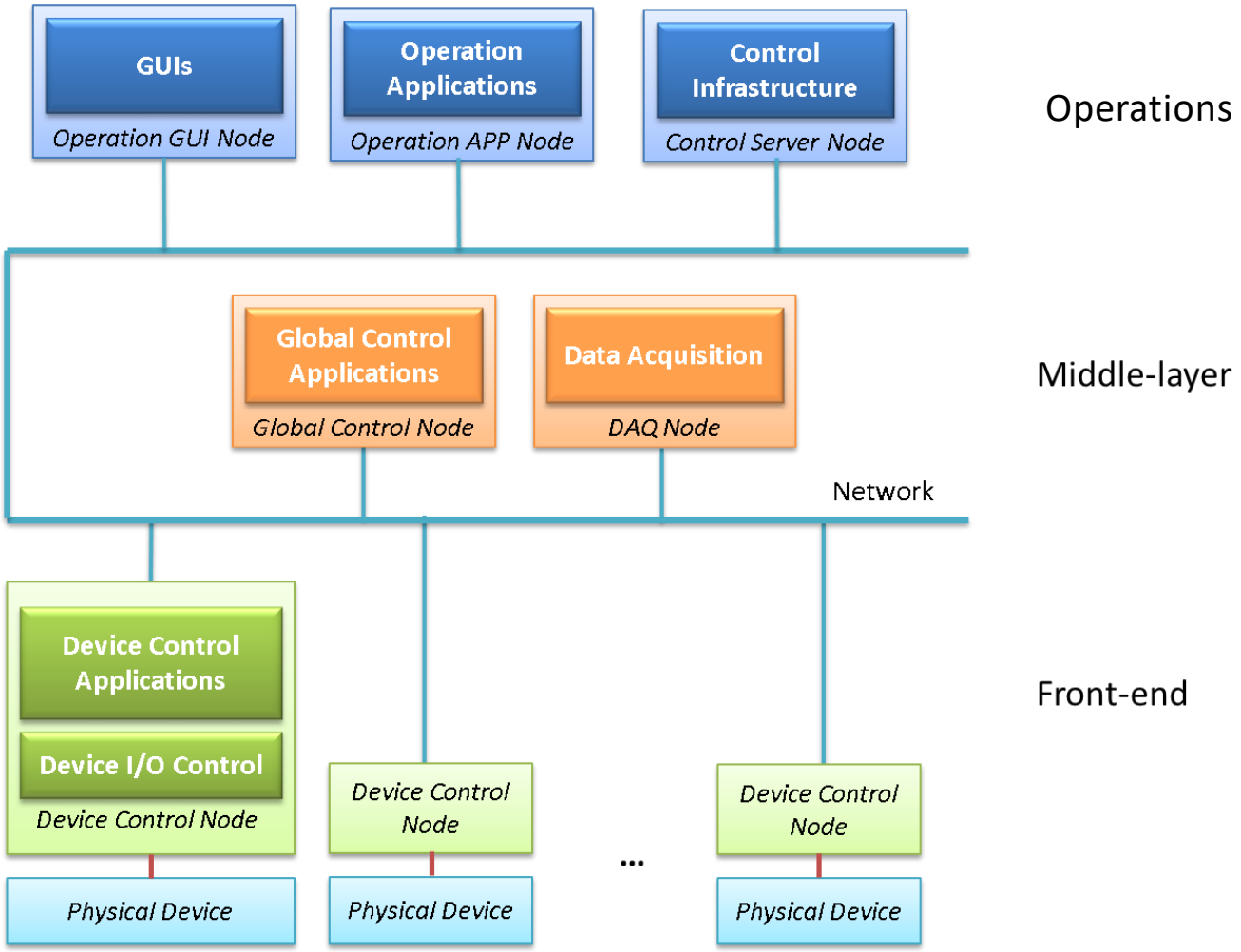
My Expectation for a Soft IOC Tool

- Fully Python based, no compilation against EPICS base needed. Use as much as possible the environment provided by the controls
- Automatic generation of EPICS database and the soft IOC, hide details of EPICS for programming
- A sort of pattern of software architecture that can be applied on many different systems

PAUL SCHERRER INSTITUT

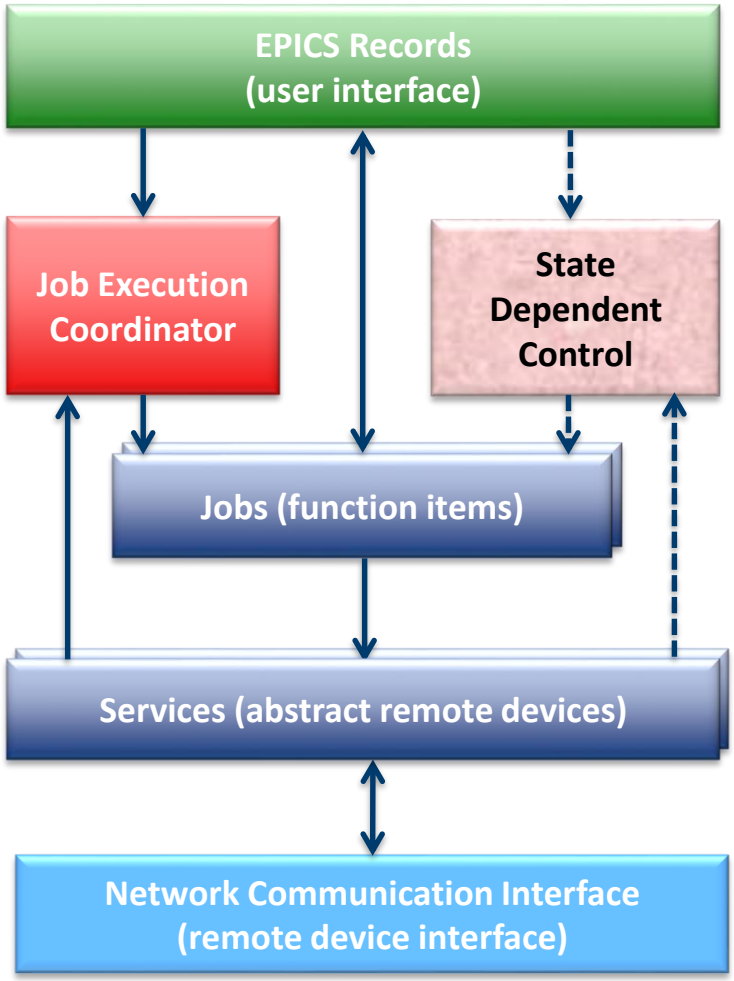
Software architecture behind ooPye

EPICS based control system



The ooEpics framework will focus on the development of front-end and middle-layer components. While the ooPye will focus on the **operation applications**.

General Architecture of Operation APP

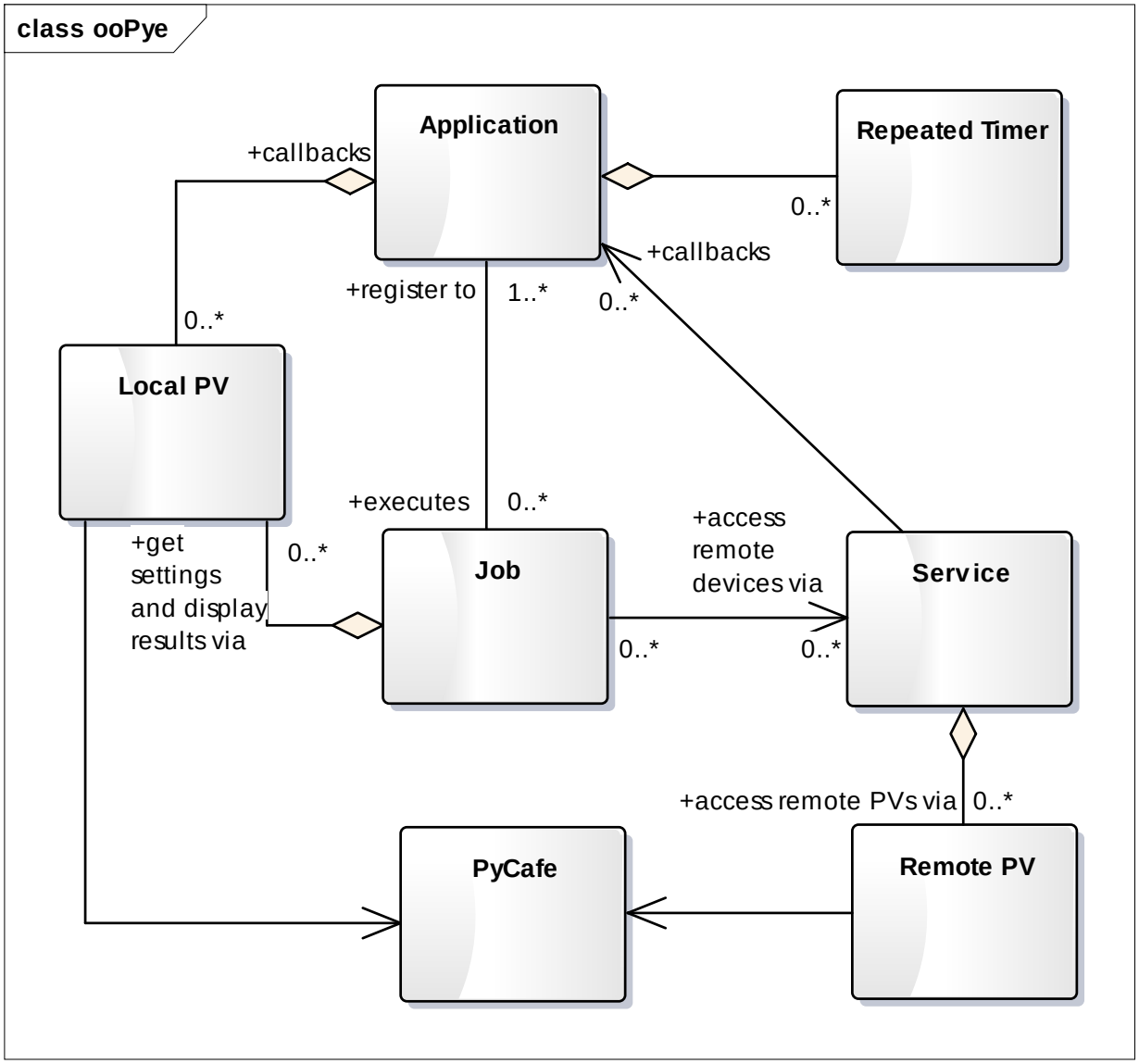


- The “Job Execution Coordinator” and “State Dependent Control” are active objects mostly driven by threads
- The “Job” is an abstraction of a function item that need to be executed (e.g. a procedure to calibrate the beam energy)
- A “Service” is an abstraction of an external device which needs to be monitored or controlled
- The “Network Communication Interface” is used to communicate with the controllers of remote devices (e.g. Channel Access client)

PAUL SCHERRER INSTITUT

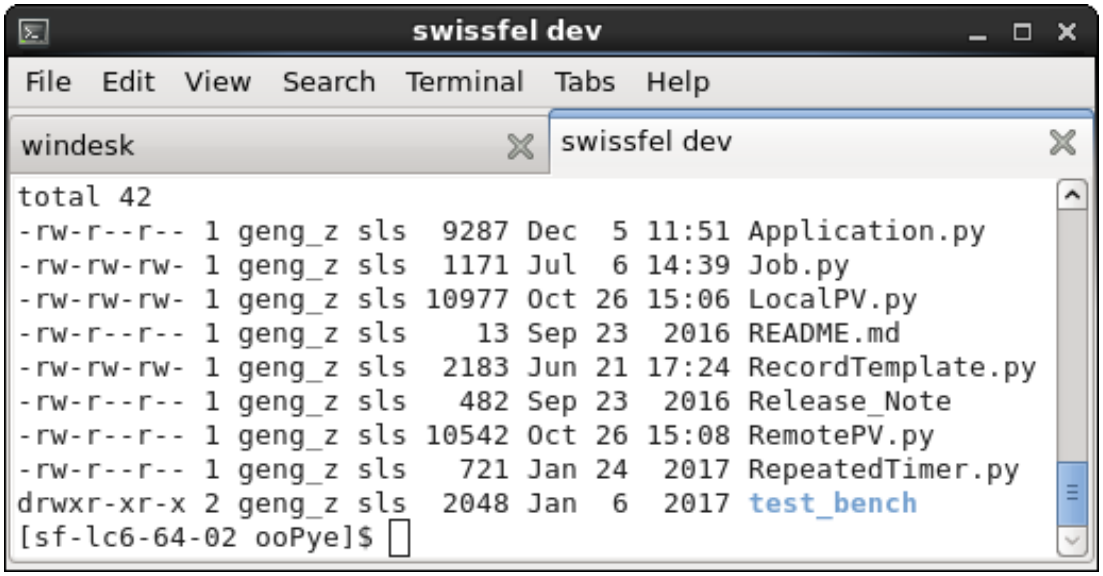
ooPye Classes and Python

Python Basic Classes of ooPye



- Base class “Service” does not exist in the ooPye package. But it is an essential class that should be implemented in custom projects.

Python Basic Classes of ooPye (cont.)



- The ooPye package can be found in git repository:

  https://git.psi.ch/llrf/ooPye

- The example soft IOC project based on ooPye can be found in git repository:

  https://git.psi.ch/llrf/ooPyeGen

- The latest code of ooPye are installed in the AFS folder:

  /sf/rf/tools/ooPye/

Procedure to Implement a Soft IOC with ooPye

- Implement Service classes to access the devices that the soft IOC will monitor or control. A good practice is to define for each physical device a corresponding Service class. In the Service classes, Remote PV objects can be created to access the PVs in the device controllers.
- Implement Job classes (as derived classes of the base class in ooPye) to do measurement, analysis, calibration and optimization. The data can be collected from the devices via the Service objects and the device parameters can be set.
- Define an object of Application and register your job objects to it.
- Edit the soft IOC installation Python code to define the soft IOC name and the necessary strings to construct the PV names.
- Install the soft IOC by executing the installation Python code. Before that, you need to follow the standard procedure defined by controls to create a soft IOC on the soft IOC server.

Some Words about Python

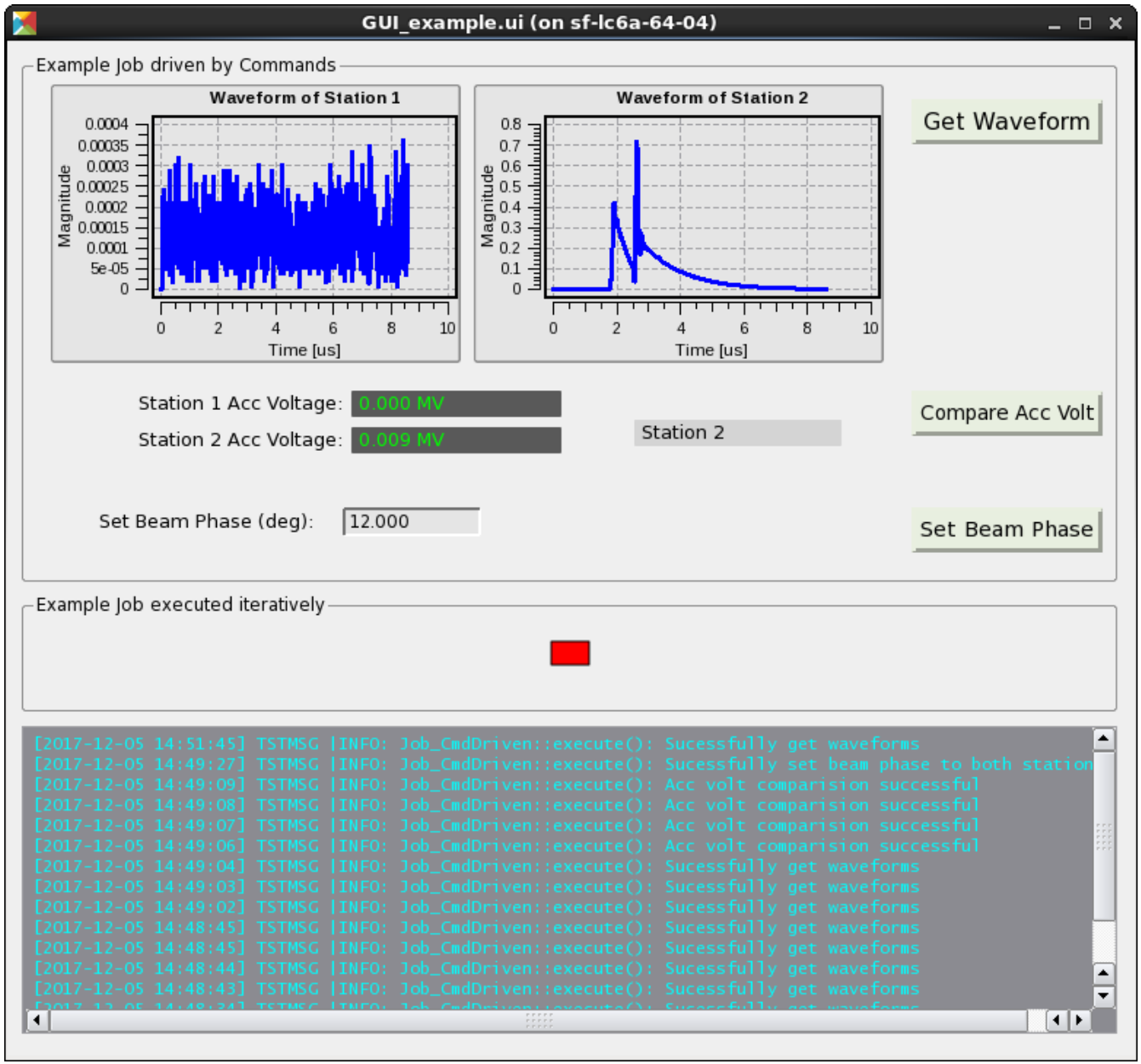
- Python is a very popular interpreted programming language. It is more than a scripting language and can be used to build large scale software:
  - Object oriented or structured programming are both fully supported.
  - Multi-thread programming supported.
  - Rich libraries for mathematics, scientific computation, matlab-like plotting …
- Python is an easy language more close to oral speaking. It can implement complex functions with less lines compared to C language
- Lots of other benefit …. You can construct your software much faster than programming C-like languages

PAUL SCHERRER INSTITUT

PSI

Code Example – ooPyeGen

GUI of Example Code



PAUL SCHERRER INSTITUT

PSI

Questions?