High Level Applications for commissioning & operation

8th Hard X-Ray Collaboration Meeting
Pohang, Korea, 24-26 October 2016

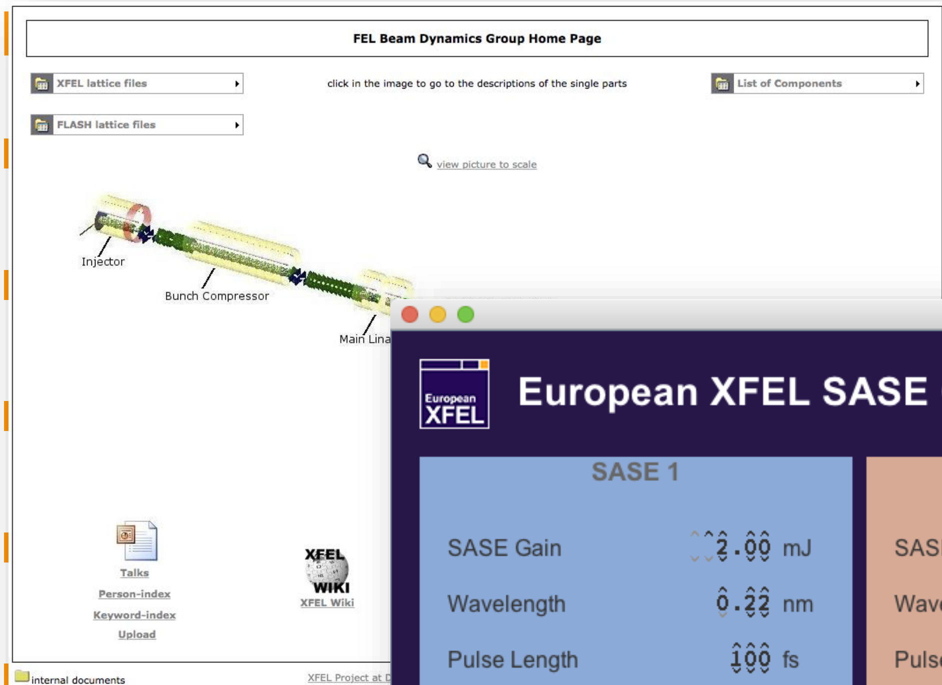
Raimund Kammering

Outline

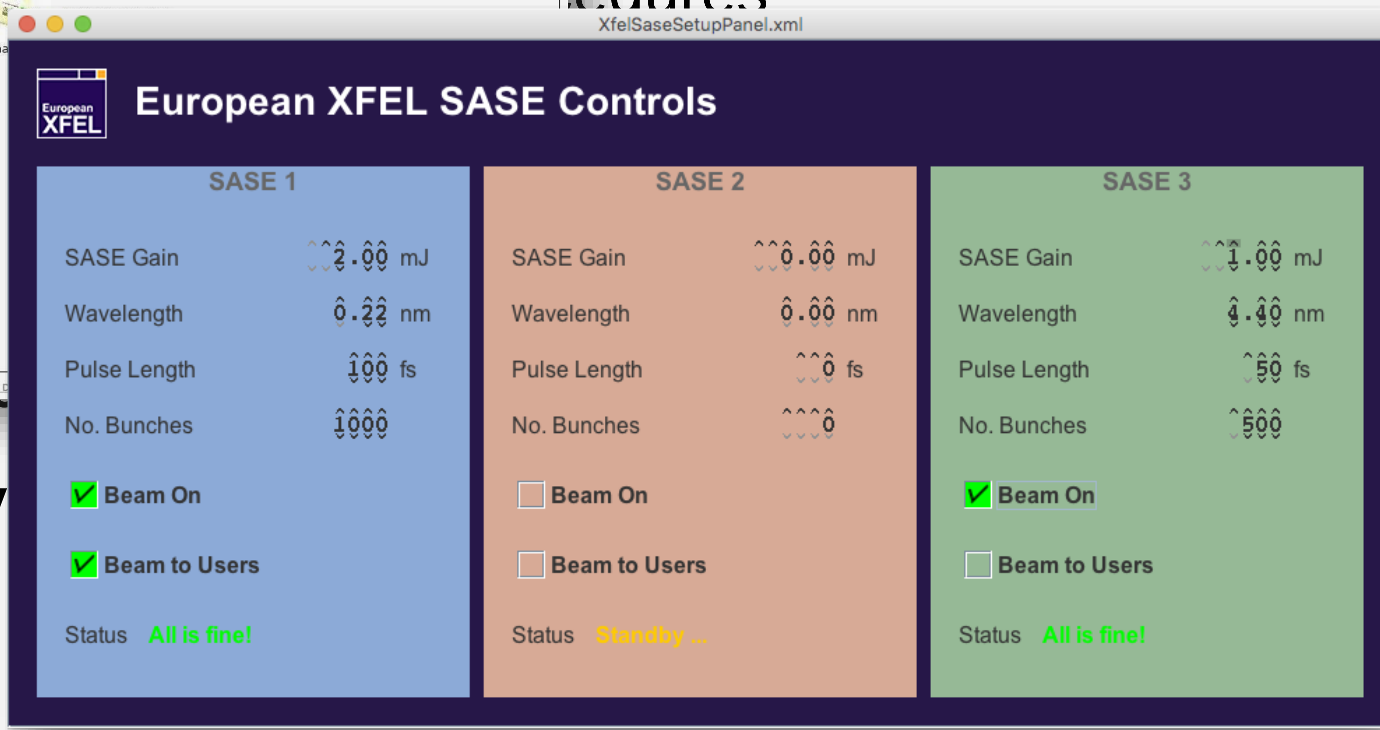
- **Intro/Motivation**
  - High Level Applications – what's meant and what for?
  - High Level Software (HLS) Developments @ EU-XFEL
- **Control Systems and HLS @ the EU-XFEL**
  - The Accelerator Control System (CS)
    - → Overall Architecture
    - → The DAQ System
  - HLS – a topological view
  - HLS – examples
  - The Virtual XFEL
- **Outlook**
  - Machine optimization using HLS
  - Big Data

Intro/Motivation – the dream

Intro/Motivation – the task



…cedures



availability

- …

Int

Self-Driving Tesla Was Involved in Fatal Crash, U.S. Says

By BILL VLASIC and NEAL E. BOUDETTE JUNE 30, 2016

" … belief that computers can operate a vehicle more safely than human drivers."

DETROIT — The race by automakers and technology firms to develop self-driving cars has been fueled by the belief that computers can operate a vehicle more safely than human drivers.

European XFEL

Intro/Motivation – the team

European XFEL Commissioning

Lessons learned from LCLS & FLASH

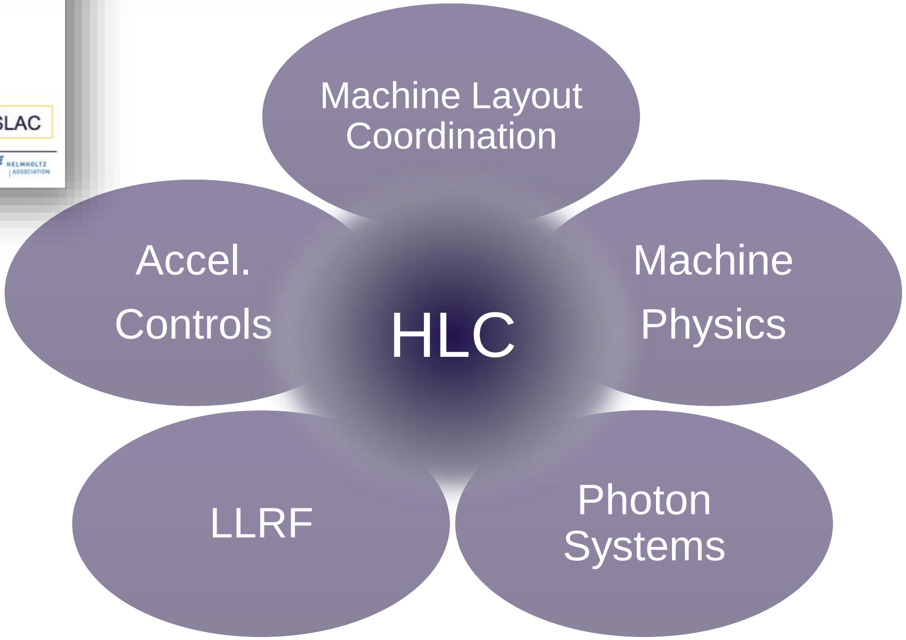
Thorough Pre-beam Checkout: Detailed pre-beam checkout conducted from the actual controls panels and verified in the tunnel with Hall probes, clip-on ammeters, simple tape measures, etc. ensures that the hardware and software is functioning at a fundamental level prior to beam.

Software Ready for Commissioning: Process automation and data analysis software fully prepared ahead of time to support commissioning activities

Reliable Diagnostics: Reliable diagnostics for all (or almost all) electron beam parameters throughout.

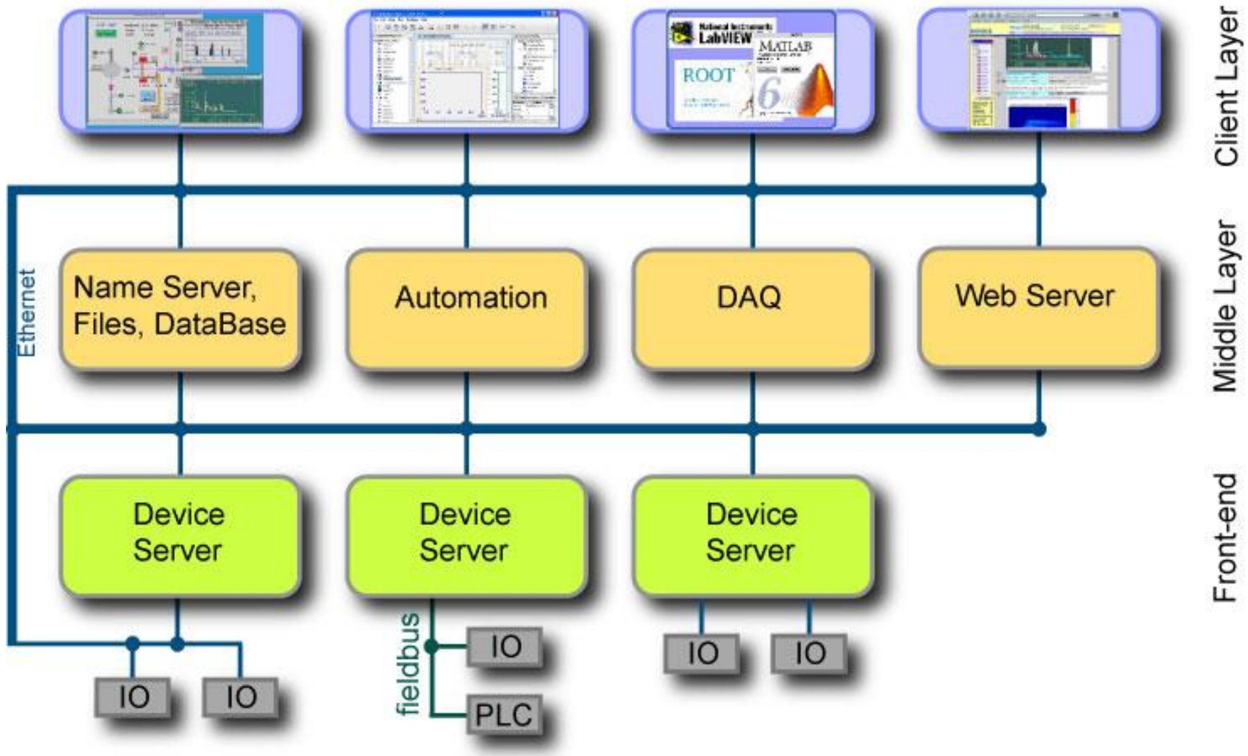
courtesy Jim Welch, SLAC

XFEL Collaboration Meeting, 08.04.2014
Winfried Decking, DESY



CS and HLS @ the EU-XFEL – Architecture

- Dominate Accelerator CS is DOOCS
- Classical three layer architecture
- Emphasize in respect to HLS on middle layer (→DAQ)



CS and HLS @ the EU-XFEL – Architecture

jddd / Karabo-GUI

Generic user interface builders/engines

- jddd: Java
- Karabo-GUI: Python 3.4 / QT

Applications

(Rich Clients)

- Matlab
- Python 3.4 / QT
- Java

Control Systems / Protocols

DOOCS, TINE, EPICS, Karabo

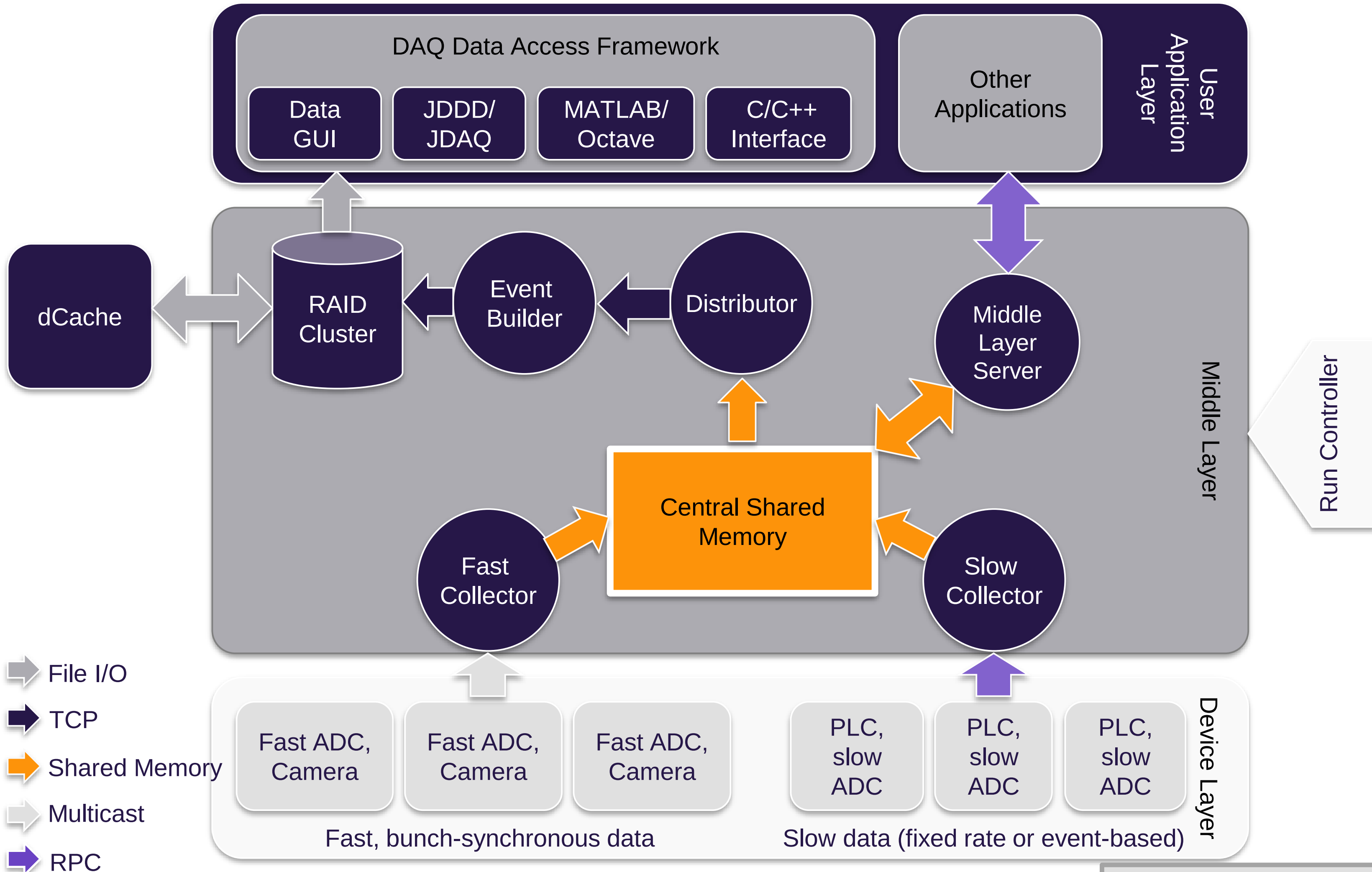
Middle Layer Servers

C++ (03, 11, 14)

Frontend Servers

C++ (03)

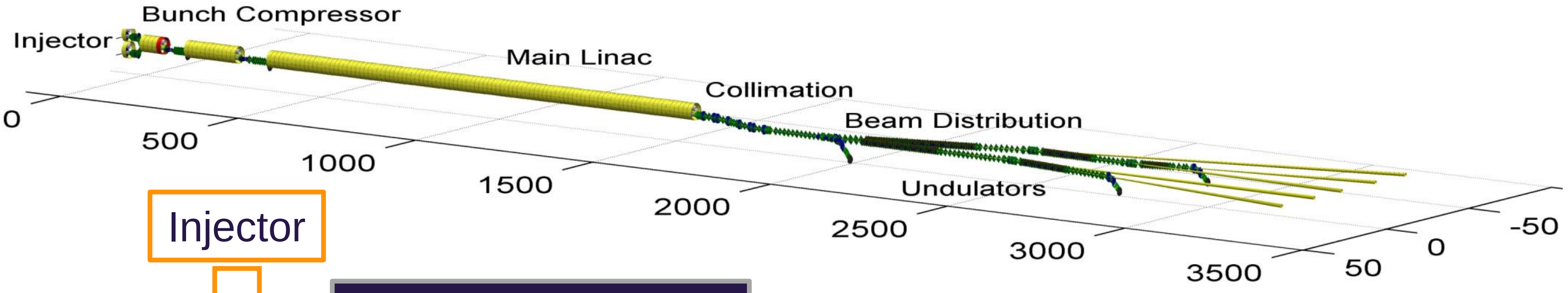
CS and HLS @ the EU-XFEL – the DAQ



Courtesy: T. Wilksen (DESY)

European XFEL

CS and HLS @ the EU-XFEL – topological view



Injector

- Timing, MPS
- Bunch Pattern Server
- …
- Orbit displays, ORM
- Laser heater
- Feedbacks
- ...
- Emmitance ...

Bunch Compression

- Energy measurements
- Chicane setup
- Longitudinal Diagnostics
- TDS
- RF Tweak
- …

Main Linac

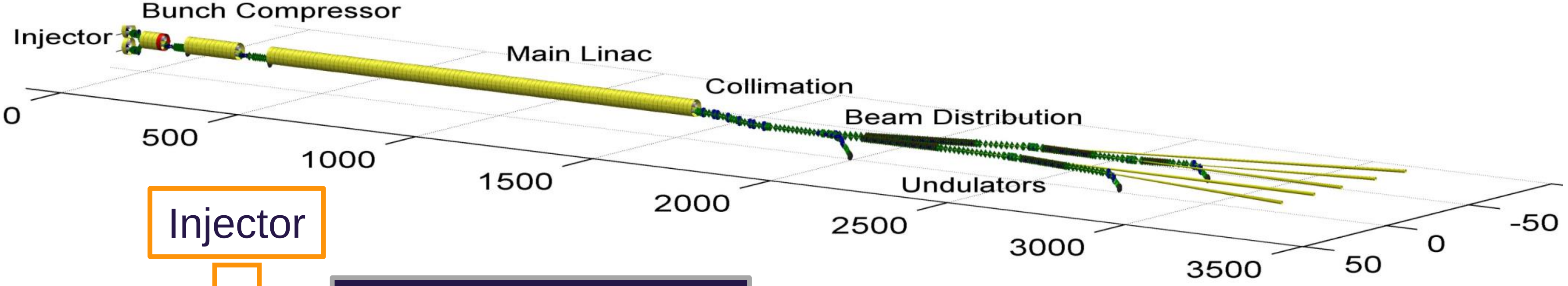
- Energy Management
- Interfaces to cryogenic
- ...

Undulators

- Kicker control
- Wavelength control, tappering, ...
- GMDs
- ...

European XFEL

CS and HLS @ the EU-XFEL – topological view



Injector

Extensively covered by Matthias S.

Bunch Compression

some words on **magnets**

Main Linac

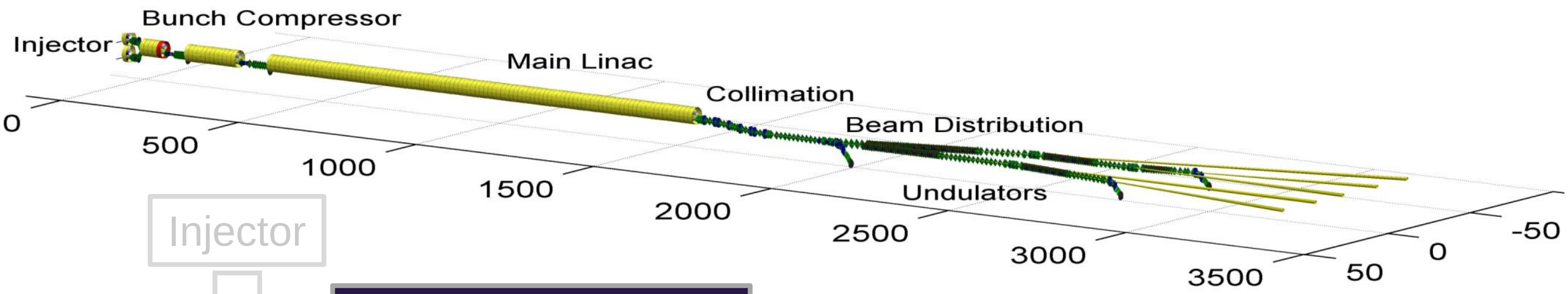
will have a look at the **energy management**

Undulators

all parts exist should just work

DESY HELMHOLTZ ASSOCIATION

CS and HLS @ the EU-XFEL – HLS examples



Injector

Extensively covered by Matthias S.

Bunch Compression

some words on **magnets**

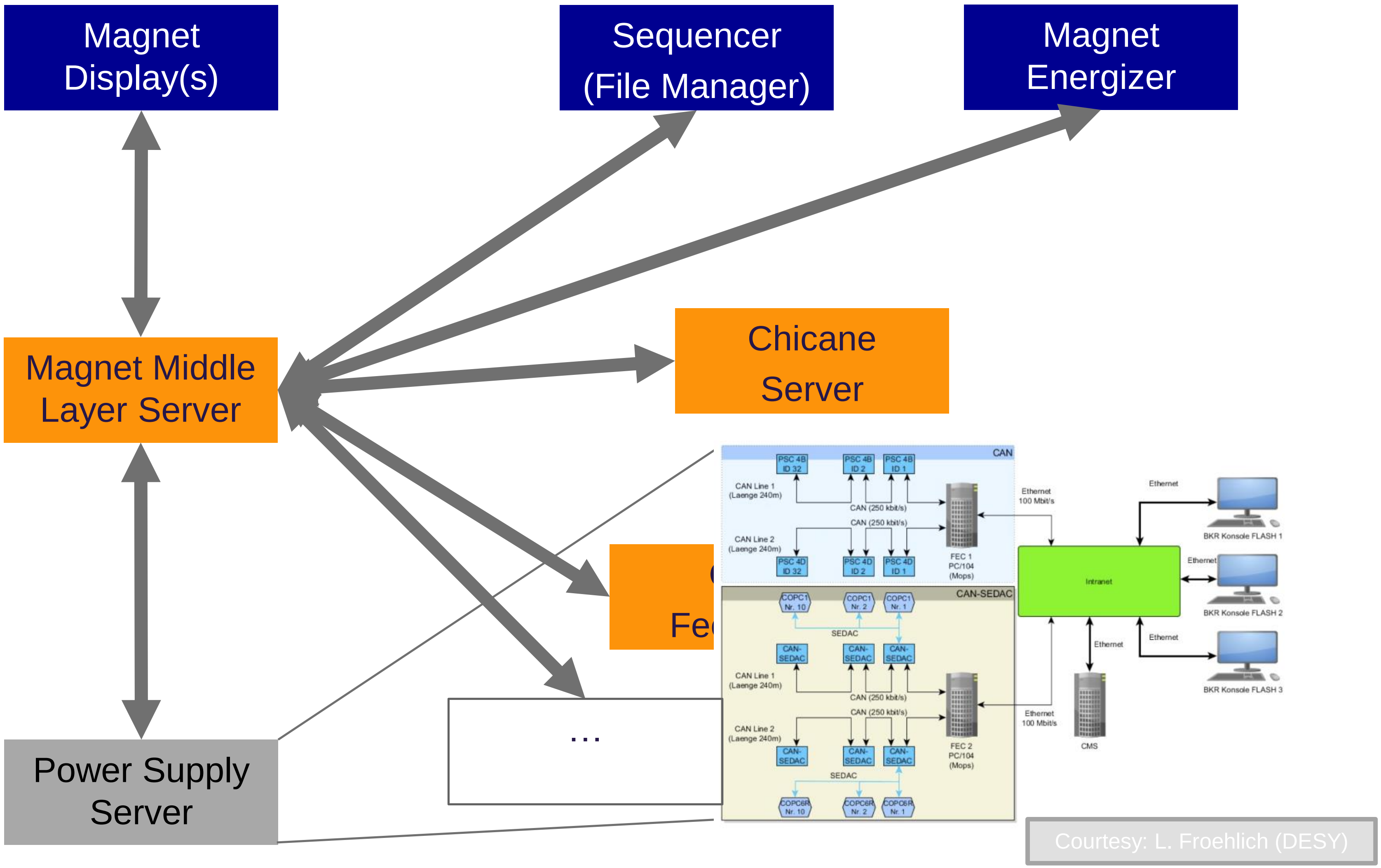
Main Linac

will have a look at the **energy management**

Undulators

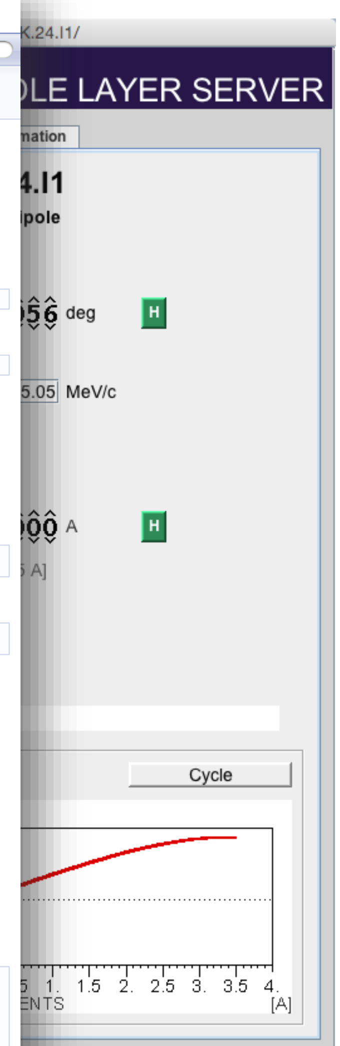
all parts exist see **outlook**

HLS examples – Magnet Middle Layer



HLS examples – Magnet Middle Layer

- Magnets … power su…
- Physical … instead …
- Hysteres… keep tra… branche…
- Cycling
- Group h…
- Databas…



High Level Controls Machine Library 18.10.9

Main Page | Namespaces | Classes | Files | Examples | Search

High Level Controls Machine Library Documentation

The High Level Controls Machine Library contains several classes for easy access to components and parameters of the various supported accelerators (currently FLASH, XFEL, VXFEL).All functions and classes are found in namespace hlc.

To use the library, include the single header file hlc_machine.h and link your code against this library and the HLC utility library (-lhlc_machine -lhlc_util).

Usage

The application typically creates a single instance of the hlc::Machine class. This can be done by constructing one of its specializations (hlc::Flash, hlc::Xfel, or hlc::VirtualXfel) directly:

```
hlc::Xfel machine;
```

Alternatively, the hlc::get_machine() function can be used to create a hlc::Machine object specialized according to a string parameter:

```
auto machine = hlc::get_machine("/XFEL.DIAG/BLUB")
```

The function uses some heuristics to infer the machine from the string; typical control system addresses work just as well as simple strings like "FLASH" or "VXFEL".

Once a machine object has been obtained, it can be used to construct objects related to this specific machine, or to retrieve information or component lists.

Classes

The HLC machine library models several accelerator components and concepts as classes: hlc::Magnet, hlc::BPM, and so on. You cannot construct objects of these classes directly because they are abstract — special implementation classes like hlc::GenericMagnet do the actual work. Instead, special factory methods on the Machine object can be used to obtain objects. For example,

```
hlc::Flash machine;
auto magnet = machine.get_magnet("D1IDUMP");
```

is much easier to use and read than:

```
hlc::Flash machine;
hlc::GenericMagnet magnet(machine, "D1IDUMP");
```

Here's a list of the available classes and their factory methods:

Class	Factory Method	Description
hlc::BPM	hlc::Machine::get_bpm()	Beam position monitor
hlc::BeamLimit	hlc::Machine::get_beam_limit()	Beam limit on the bunch pattern server
hlc::ChargeMonitor	hlc::Machine::get_charge_monitor()	Charge monitor (toroid or BPM)
hlc::EnergyGainInfo	hlc::Machine::get_energy_gain_info()	Access object for the LLRF energy gain server
hlc::LaserController	hlc::Machine::get_laser_controller()	Photoinjector laser controller (shutter etc.)
hlc::Magnet	hlc::Machine::get_magnet()	Magnet (corrector, quadrupole, ...)

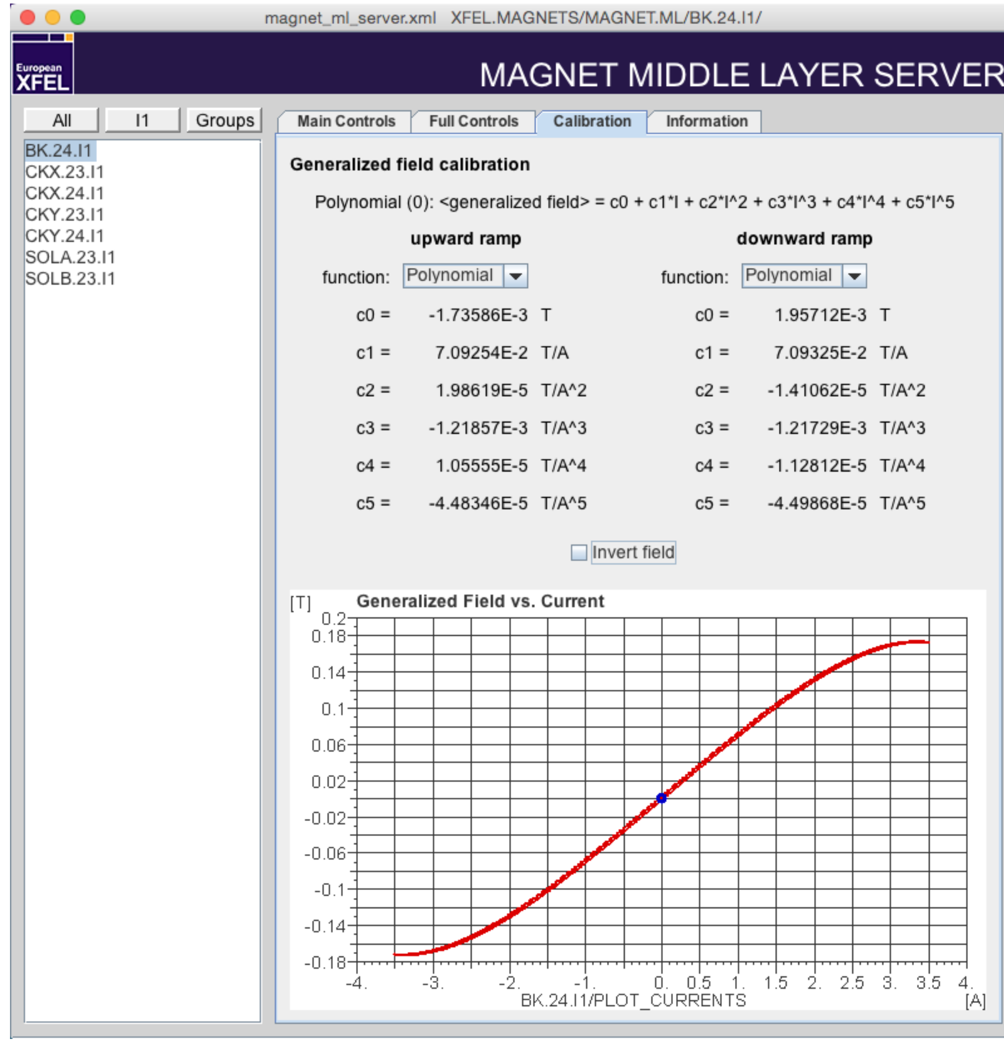
Example: Retrieve and use magnet list

Determine the current machine from a string, obtain a list of magnets, and print all currents:

```
auto machine = hlc::get_machine("XFEL"); // Also works with "/TTF2.DIAG/..." or similar strings
auto magnets = machine->get_magnet_list();

std::cout << "Found " << magnets.size() << " magnets for machine "
          << machine->get_name() << ":\n";

for (const auto &magnet : magnets)
{
    std::cout << magnet->get_name() << ": " <<
        std::to_string(magnet->read_current_sp()) << " A\n";
}
```

HLS examples – Magnet Middle Layer

- Magnets instead of power supplies
- Physical parameters instead of currents
- Hysteresis curve: keep track, use both branches
- Cycling
- Group handling
- Database functionality



HLS examples – Magnet Middle Layer

- Magnets i
  power su
- Physical p
  instead of
- Hysteresi
  keep track
  branches
- Cycling
- Group ha
- Database

High Level Controls SQL Library 18.10.9

Main Page | Namespaces | Classes | Files | Examples

High Level Controls SQL Library Documentation

The High Level Controls SQL Library contains several generic classes for easy access to an SQL database via ODBC and some specialized functions for interfacing with the MCADB configuration database in particular. All functions and classes are enclosed in namespace hlc. The single header file hlc_sql.h contains all necessary definitions.

Note

For information on build/system requirements and on database configuration, please refer to the Requirements and Configuration section.

Examples

Direct SQL access with data access functions

The main steps for connecting to the default database, sending a query, and retrieving data using the get_column() family of functions:

```
// Open database connection and send query
hlc::SqlDatabase db;
hlc::SqlQuery query = db.send_query("SELECT NAME1, Z FROM MCADB.XFEL_LONGLIST");

while(query.fetch())
{
    std::string name1 = query.get_column_as_string(1);
    double z = query.get_column_as_double(2);

    std::cout << name1 << ": " << z << std::endl;
}
```

Direct SQL access via data binding

The main steps for connecting to the default database, sending a query, and retrieving data using the bind_column() family of functions:

```
// Open database connection and send query
hlc::SqlDatabase db;
hlc::SqlQuery query = db.send_query("SELECT NAME1, Z FROM MCADB.XFEL_LONGLIST");

// Bind columns to variables
char name1[64];
query.bind_column_char(1, name1, sizeof(name1));
double z;
query.bind_column_double(2, &z, sizeof(double));

// Retrieve data
while(true)
{
    // Clear fields that might contain NULL values
    name1[0] = 0;

    if (!query.fetch())
        break;

    std::cout << name1 << ": " << z << std::endl;
}
```

Retrieving a component list

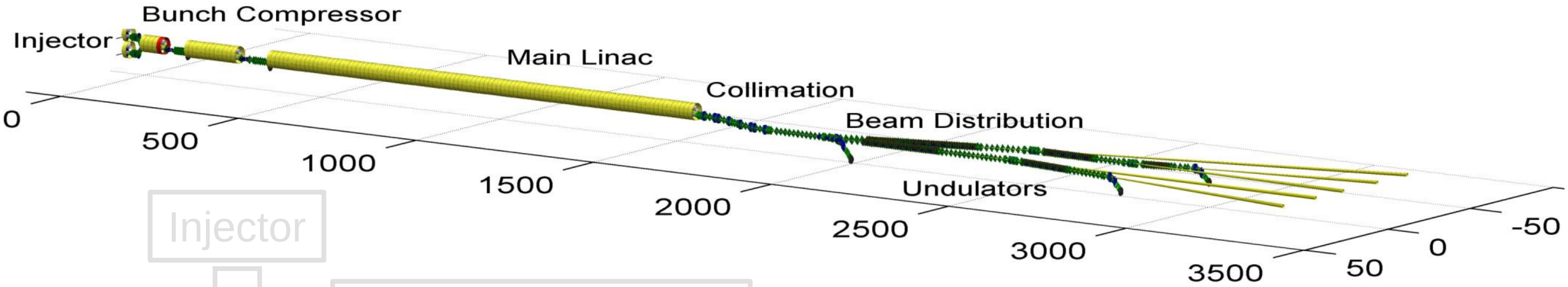
The main steps for connecting to the default database and retrieving the component list of the European XFEL:

```
// Open database connection and retrieve component list
hlc::SqlDatabase db;
hlc::SqlComponentList list;

std::string err_msg = list.read_from_db_with_backup(db, "XFEL");
if (!err_msg.empty())
{
    std::cerr << "Component list could not be read from database:\n";
    std::cerr << err_msg << "\n";
    std::cerr << "The list has been read from a backup file instead.\n\n";
}

int num_components = list.components_.size();
int num_flags = list.flag_names_.size();

// List available flags
std::cout << "Found " << num_flags << " flags:\n";
for (const std::string &flag : list.flag_names_)
```

LAYER SERVER

Tilt angle: 0.0 deg

CS and HLS @ the EU-XFEL – HLS examples



Injector

Extensively covered by Matthias S.

Bunch Compression

some words on **magnets**

Main Linac

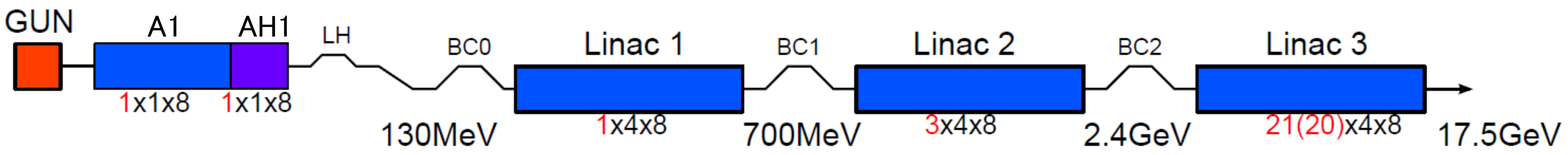
will have a look at the **energy management**

Undulators

all parts exist see **outlook**

HLS examples – Linac Energy Control

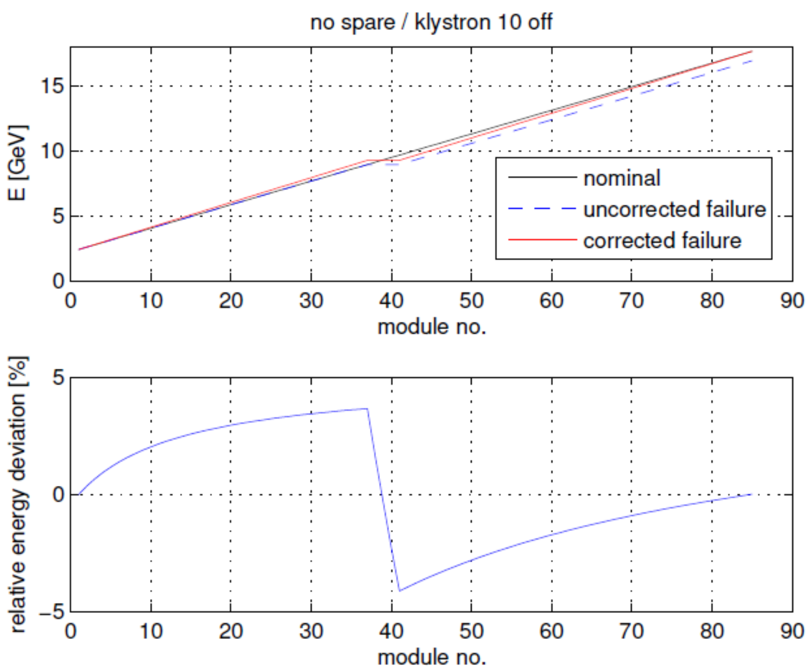
- No energy management in Injector and Linac 1 (no spare klystron)
  - Energy setup using *sum voltage server* ("physical knob")
- No spare klystron in Linac 2 …
  - but dedicated modules → some headroom
- One klystron as spare in Linac 3
  - Investigate impact off different usage scenarios

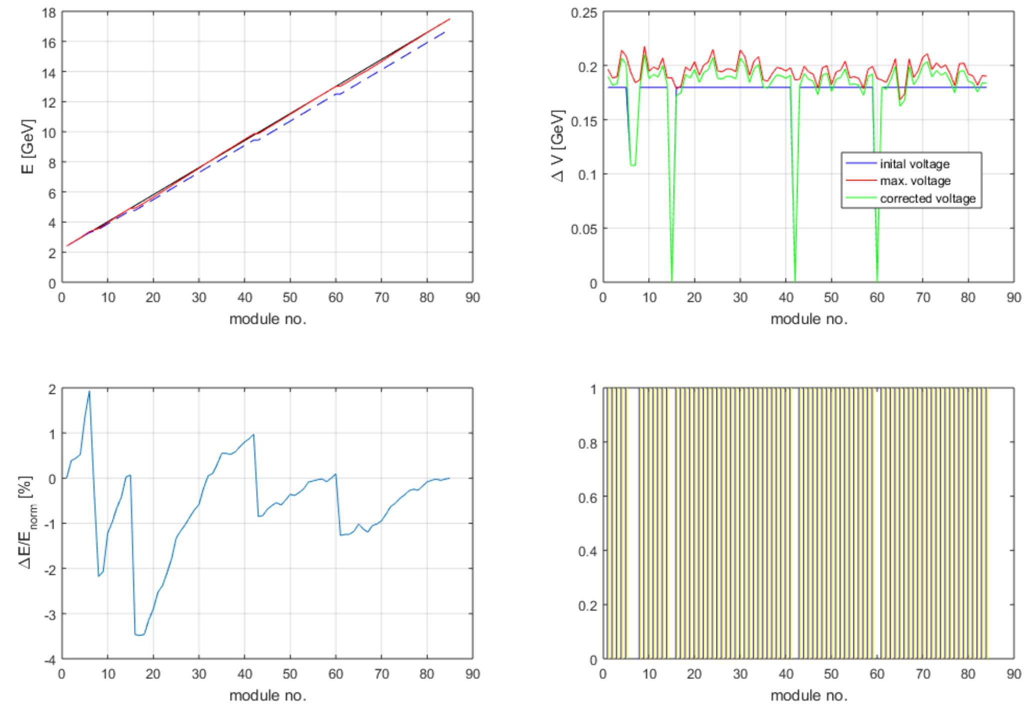


HLS examples – Linac Energy Control

Iterative adjustment of voltage per station respecting individual station capacity

$$\Delta V_i' = \frac{E_{\text{nominal}} - \sum_{j \notin I} \Delta V_j}{|I|} \frac{w_i}{\langle w_i \rangle}, i \in I$$

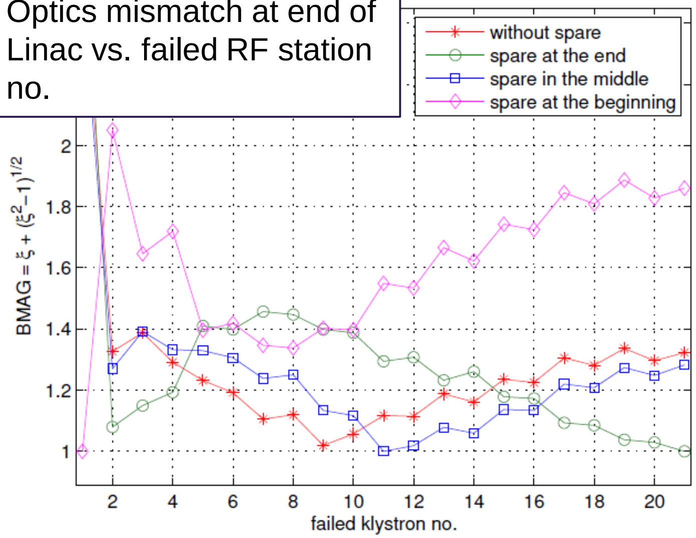




Courtesy: B. Beutner (DESY)

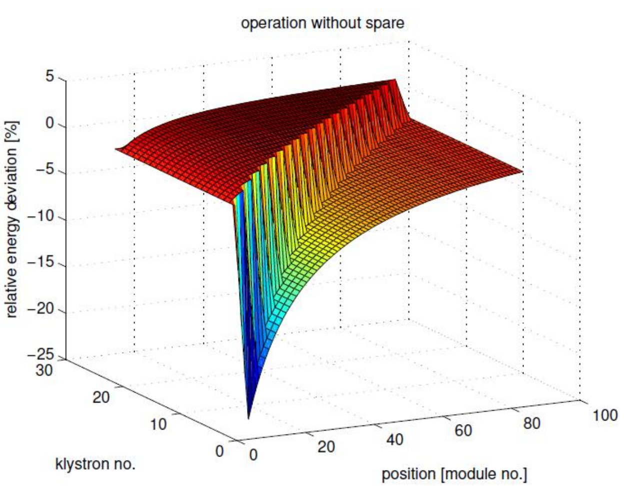
HLS examples – Linac Energy Control

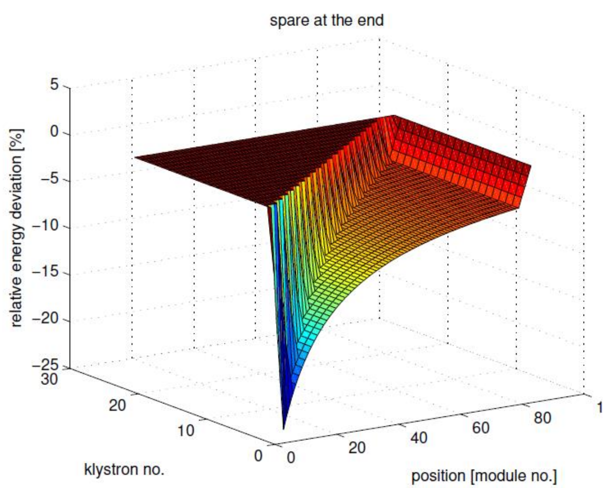
Optics mismatch at end of Linac vs. failed RF station no.

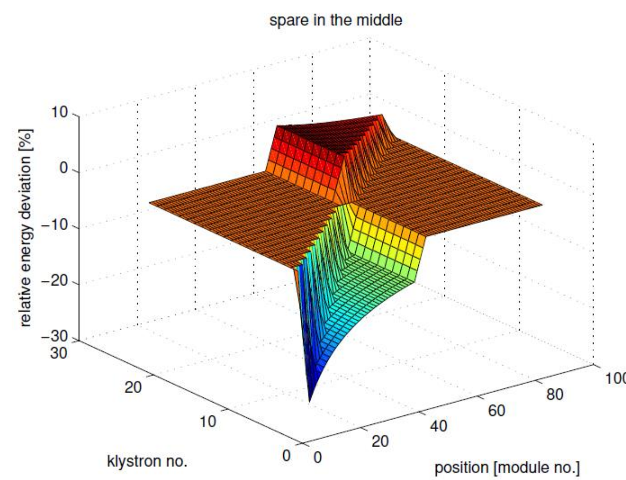


A configuration with all stations in operation at reduced voltage (red) is most favorable in terms of:

- Optics mismatch for random station failure
- Reduced average power per station



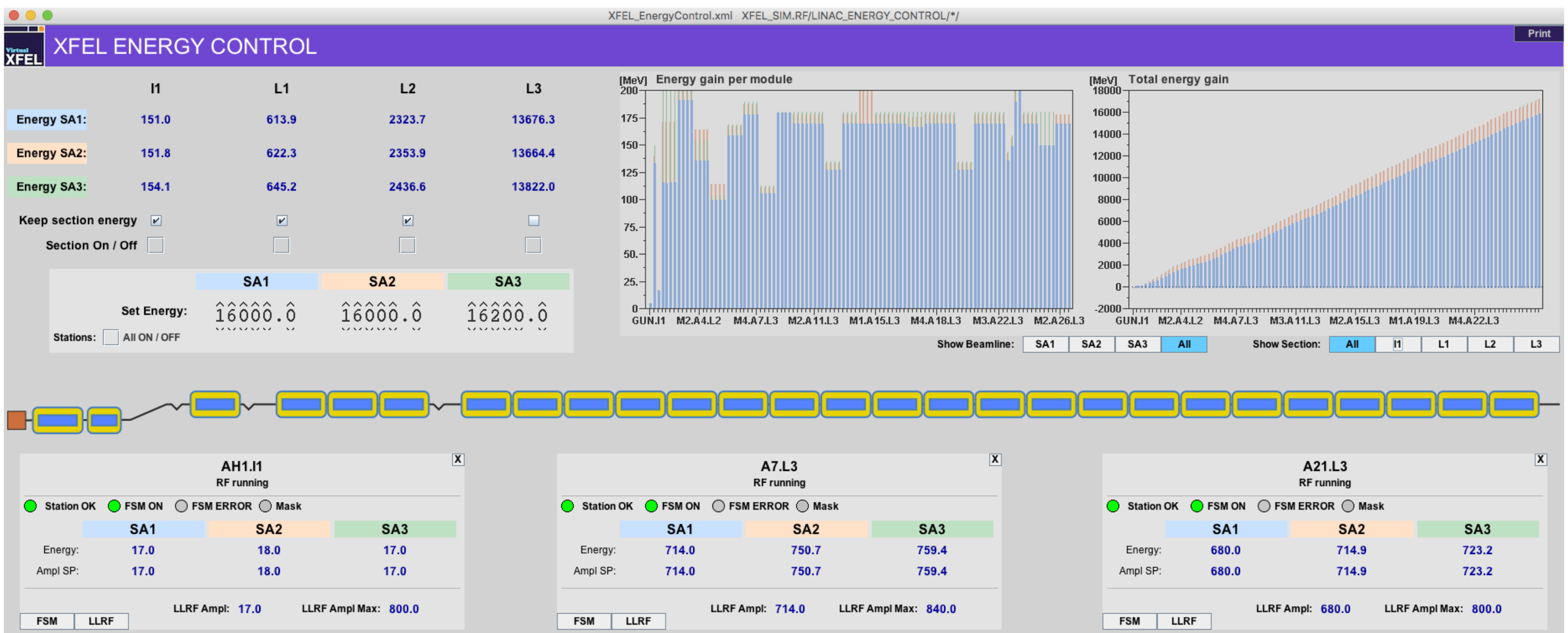




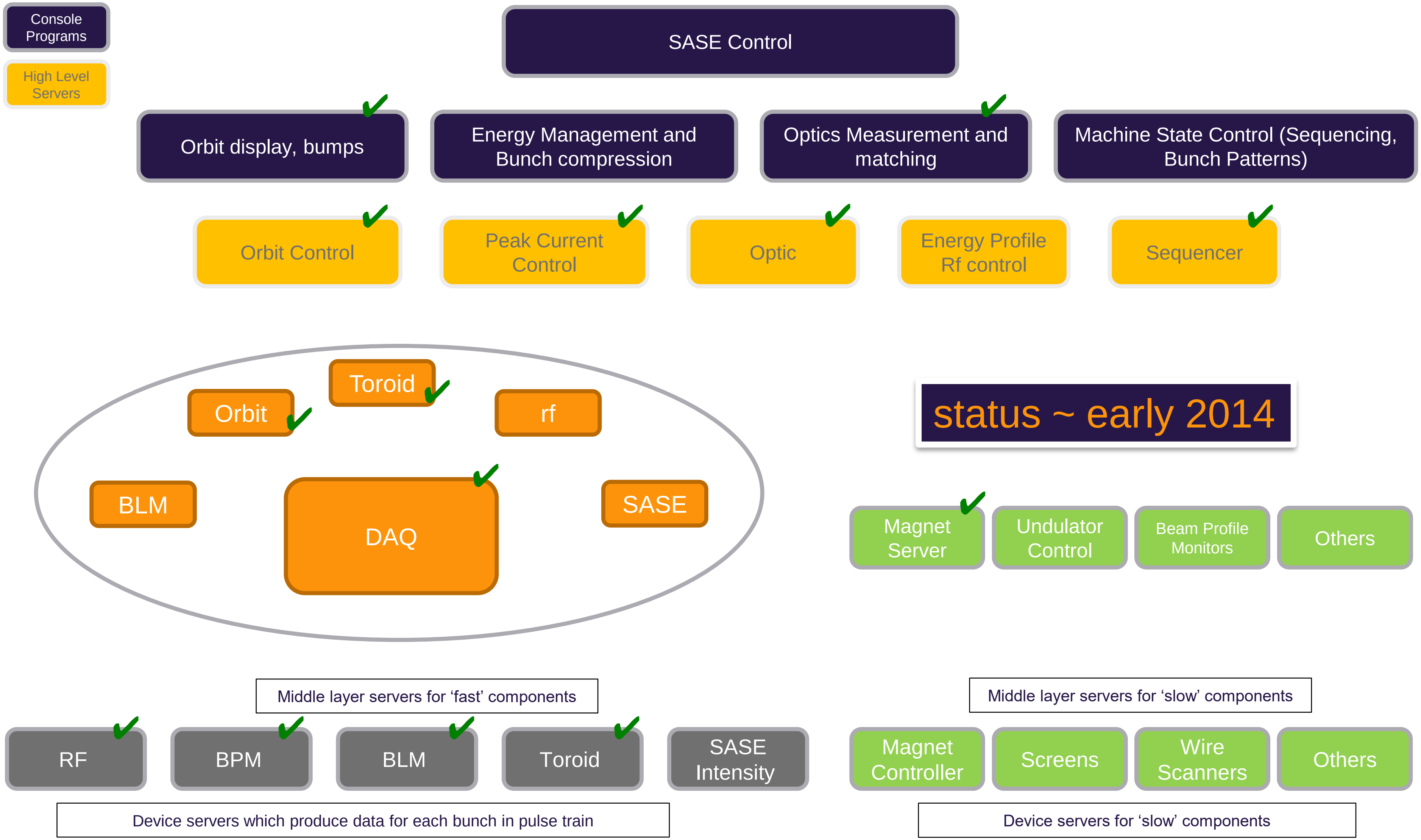
Courtesy: B. Beutner (DESY)

HLS examples – Linac Energy Control

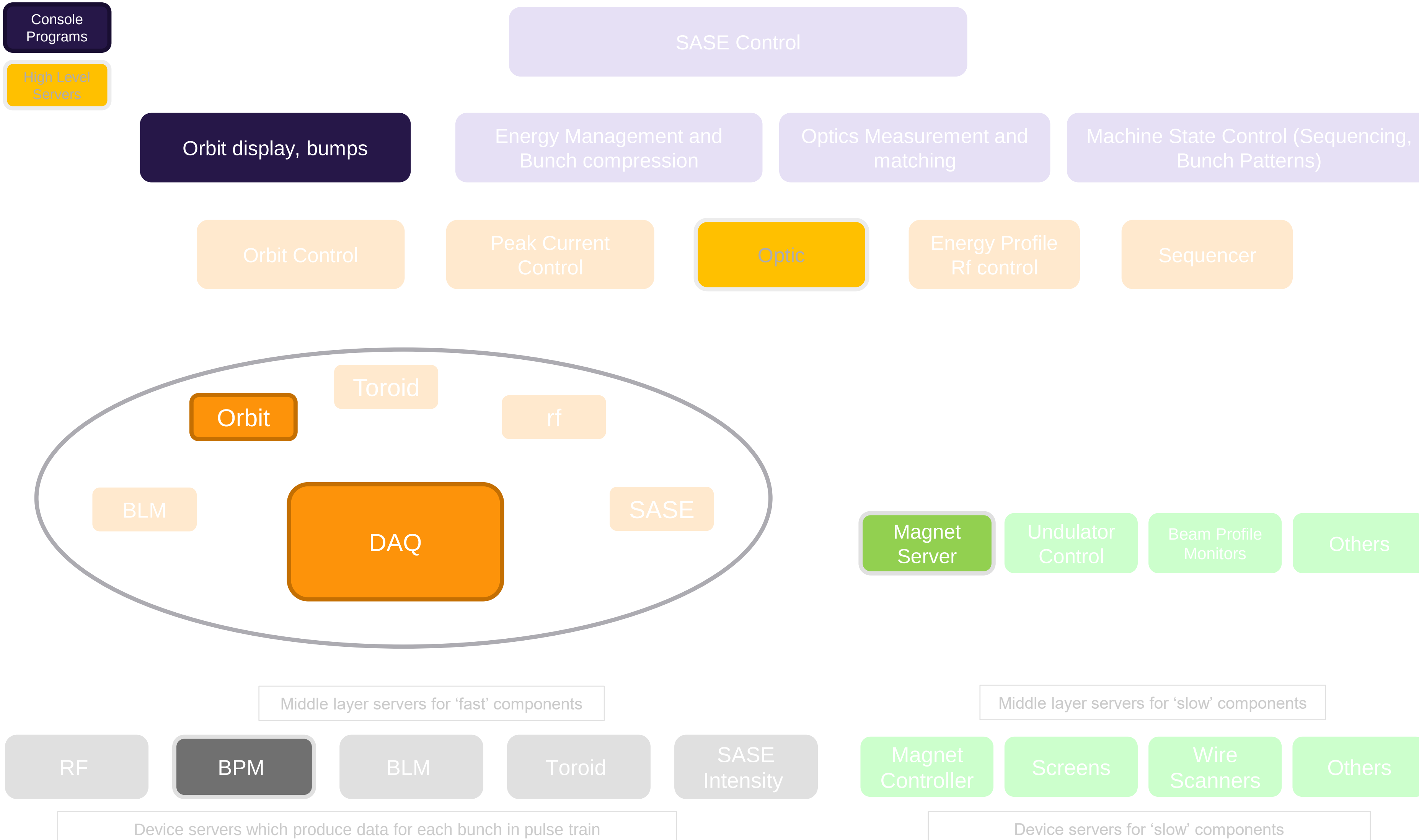
- This recipe has been implemented in C++ based middle layer server: *Linac Energy Control*
- First tests in Virtual XFEL worked out well



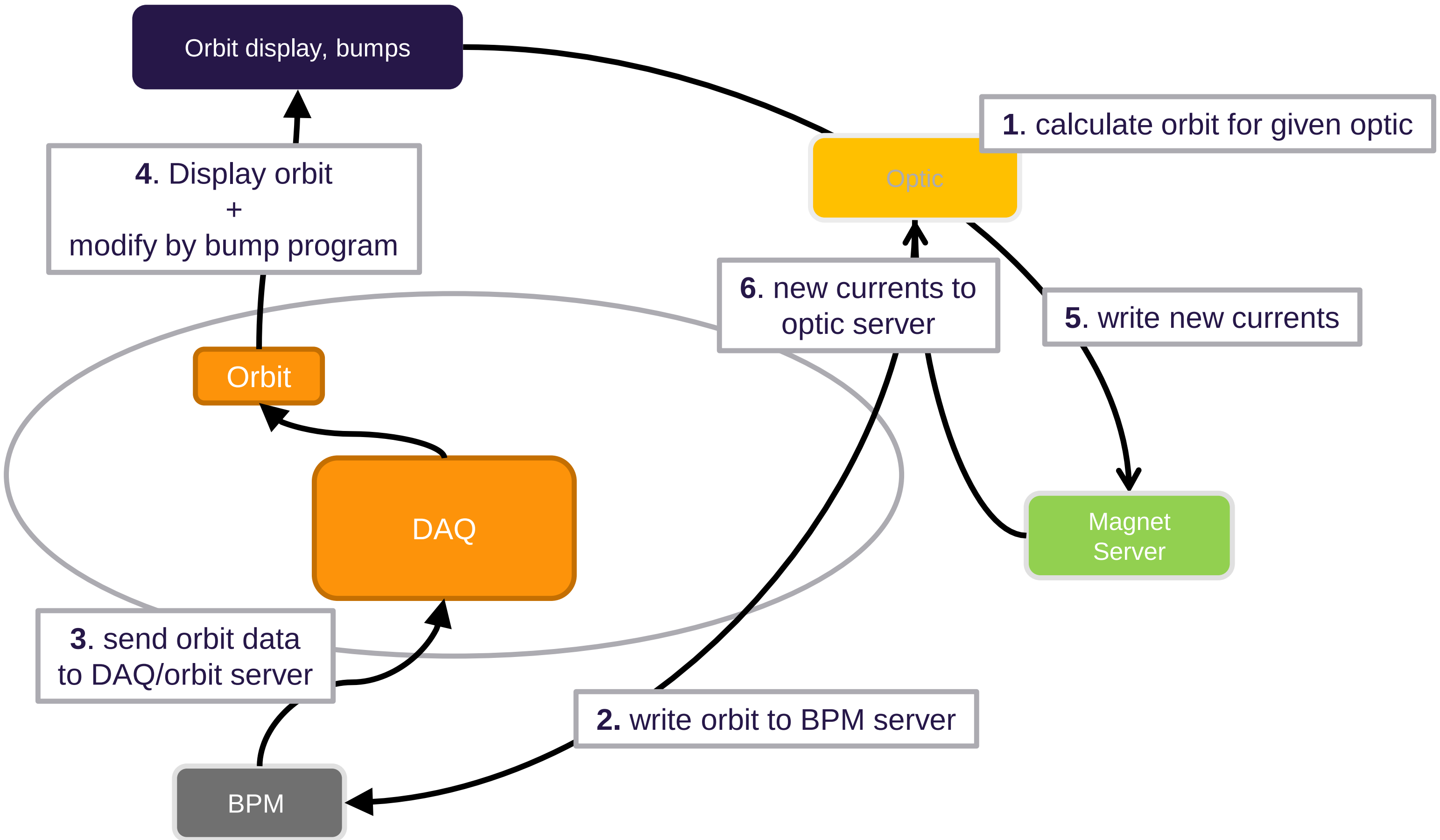
The Virtual XFEL – The idea



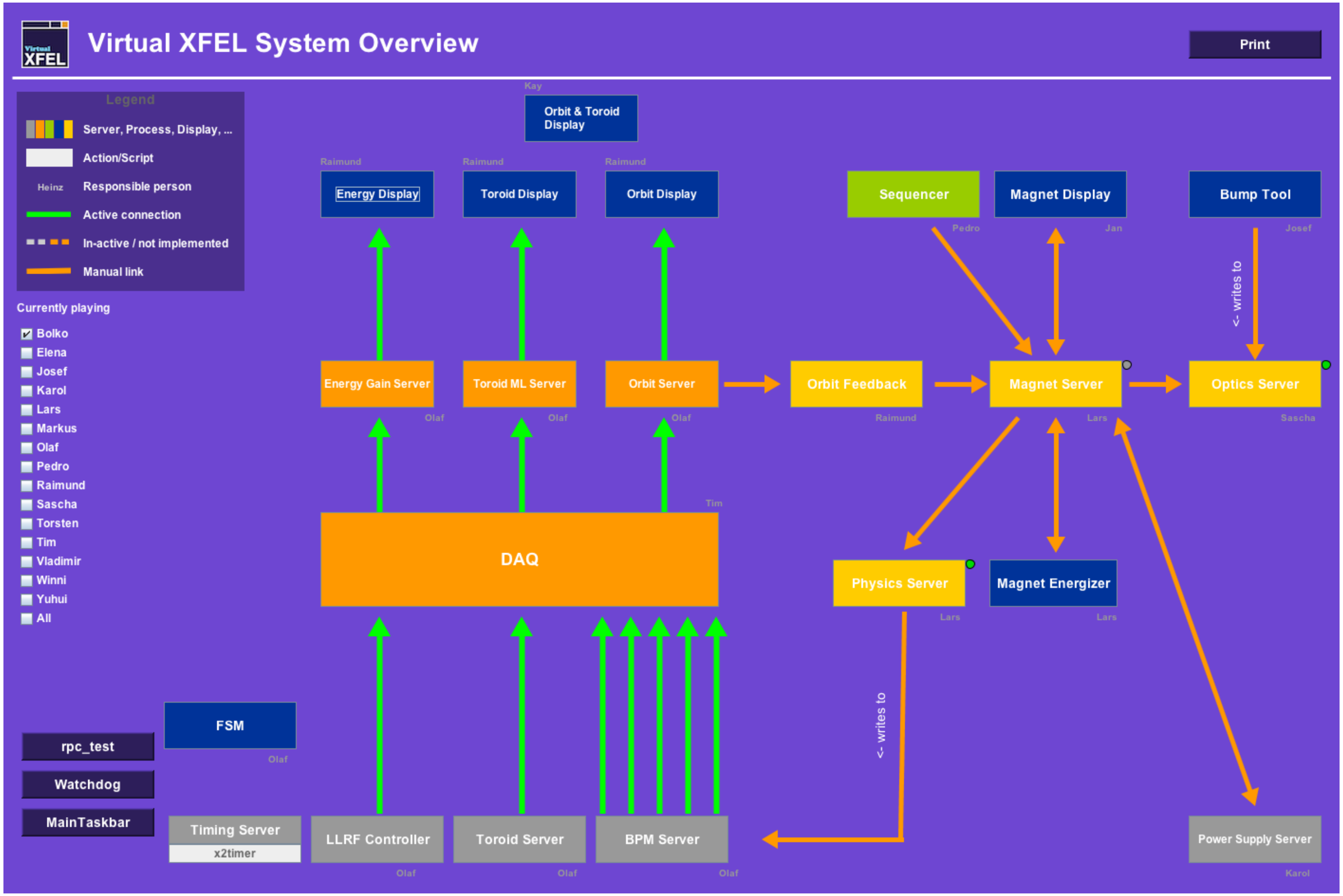
The Virtual XFEL – The idea



The Virtual XFEL – The idea



The Virtual XFEL – Overview



The Virtual XFEL – Summarizing

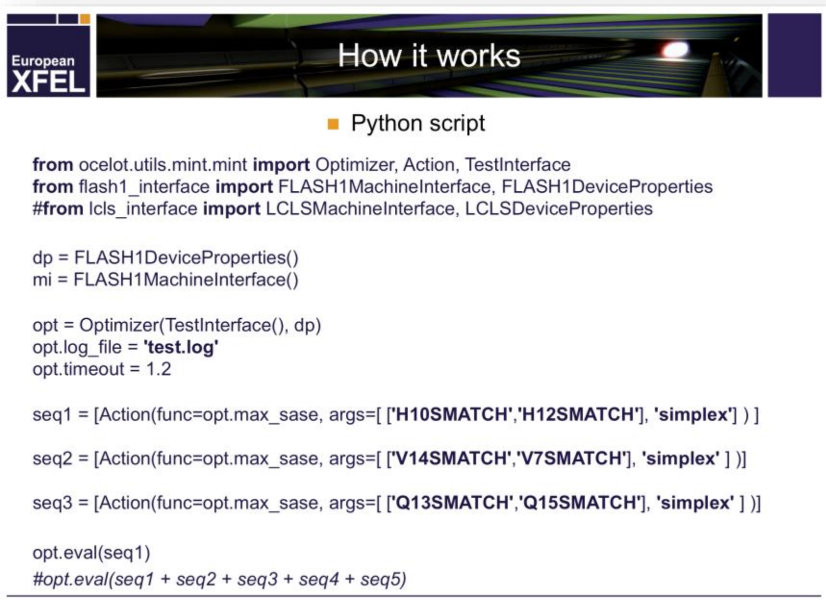
- The VXFEL **allows** us to:
  - Test the **network** and **data throughput**
  - Tests of the **Timing System** and **Bunch Pattern Handling**
  - Is **the** test bed for all **High Level Software**
  - Test **naming conventions** and prepare server **configurations**
  - **Port** software from the VXFEL **1:1** to the XFEL
  - Develop and test **display concepts and displays**
  - ...

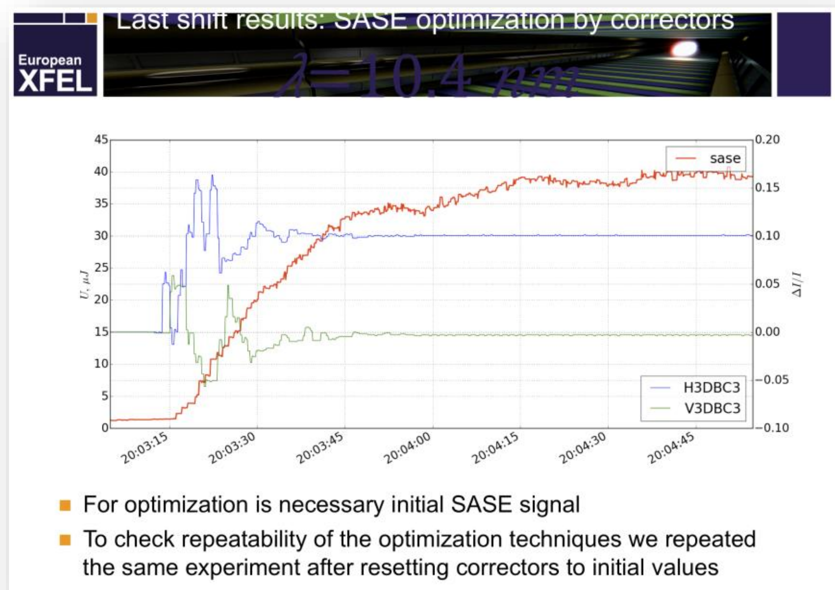
- The VXFEL **does not** or only partly allow to:
  - Test **hardware**
  - Do full **featured physical simulations**

(for more see e.g.: ICALEPCS 2015 TUD3O04)

Outlook – Machine Optimization using HLS

- At FLASH a SASE optimization is done by operators
- To get optimal performance within shortest time machine driven procedures might be faster
- First tests using OCELOT have been done at FLASH





Courtesy: S. Tomin (European XFEL)

Outlook – Machine Optimization using HLS

- Needed interfaces (DOOCS) for EU-XFEL already exist
- Integration in existing HLS landscape should be easy
- OCELOT developers started to work in close collaboration with HLS team

A generic optimizer has still been missing on our shopping list!

Outlook – Big Data

- EU-XFEL will produce **TB per day** of pure machine data
- A good part of this is available via the DAQ
- But also the CS infrastructure itself will provide a very large amount of diagnostic data (e.g. TBs of log files)

➔ **Use 'Big Data' tools to get insides**

- We evaluated *Apache Spark* on several data sets
- First results look promising

APACHE Spark™

Summary

Much could **not been covered**!

- Timing system, multi-beam line operation
  - see e.g. http://doocs.desy.de → µTCA Development
- Machine Protection System
  - see e.g. ICALEPCS 2013 TUCOCA01
- Matlab tools
  - see e.g. www.desy.de/fel-beam → Talks
- Libraries
  - http://xfel-wiki.desy.de/Control_system
- Feedbacks
  - see e.g. ICALEPCS 2013 THPPC121
- ...

European XFEL

Summary

- We got a good **team** focusing **exclusively** on the **high level view**
- This **focused** on **architecture**, **concepts** and common **algorithms**
- to **develop**, test and implement **libraries** with common high level functionality
- With the **Virtual XFEL** we have a **great environment** to do prior test and debug high level software
- Many **servers** and **tools** have been created this way and we:

➔ think **we** are well prepared – let's get started!

Thank you for your attention!