High Level Application Development in EPICS

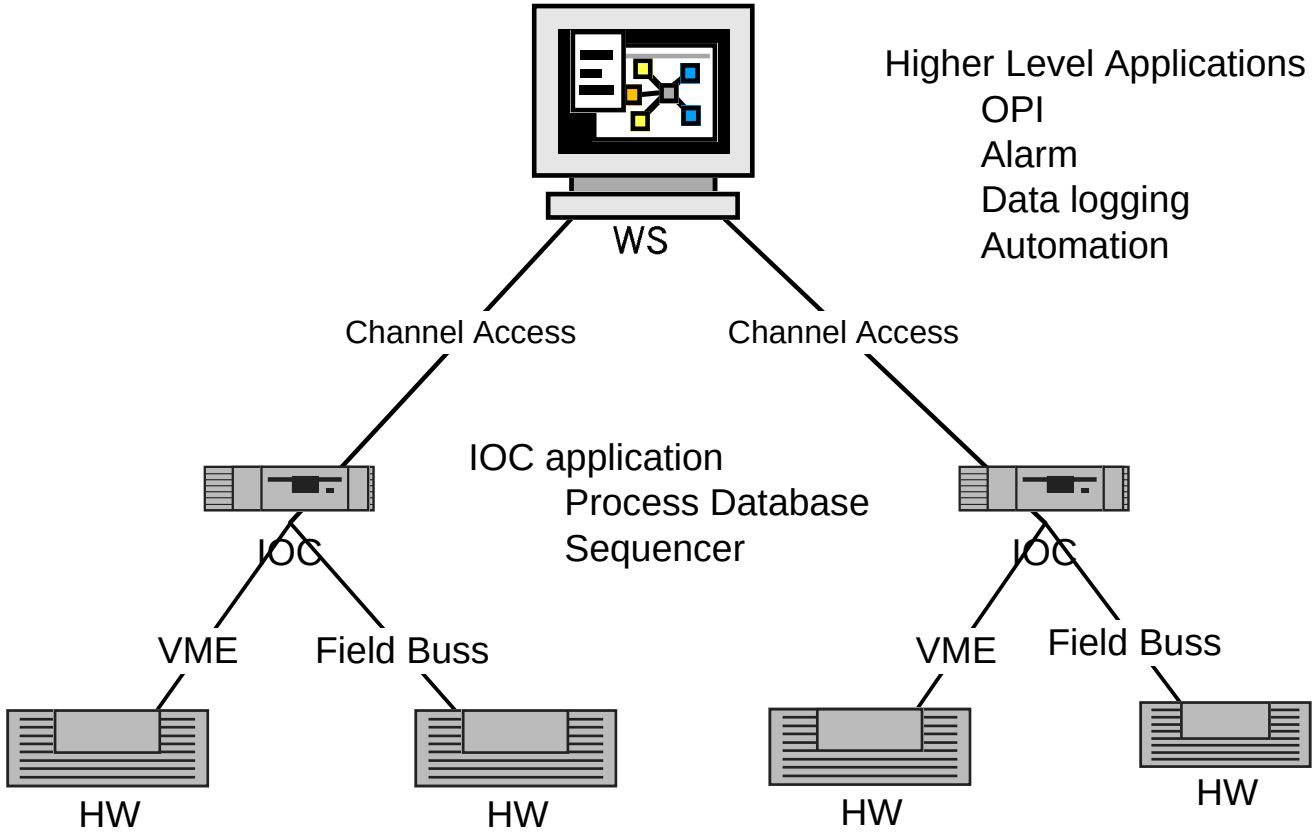
IHEP EPICS seminar, Aug. 22, 2001

Noboru Yamamoto
KEKB control group
KEK, JAPAN

EPICS Application

- EPICS is a toolkit, i.e. a set of software, to develop control system.
  - In the other words, It is not a control system itself. YOU must build your own control system for your equipment.
- EPICS includes lot of powerful tools(generic applications) to develop your control system ~ Application.
  - Proper use of these tools will reduce cost of development.
  - 99% of the system may be covered by these tools.
  - You also need tools to cover residual 1 % of the system, which makes you system unique.

EPICS Application



There are many places/ways to implement one control algorithm in EPICS!

EPICS Software Tools

IOC

- Process Database
  - Records
  - Record Links
  - Record/Device Support
  - Device Driver
- Sequencer program
  - (Fast) Feedback
  - sequential control
- Custom applications

  Tools:
  - jdct, gdct, CAPFAST
  - snc (SNL compiler)
  - Tornado & C/C++

Host Work Stations

- Generic Applications
  - EDD/DM, MEDM, dm2k
  - ALH (Alarm Handler)
  - Channel Archiver/
  - Strip Tool
- Sequencer program
- Custom applications
  - Machine specific feature
  - User Interaction using GUI

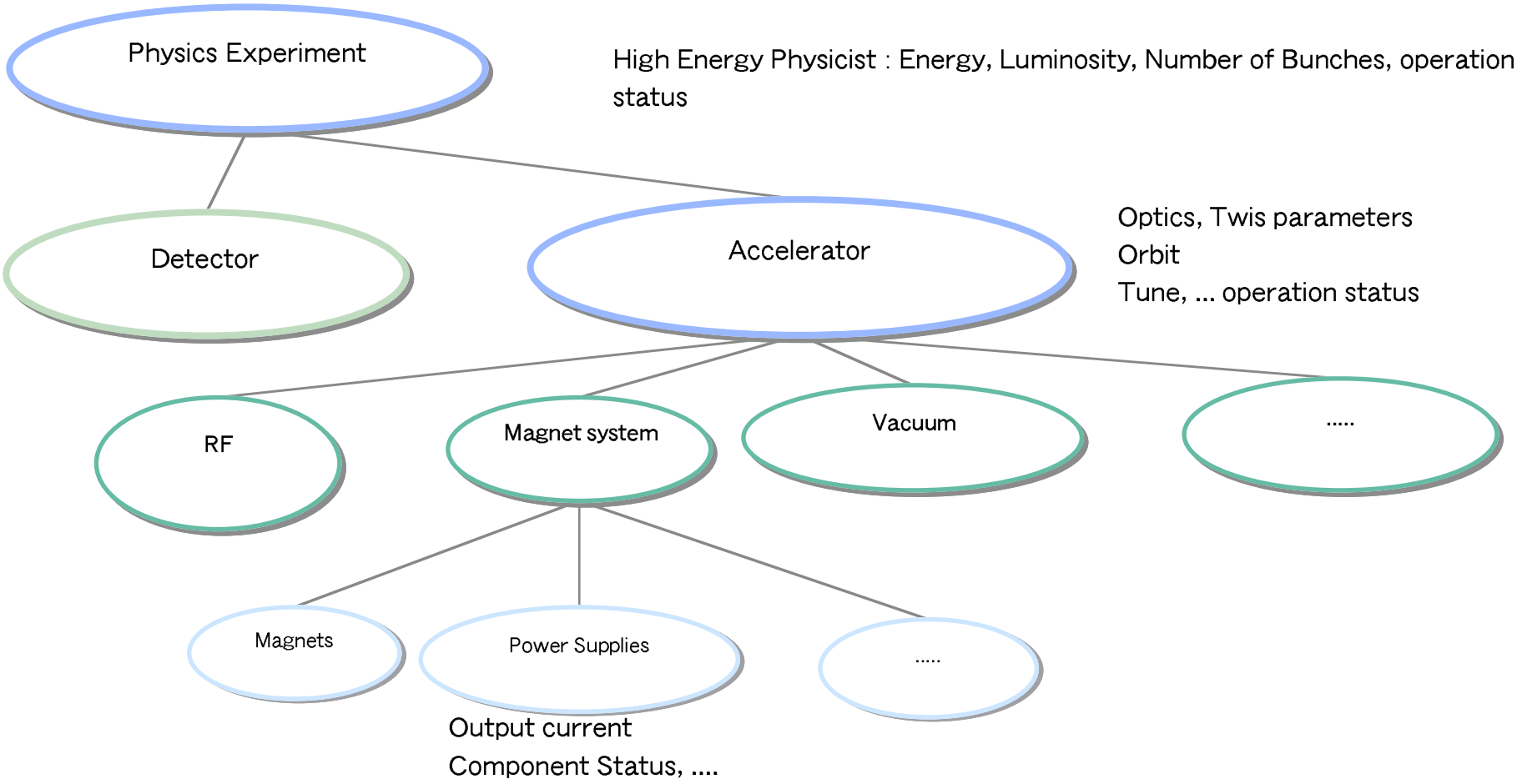
  Tools:
  - EPICS CA clients
  - SNL/snc
  - C/C++ with X window/Motif
  - *Scripting Languages*

Hierarchy of applications

- A set of controls for each equipments is not enough.
- You must have an application to control an accelerator as whole.
- Controls software will have natural hierarchical structure reflects an accelerator HW.
- Each sub control system/application should have clear interface to
- upper level application*).

*) However, we don't need to have special protocol to exchange data at each interfaces. EPICS CA can be (should be?) used as a standard protocol.

Hierarchy of applications



Communication between components

In each level of the sub system interface:

- Set parameters
- Read parameters
- Monitor parameters/Status Change/Alarm
- Notify Status change to upper level
- Logging Data
- Analyzing Data

In each level of the sub system interface:

- Set parameters
- Read parameters
- Monitor parameters/Status Change/Alarm
- Notify Status change to upper level
- Logging Data
- Analyzing Data

EPICS CA and EPICS tools can do these Work!

Application or component

Software in each system, except Top Level, has interfaces for upper and lower level software.

They are not simple application, but Components.

How each components can communicate

- CORBA ?
- EPICS channels on IOC?
- Portable Channel Access Server with dynamic channel?
- ???

Programming tools:

You can use virtually any tools to develop application.

- EPICS Database
- Sequencer
  - On IOC
  - On Host
- C/C++
- Java
- RAD tools
  - Visual Basic
  - Tcl
  - Perl
  - Lab View.....

Control software

Lower Level:

- Simple, well understand behavior
- Control algorithm is also well understood and stable

Higher Level:

- Complicated, Many information
- Unexpected surprise during operation
- Frequent Change in a control algorithm, or algorithm "grows" in the operation of machine.

Programing Languages

You need , at least , TWO programing languages.

- System Language
  - Usually compiled language
  - Fast, Static Typing
  - Define Object
  - Long development
  - C, C++, Java
- Glue Language
  - Rapid Prototyping, Short development, even One-liner program.
  - Usually interpreted Language
  - Dynamic Typing
  - sh, tcl, perl, python(CPython, JPython)

CA Interfaces to other tools and languages

- CDEV: Common Device application framework and API (JLAB)
- ActiveX-CA: CA client for Active-X programs (LANL)
- CA-ActiveX: CA server of Active-X objects (LANL)
- EZCA: Simplified Channel Access interface library for C
- SCA : Simple Channel Access (LBL/ALS)
- JCA : Channel Access client class library for Java
- ezcaFort: Simplified Channel Access interface library for Fortran

- CaMath: CA client interface for Mathematica
- caPython & caChannel: Python interfaces to CA (FNAL)
- CaWave: CA client for Precision Visuals' analysis package
- CaWingz: CA client for the Unix Spreadsheet
- PEZCA: CA interface to Perl, using EZCA (BESSY)
- SDDS: The Self-Describing Data Sets analysis package
- Tcl Interfaces: ET and IT
- SAD/Tk : Accelerator Modeling program with Tk and CA Interface. (KEK/KEKB)
- Python/Tk : CA interface to Python. (KEK/KEKB)

- Gateway: Channel Access proxy server (BESSY)
- PADL: Portable CA Server interface to ILL (LANL)
- PICAS: Channel Access Portable Server (LANL)

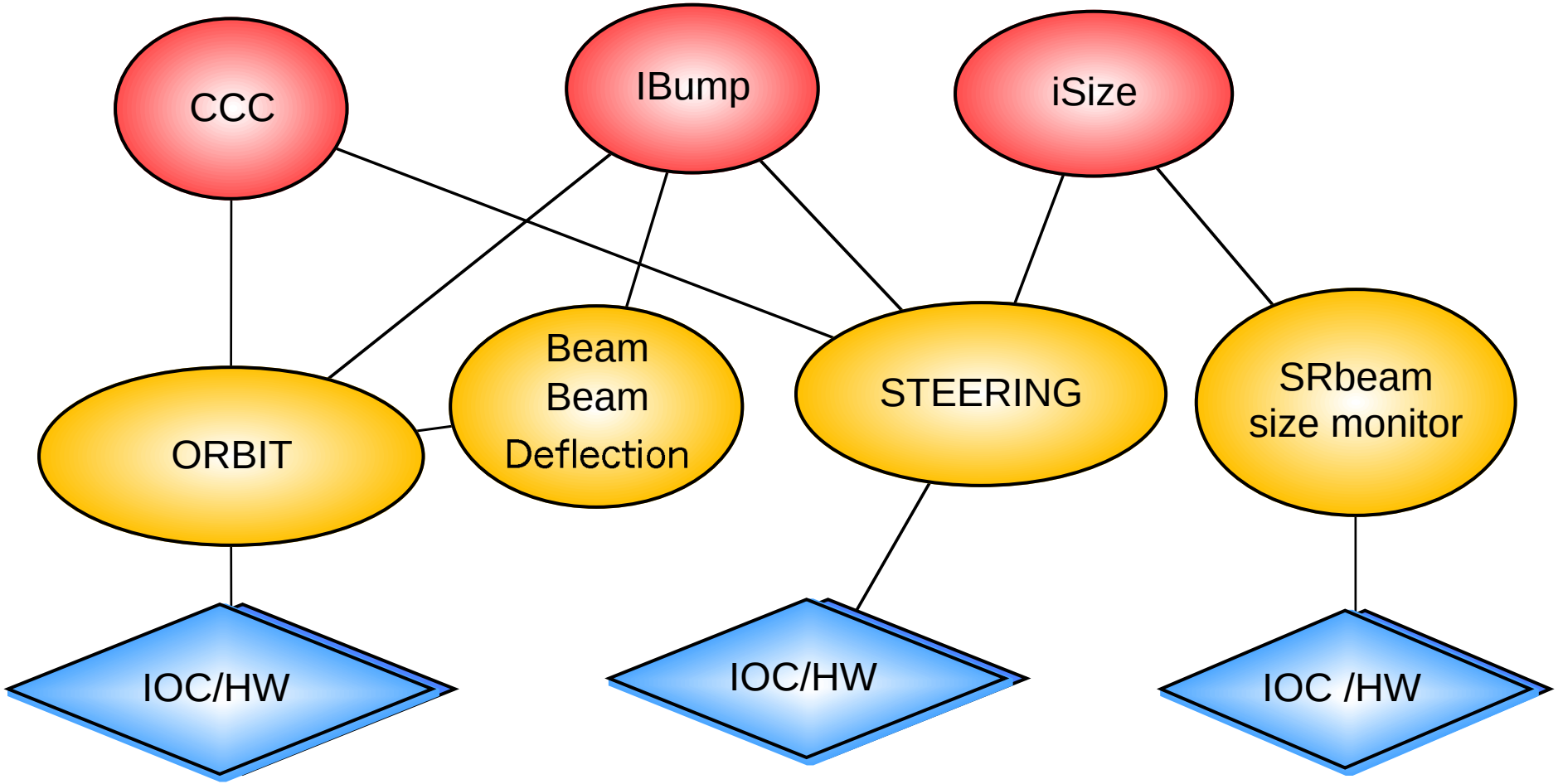
So Many Tools ! *But Why?*

Case Study: Orbit correction in KEKB

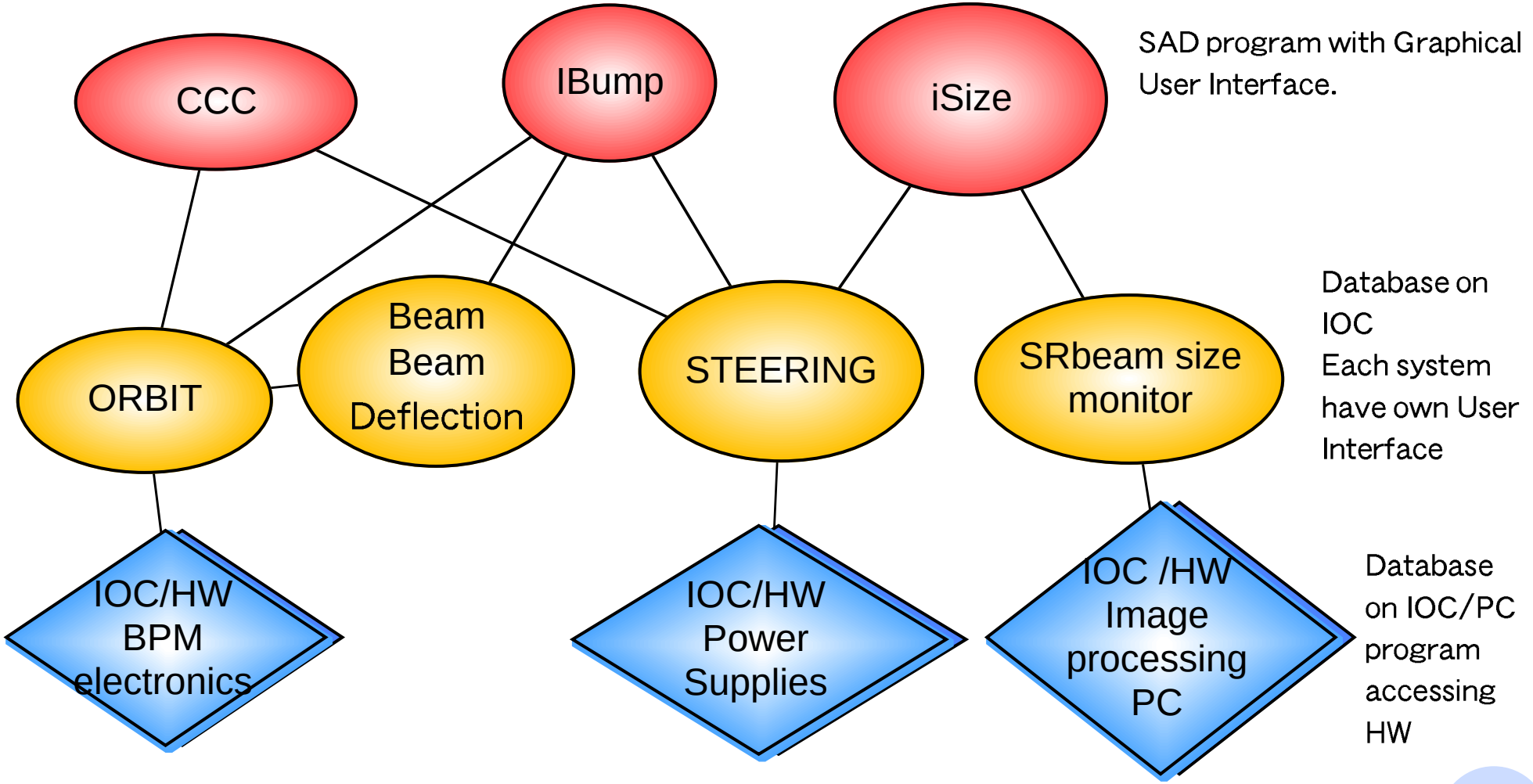
Beam Orbit control applications in KEK

- CCC: Continuous COD correction
  - Periodically read orbit and correct difference from the reference orbit, the golden orbit.
- iBump:IP orbit feedback. Controls relative displacement and angle of beams.
  - Read Orbit data near IP.
  - Calculate relative displacement and angle of electron and positron beams from beam–beam deflection.
- iSize: Bump orbit to control beams
  - Get measured beam size from beam size monitor system.
  - adjust beam orbit, BUMP, to control beam size.

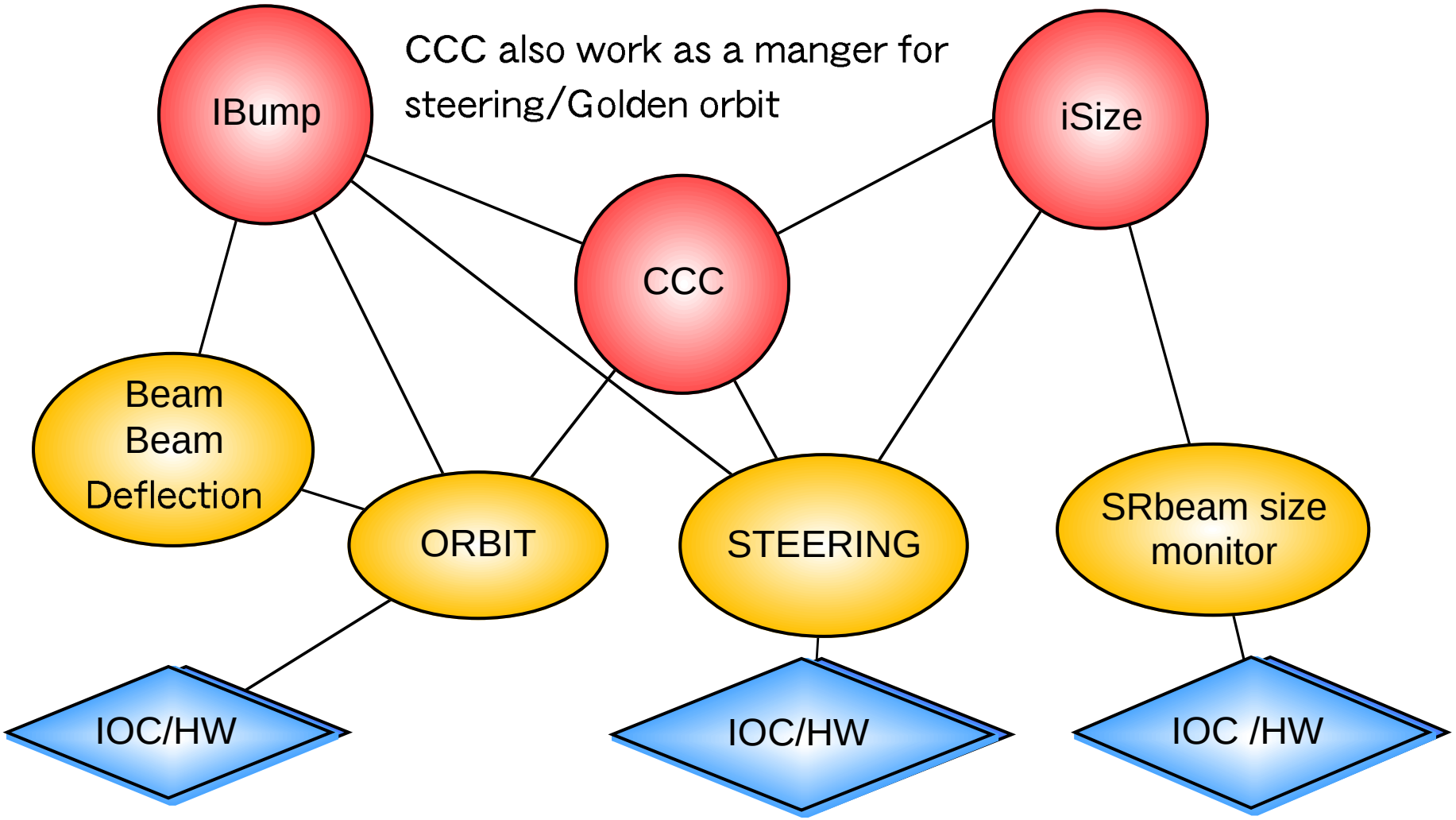
Case Study: Processes/Databases



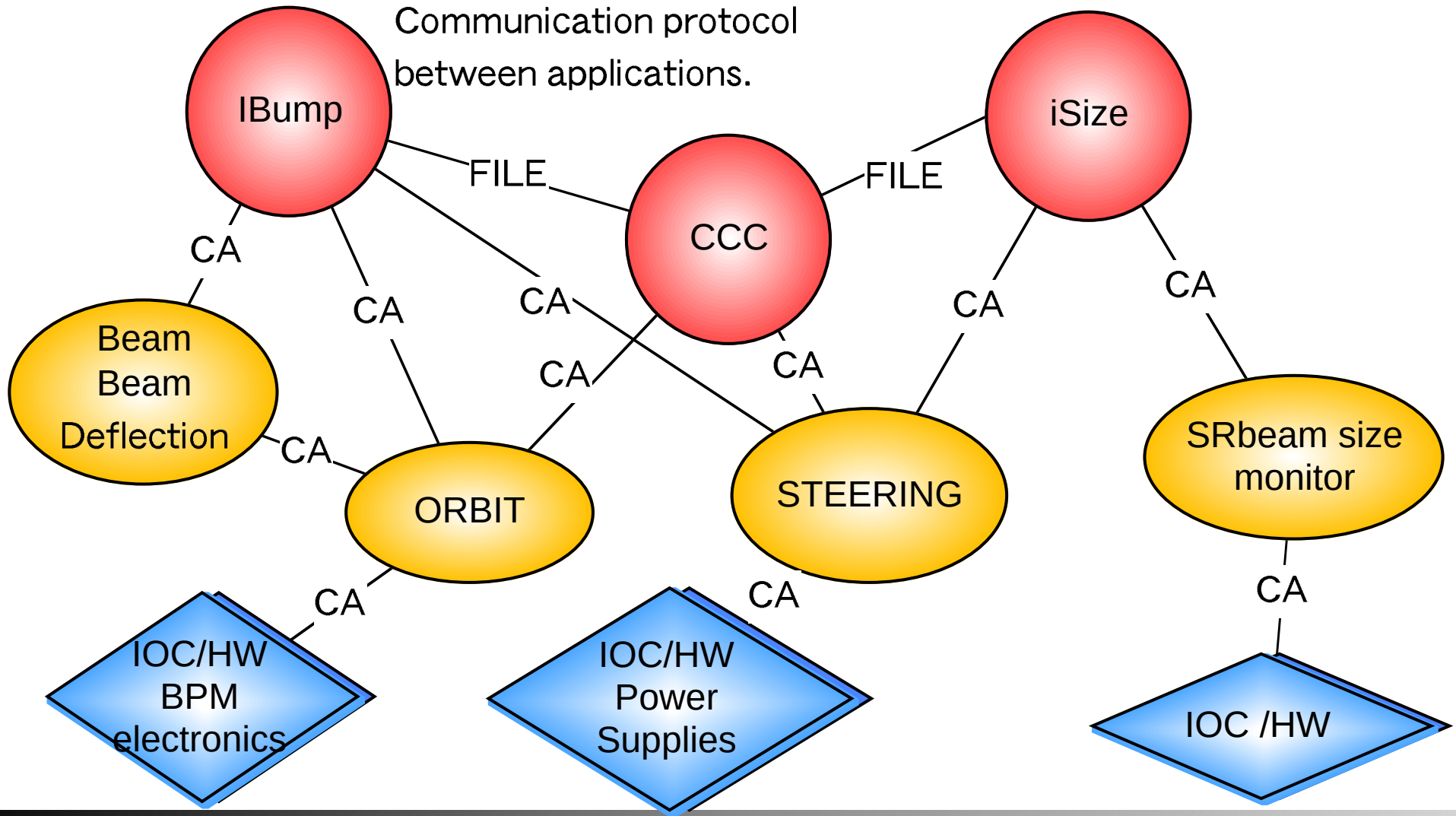
Case Study: Tools



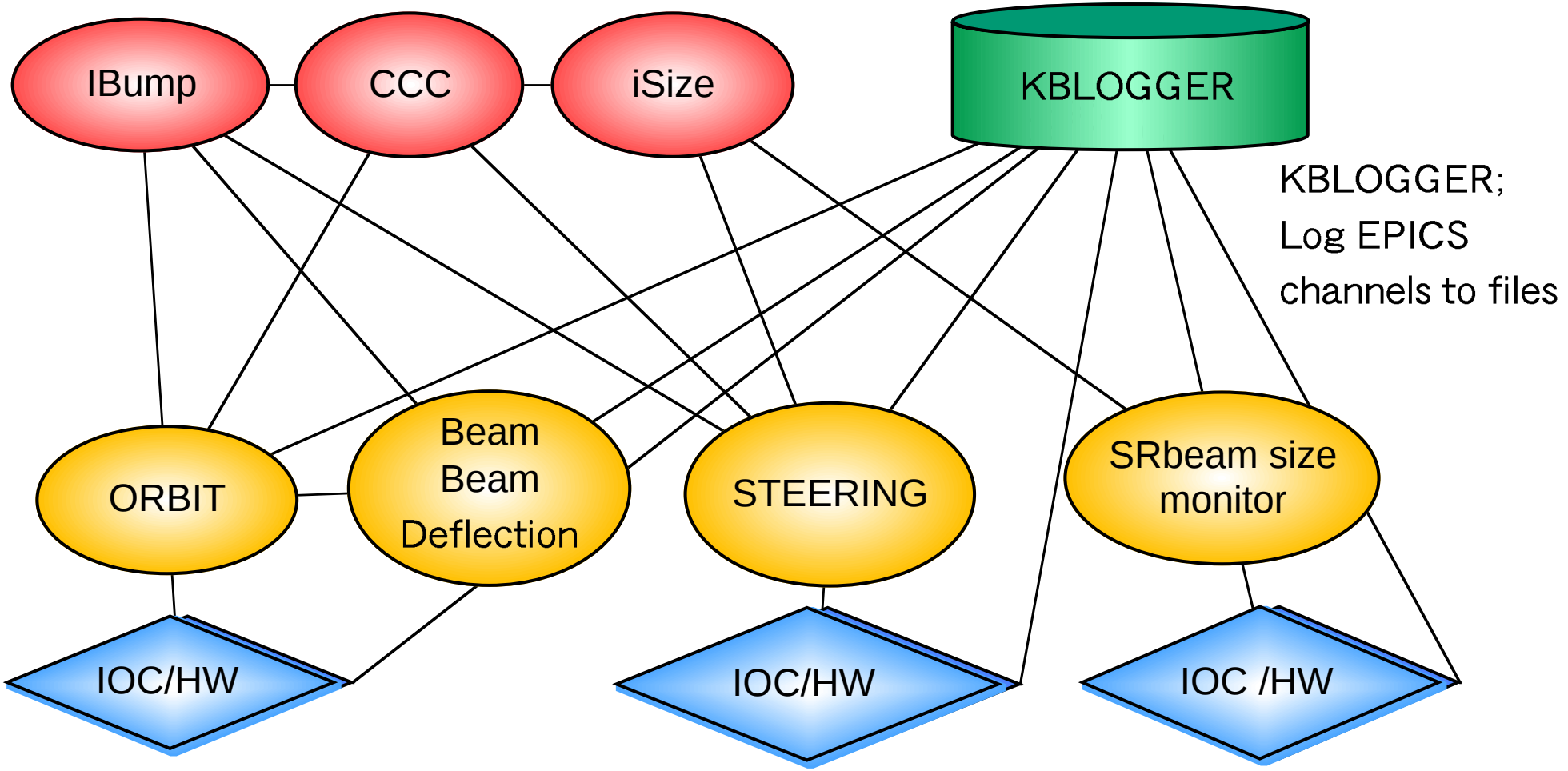
Case Study: Synchronization



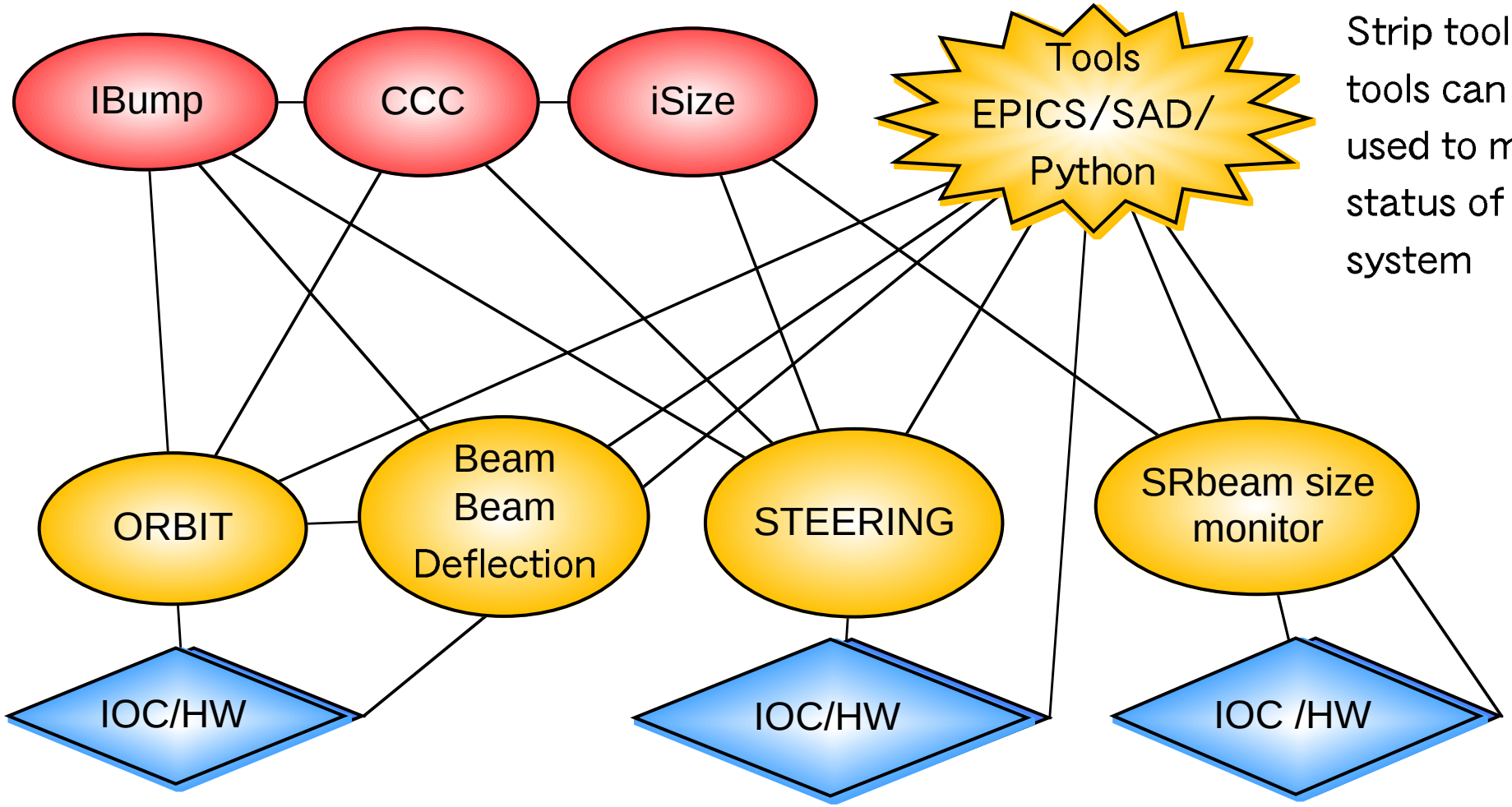
Case Study: Inter-Process communication



Case Study: Data logging

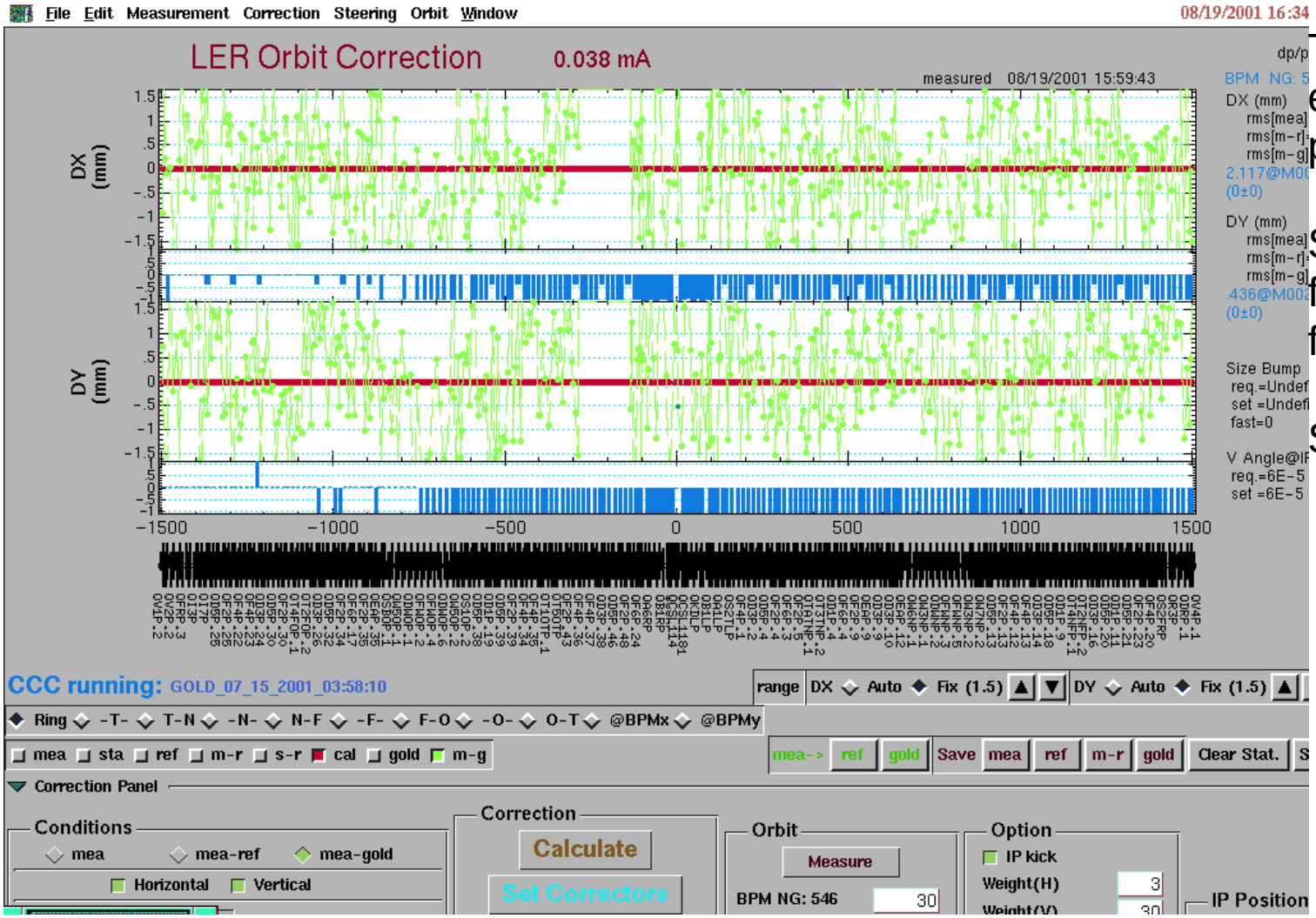


Case Study: Orbit correction in KEKB



Strip tool like tools can be used to monitor status of the system

CCC: User Interface

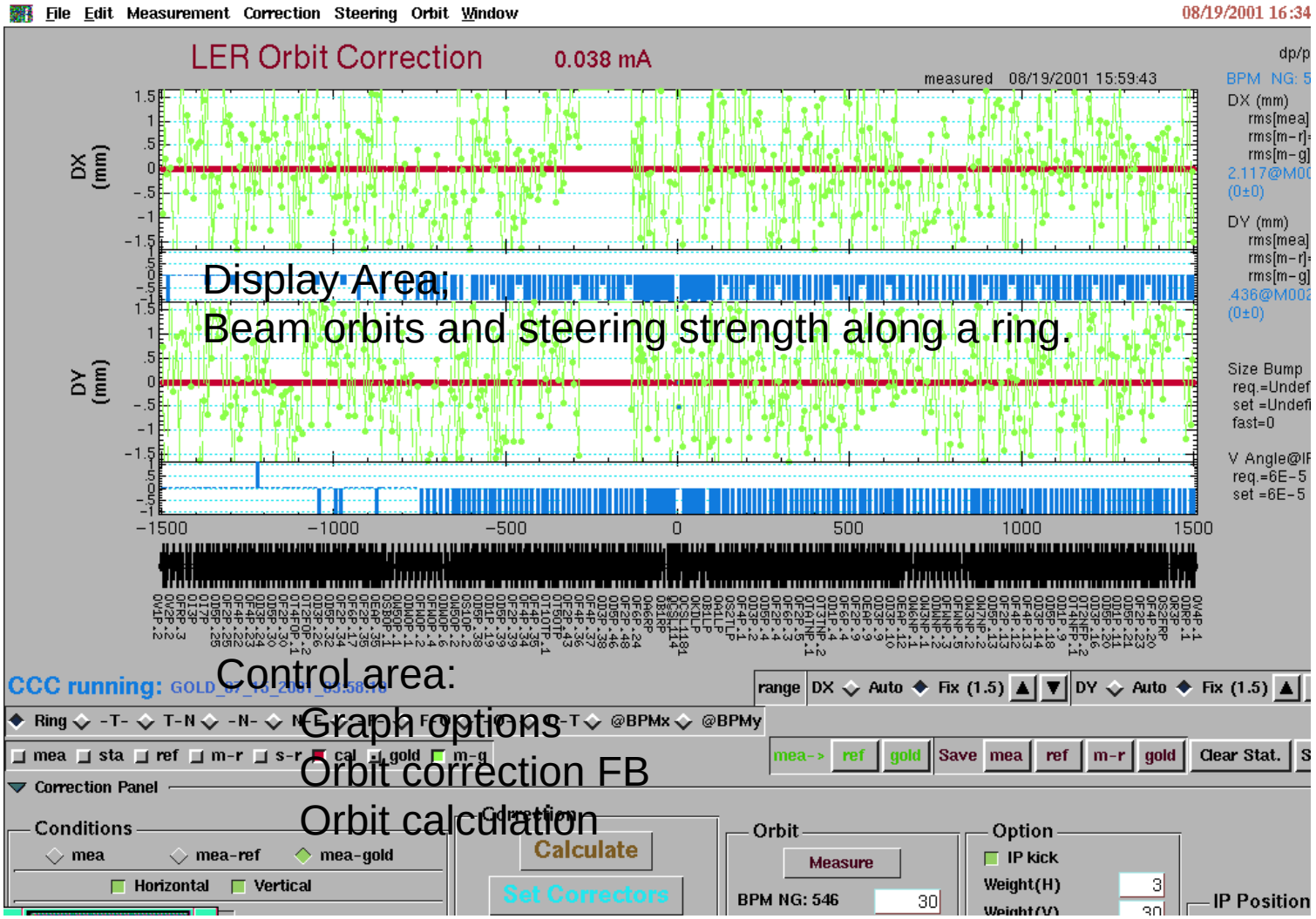


This program is written entirely by SAD/FFS programming language.

SAD has many functions for optics calculation and for graph.

SAD has API for
- Tk Widget
- EPICS CA

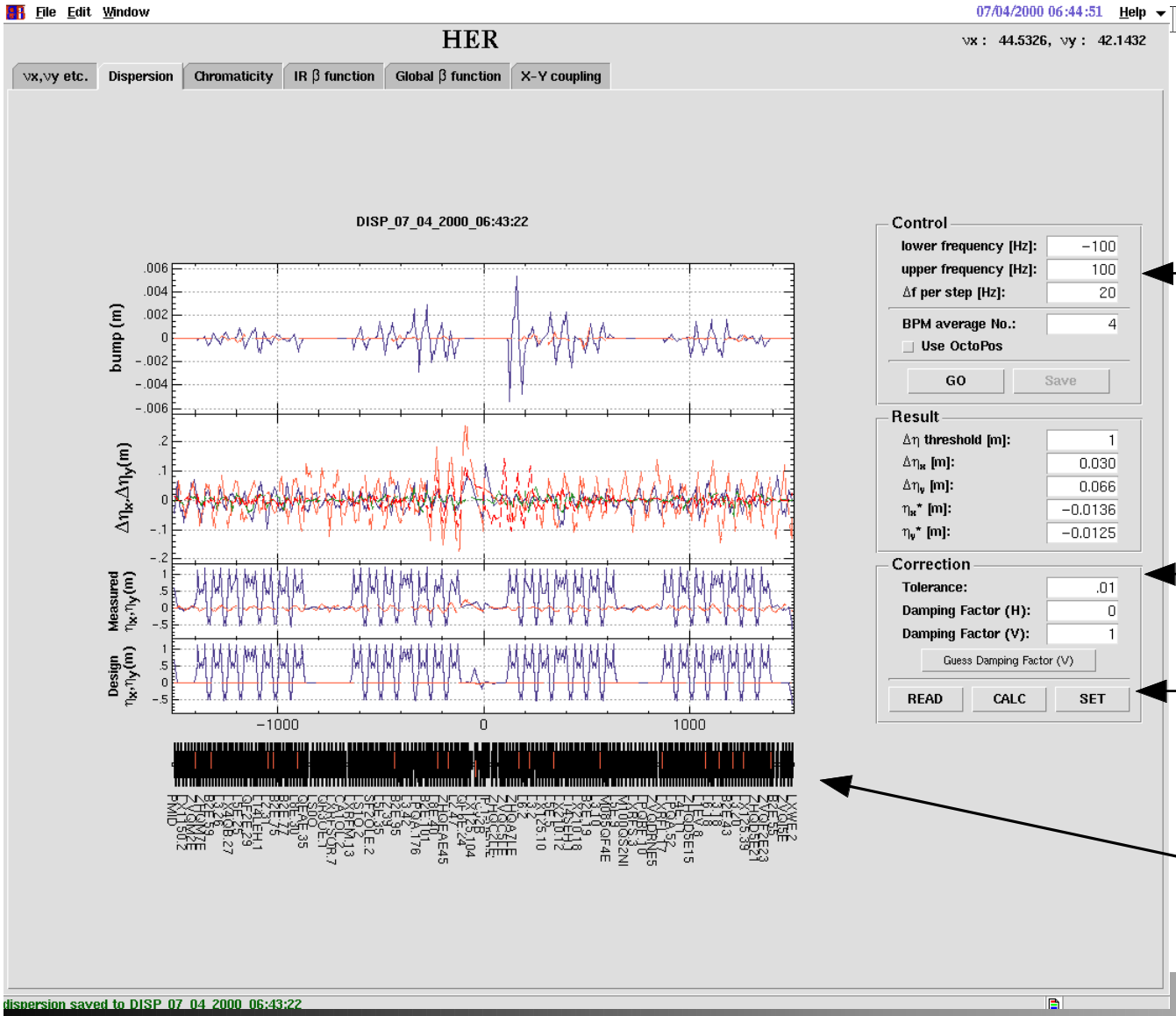
CCC: User Interface



Conclusion

- Control system/Software has hierarchical/layered structure
- Uniform access method, i.e. EPICS CA, reduces cost of software development.
- Generic EPICS tools can cover most of your needs.
- Create Components rather than Applications

Example of Higher Level Application



A dispersion correction panel in the KEKB optics correction application. SAD languages is used for this application.

Measure dispersion function changing RF frequency .

User can change conditions for dispersion correction.

Set New config to magnets using CA.

Drawn dynamically based on the lattice description in SAD format

Use of SAD/medm/Python in KEKB

Applications and Operator Display

	SAD	medm	python	misc*	Total
In Top level Applications	268	19	31	5	263

Number of Applications registered in the KEKB control application launcher.

Python:

Python

- Object Oriented Scripting Language
- Developed by Guild vonRossum @ CNRI, US(recently he and the development group moved to BeOpen.com)
- Free Ware : http://www.python.org
- Runs on Unix, Windows, Macintosh
- Has many extensions including Tkinter and PyGtk
- General Purpose
- Several books on Python available in the market.
  - http://www.python.org/psa/bookstore/
- EPICS/CA interface is developed (FNAL, Newton Group and KEK).

SAD:

SAD

- Accelerator Modeling program developed at KEK
  - http://acc-physics.kek.jp/SAD/sad.html
- Free
- Runs on Unix platform (Linux, HP-UX, Digital Unix) with gcc/g77.
- Beam line description
- Optics Matching
- Particle Tracking
- SAD script: Mathematica® type programing Language
- Tkinter : Interface to TkWidget for GUI
  - KBFrame : GUI Framework
- EPICS CA interface

SAD: EPICS CA interface

CaOpen[<Channel name> | <list of channel names>]

- returns Channel ID

CaRead[<Channel name> | <list of channel names>]

- returns a list { value, status, severity, EPICS time-stamp}

CaWrite[<Channel name or Channel ID>, Value]

or CaWrite[<list of Channel name or Channel ID>, <list of Values>]

- returns a list { value, status, severity, EPICS time-stamp} or list of these.

CaClose[<Channel name> | <list of channel names>]

- returns nothing

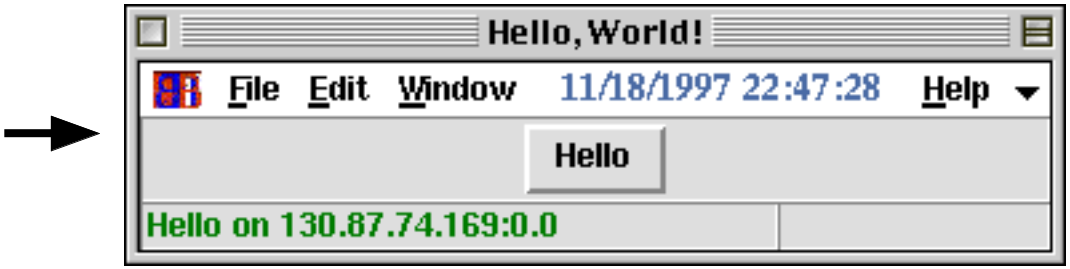
CaMonitor[<Channel Name>, ValueCommand:><Callback function>]

- Returns Channel Object

SAD: Tkinter/KEKBFrame

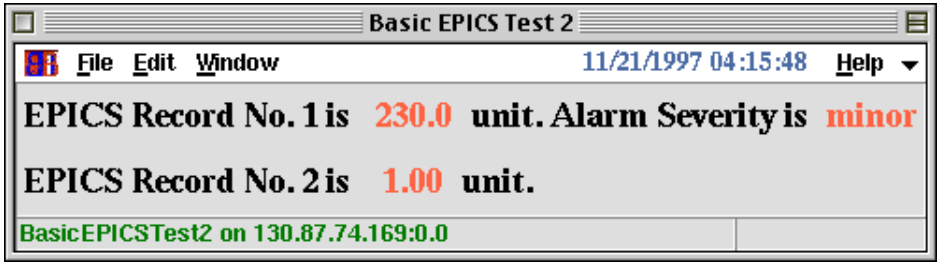
```
!Hello world in SAD Tkinter
w = Window[];
b = Button[w,
        Text -> "Hello",
        Command :> Print["Hello, World!"]];
TkWait[];
```


```
! Hello World in KEKBFrame
FFS;
w = KBMainFrame["Hello", f, Title->"Hello, World!"];
b = Button[f,
      Text -> "Hello",
      Command :> Print["Hello, World!"]];
TkWait[];
```



SAD: CA and KEKBFrame

```
w = KBMainFrame["BasicEPICSTest2", f, Title->"Basic EPICS Test 2"]
f1 = Frame[f, PadY->5, Fill->"x"];
f2 = Frame[f, PadY->5, Fill->"x"];
v1 = ""; s1 = "normal";
ch1 = CaOpenMonitor["some_record_name",
  ValueCommand:>(
    $FORM = "6.2";
    v1=ToString[ch1[Value]];
    $FORM = "";
    s1=Switch[ch1[Severity],0,"normal",1,"minor",2,"major"];
  )];
v2 = "";
ch2 = CaOpenMonitor["some_other_record_name",
  ValueCommand:>(
    $FORM = "6.2";
    v2=ToString[ch2[Value]];
    $FORM = "";
  )];
font = Font->TextFont["times","bold",18];
side = Side->"left";
l1a = TextLabel[f1, Text->"EPICS Record No. 1 is ", side, font];
l1b = TextLabel[f1, TextVariable:>v1, FG->"tomato", side, font];
l1c = TextLabel[f1, Text->" unit. Alarm Severity is ", side, font]
l1d = TextLabel[f1, TextVariable:>s1, FG->"tomato", side, font];
l2a = TextLabel[f2, Text->"EPICS Record No. 2 is ", side, font];
l2b = TextLabel[f2, TextVariable:>v2, FG->"tomato", side, font];
l2c = TextLabel[f2, Text->" unit.", side, font];
TkWait[];
```



Python:EPICS CA interface

_ca.c : C Wrapper over EPICS CA library

ca.py : Basic Python module for EPICS CA interface

cas.py: Another Python module for EPICS CA interface

```
#! python
# fred.py
import ca

ch=ca.channel("fred")
ch.wait_conn()
ch.get()
ch.pend_event(0.05)

print  ch.name, ch.val , ch.hostname
```

```
abco1.22: python fred.py
fred -0.10866663605 ahsad3:5064
```

Python: Tkinter/Pmw

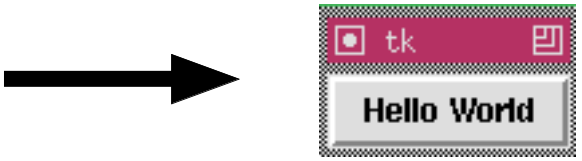
Tkinter: Tk Widget Interface

Pmw(Python Mega Widget): A set of Widgets using Python.

```
import Tkinter #  use Tkinter module

def main(): #define main routine
    b=Tkinter.Button(text="Hello World")
    def cb():
        print "What? "
    b.configure(command= cb)
    b.pack()
    Tkinter.mainloop()


if __name__ == "__main__":
    main()
```



Python:CA and Tkinter

```
import sys, ca # this program uses sys module from standard library and EPICS ca module
from Tkinter import * # import all names from Tkinter module
from CaVariableMixin import CaDouble # import CaDouble from CaVariableMixin module

class Simple(Frame): # define Simple class. It is a descendent of Frame class
        def __init__(self,name,master=None,*cnf): # object initialization method
                Frame.__init__(self,master)       # initialize parent object.
                self.pack(expand=1,fill='both')   # show this Widget on a screen
                self.var=CaDouble(name,master)    # assign instance of CaDouble class to instance variabl
                l=Label(self,text="CA readback",font="fixed",
                        textvariable=self.var)
                l.pack()       # create label widget and show it on a screen
                s=Scale(self,orient="horizontal", label=name, font="fixed",
                       variable=self.var,command=self.var.updateVar)
                s.pack(expand=1,fill='x') # create scale widget and show it on a screen
                self.s=s
                self.l=l

def test():
        f=Frame()
        f.pack(side="left")
        b=Button(f.master,text="Quit",font="fixed")
        b.config(command=b.quit)
        b.pack(side="bottom",fill="x",expand=1)
        for chan in sys.argv[1:]:
            Simple(chan).pack(side="left")
        f.mainloop()
test()
```