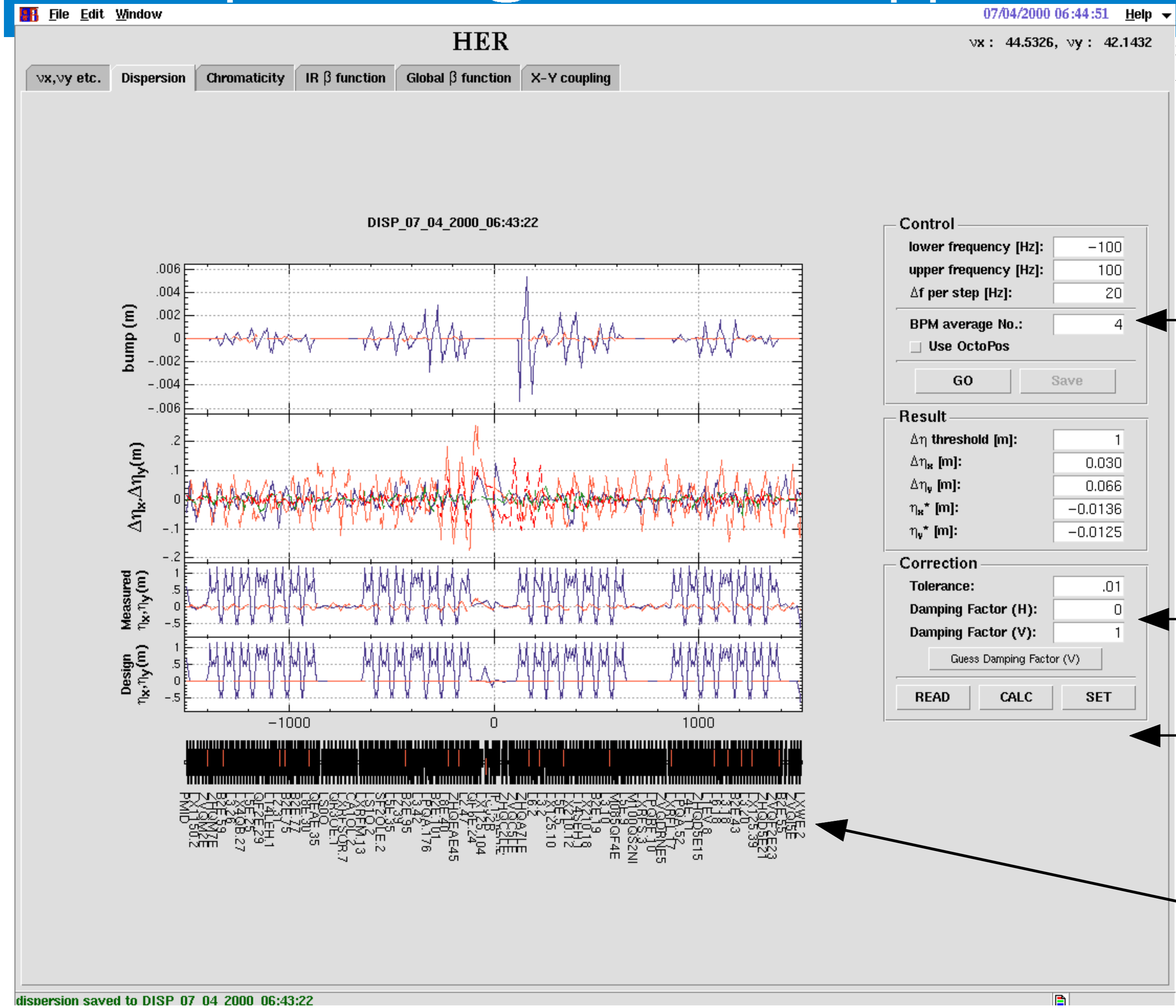
SAD/Python

IHEP EPICS seminar, Aug. 24, 2001

Noboru Yamamoto
KEKB control group

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

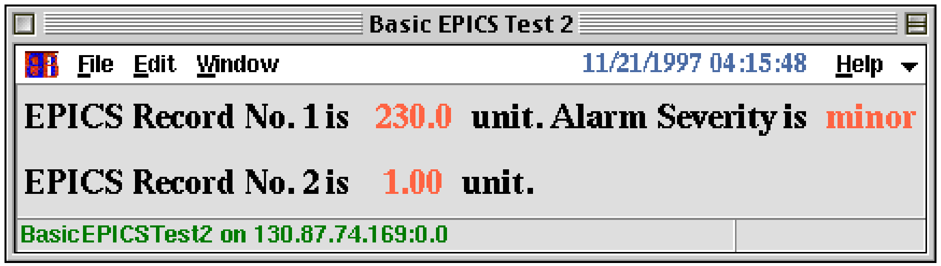
SAD: Tkinter/KEKBFrame

```
!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

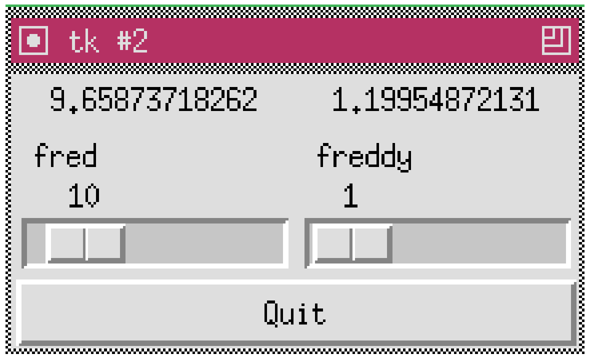
Tkinter: Tk Widget Interface

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 decendent 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()
```



Conclusion

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