SOLARIS
NATIONAL SYNCHROTRON
RADIATION CENTRE

JAGIELLONIAN UNIVERSITY
IN KRAKOW

JAGIELLONIAN UNIVERSITY
IN KRAKOW

SOLARIS
NATIONAL SYNCHROTRON
RADIATION CENTRE

An Implemetation of the Virtual Accelerator in the Tango Control System

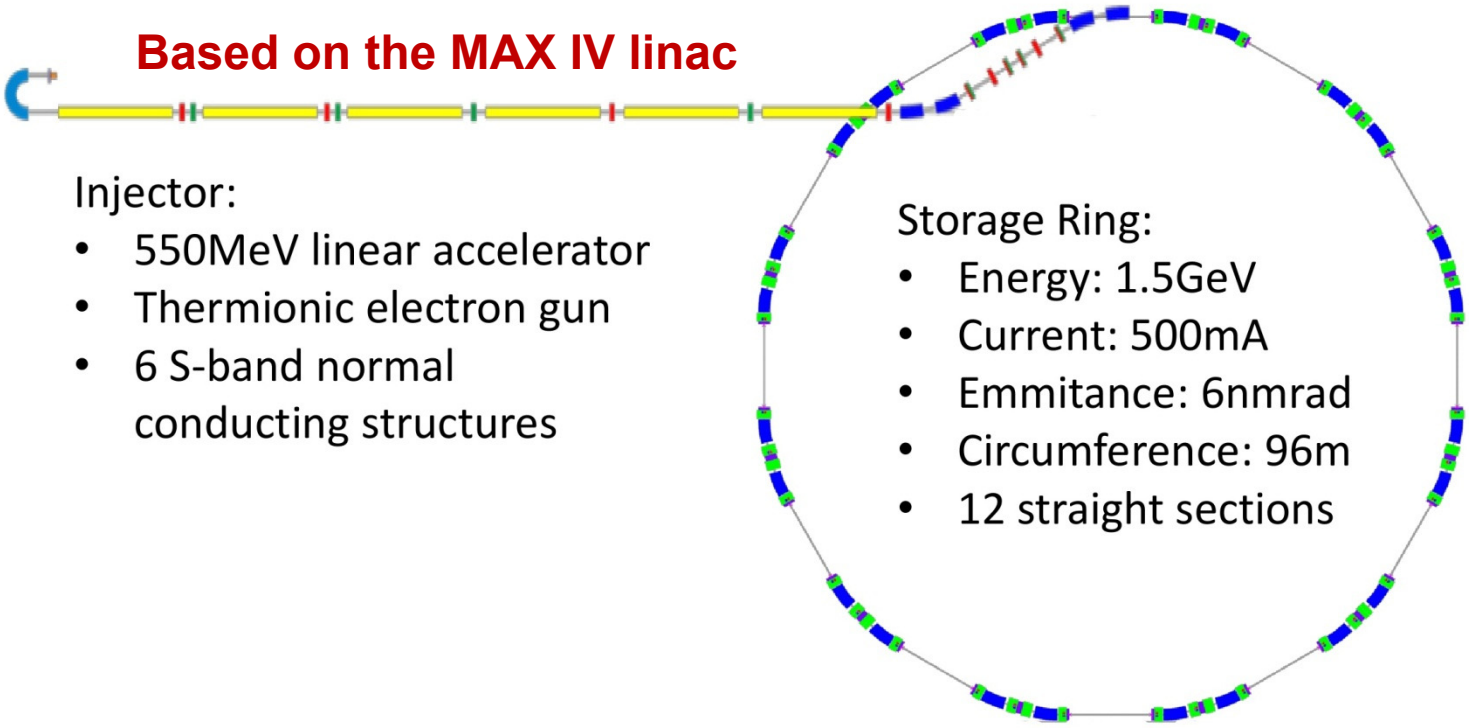
ICAP-2012, Rostock, 20-08-2012

Piotr Goryl
The Solaris, Jagiellonian University, Krakow, PL
The MAX IV Laboratory, Lund University, Lund, SE

The Solaris

The Solaris Machine

Based on the MAX IV linac



Injector:

- 550MeV linear accelerator
- Thermionic electron gun
- 6 S-band normal conducting structures

Storage Ring:

- Energy: 1.5GeV
- Current: 500mA
- Emmitance: 6nmrad
- Circumference: 96m
- 12 straight sections

The MAX IV 1.5 storage ring twin

Collaboration with the MAX IV Laboratory

The Project time frame

- Construction already ongoing, to be ready August 2013
- The installation expected to be finished in June 2014
- **First Light in September 31, 2014**
- **Operation**
- **Upgrades**
  - New beamlines
  - Top-up
  - Short Pulse linac option with potential FEL operation

The Solaris

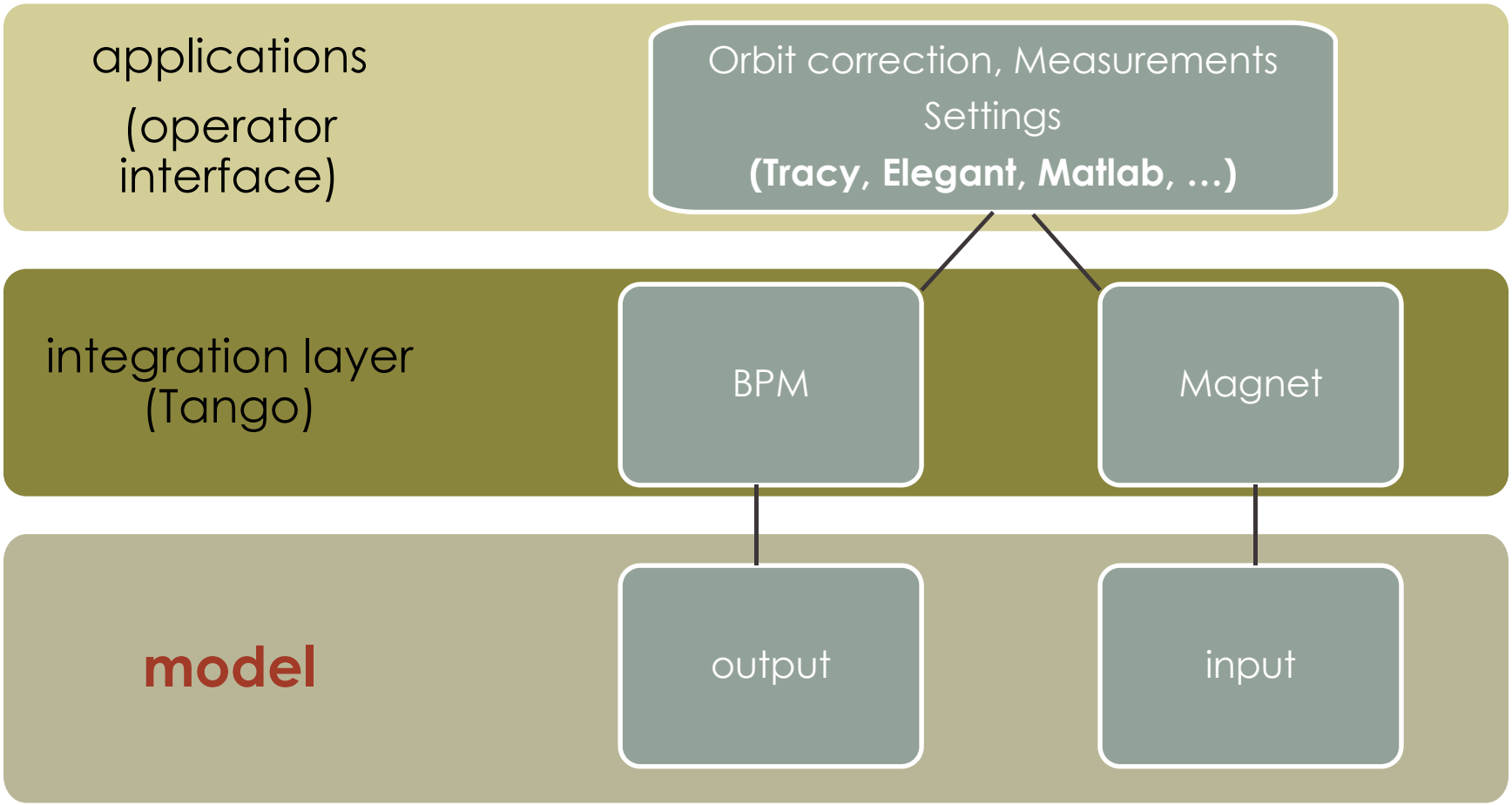
Beamlines

- Photoelectron Emission Spectroscopy (PEEM) – IKiFP PAN oraz AGH - project
    - Source: Bending magnet
    - Energy range: 200 – 1600 eV
- Ultra Angle Resolved Photoelectron Spectroscopy (UARPES) – JU – app. submitted
    - Source: Apple type undulator – variable polarization
    - Energy range: 8 – 100 eV
- X-ray Photoemission Spectroscopy (XPS) – Silesian University – app. submitted
    - Source: undulator
    - Energy range: 40-1500 eV
- Hard X-ray beamline – Poznan University – app. in writing
    - Source: SC 3-3.5T Wiggler
    - Energy range: x -15keV

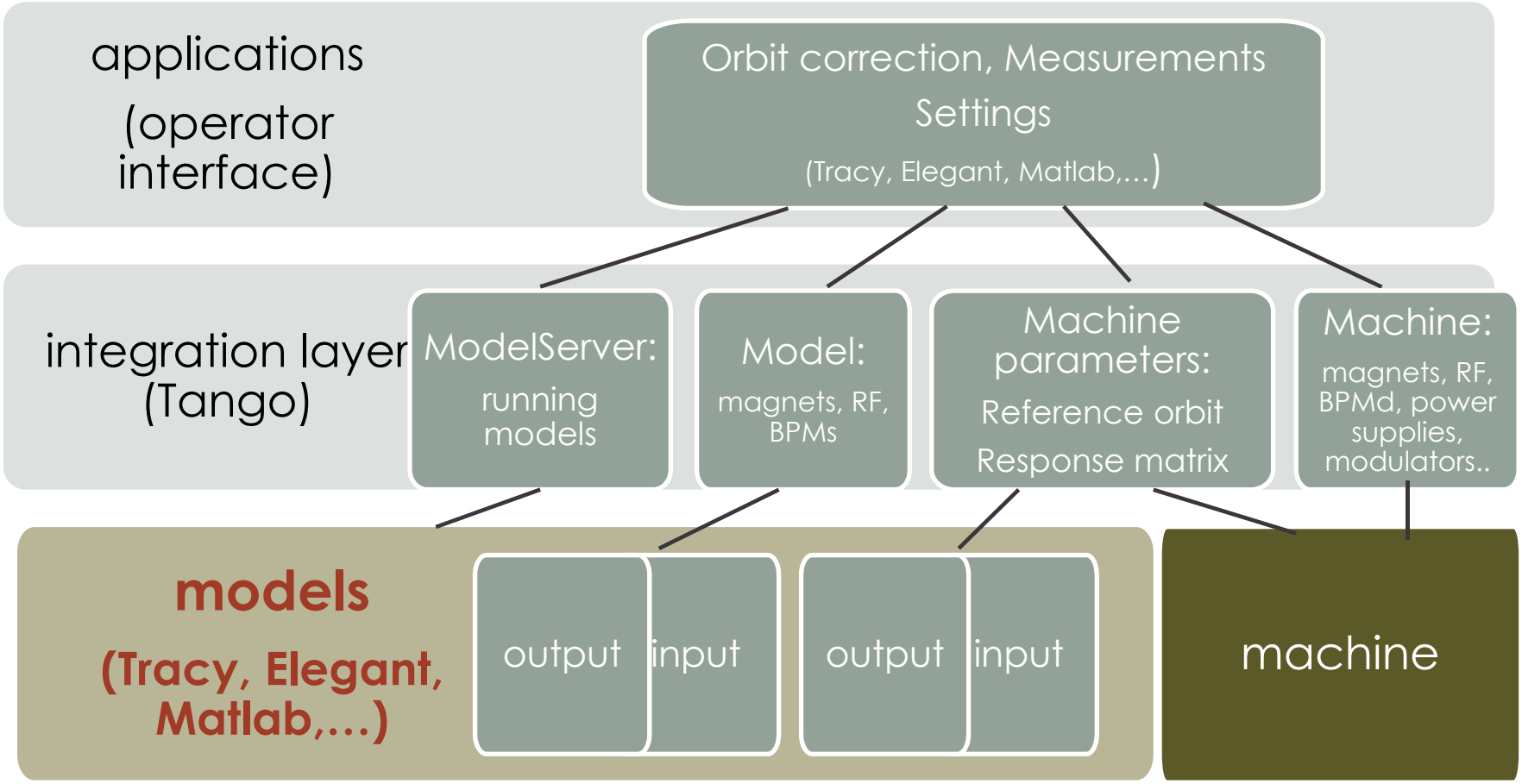
Virtual Accelerator

- **Integration of simulation code into the control system**
  - Run a computation
  - Provide an input to models
  - Get computation output
  - Keep track of a machine physics properties
- **Many simulation tools:**
  - Tracy
  - Elegant
  - Matlab
  - …

The VC idea


applications
(operator interface)
Orbit correction, Measurements
Settings
(Tracy, Elegant, Matlab, …)
integration layer
(Tango)
BPM
Magnet
model
output
input

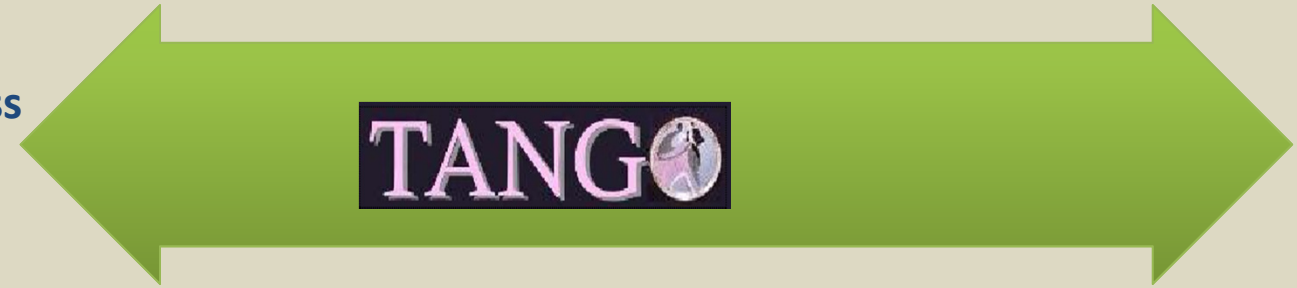
The VC idea


applications
(operator interface)
Orbit correction, Measurements
Settings
(Tracy, Elegant, Matlab,...)
integration layer (Tango)
ModelServer:
running models
Model:
magnets, RF, BPMs
Machine parameters:
Reference orbit
Response matrix
Machine:
magnets, RF, BPMd, power supplies, modulators..
models
(Tracy, Elegant, Matlab,...)
output
input
output
input
machine

Applications

Device:

- Object of a certain **class**
  - Attributes
  - Operations
  - Properties
- **Logical abstraction of hardware**

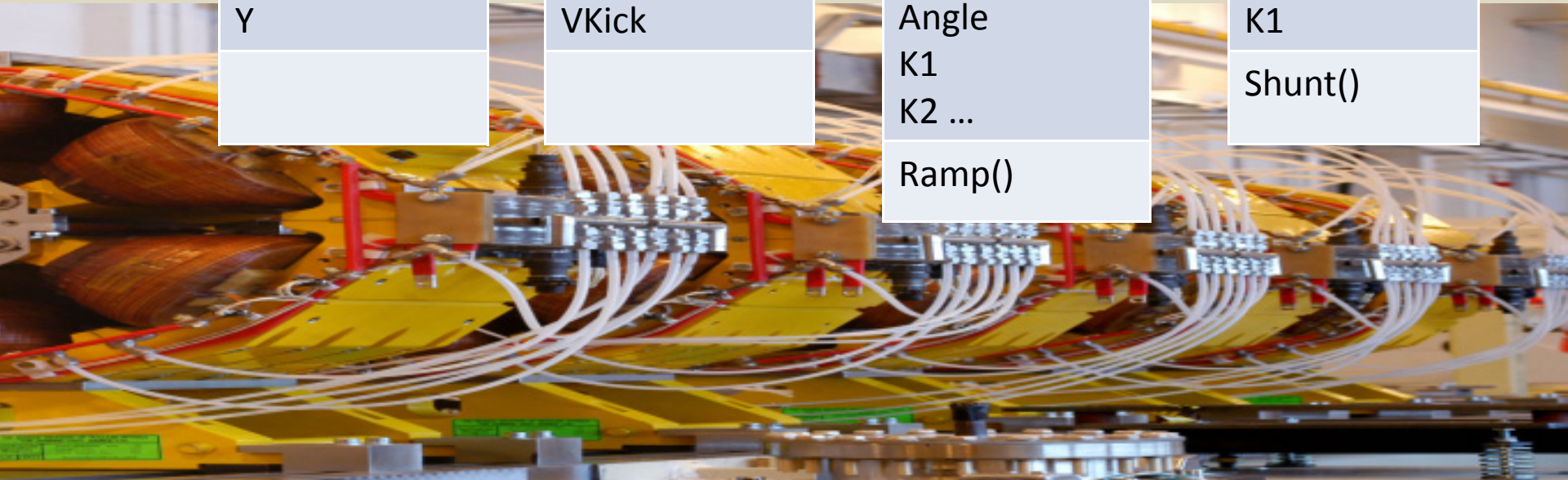


BPM
X Y

Corrector
HKick VKick

Dipol
L Angle K1 K2 …
Ramp()

Quad
L K1
Shunt()



Features

- **Integrates accelerator models into the control system**
  - To find operating points
  - To calculate machine's parameters
  - To keep track of the above (using CS tools)
  - **To help design the machine and the control system**
  - **For debugging**
- **Typical tools:**
  - **Tracy-3** – C/C++ library
    - Lattice file – a machine's description
    - Code using the library
  - **Elegant** - a computation program
    - Lattice file
    - Input file – what to calculate
  - **Matlab**
  - **….**

Modules

- **Virtual Accelerator**

  the whole idea/project

  - **ModelServer device**

    To let remotely run:

    - **Job**

      A program/script that can do something useful – dance in the rythm

    - **Online model**

      An accelerator model that do some useful computations using inputs from and outputs to the control system

  - **tango2elegant**

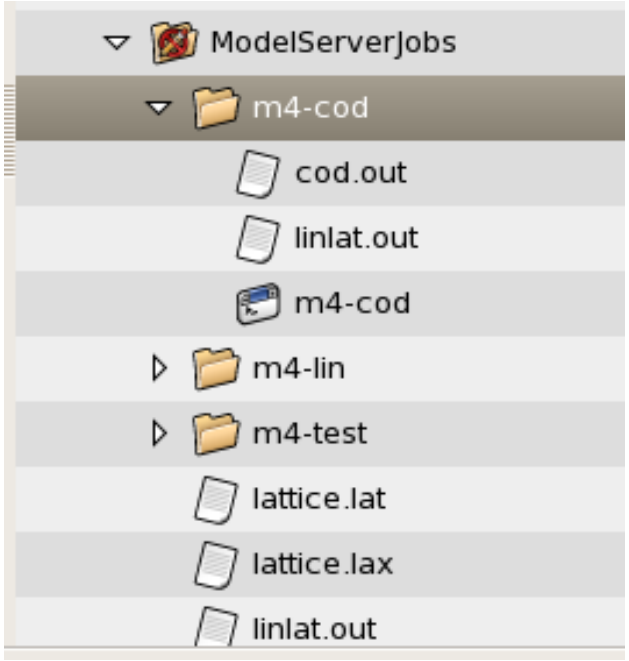
    A script that let the Elegant dance

  - **simple-tango**

    C/C++ API (library) that simplify talking with the Tango from C/C++ code (could be useful for the Tracy)

The Model Server

- Let run in a loop any arbitrary program or script (the Job)
- Let it to be run for certain number of iterations or until it is stopped
- Provides the Job with a lattice file if defined
- Provides the Job with a default tango device
- Let define tango attributes to be used by the Job
- Provides some status of the running Job



- Jobs are kept in a certain folder
- Jobs are started as a defined system user (now using a hard-coded *sudo* command)

- One ModelServer device can run only one of the Jobs at time
- One could run multiple devices to run Jobs concurrently

The Model Server

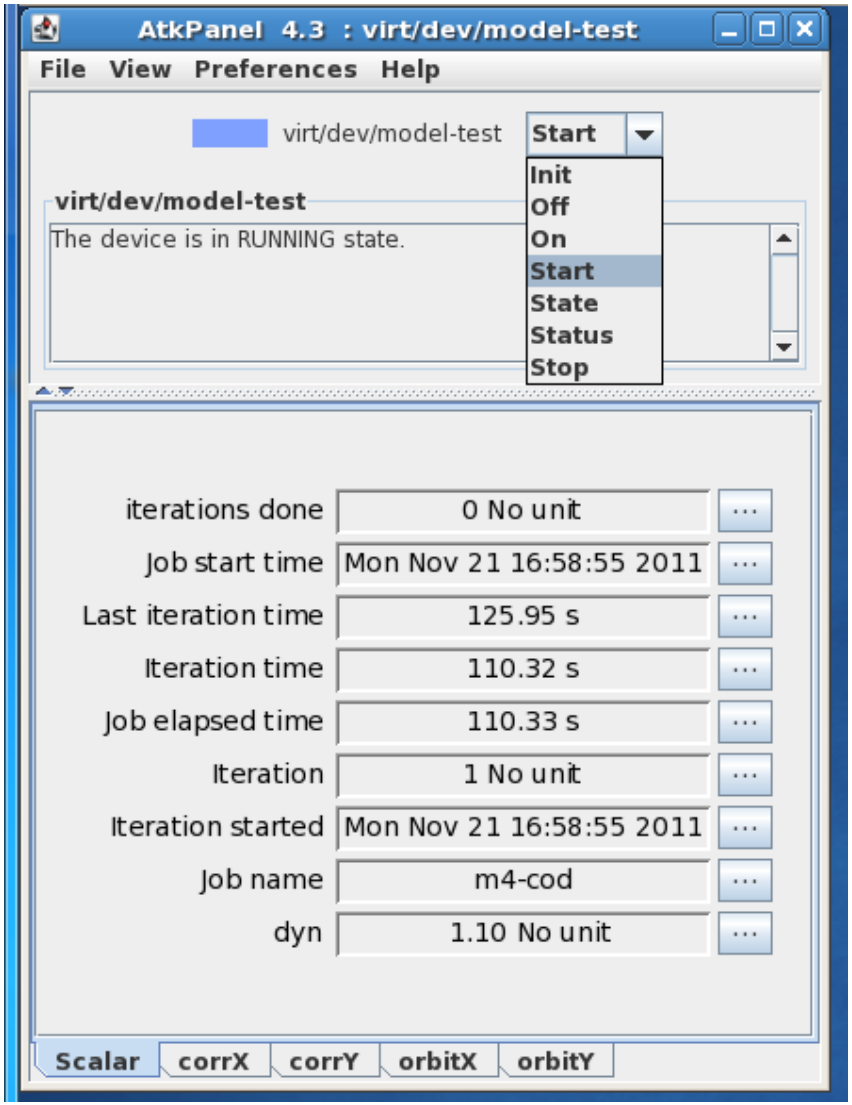
Device properties [virt/dev/model-test]

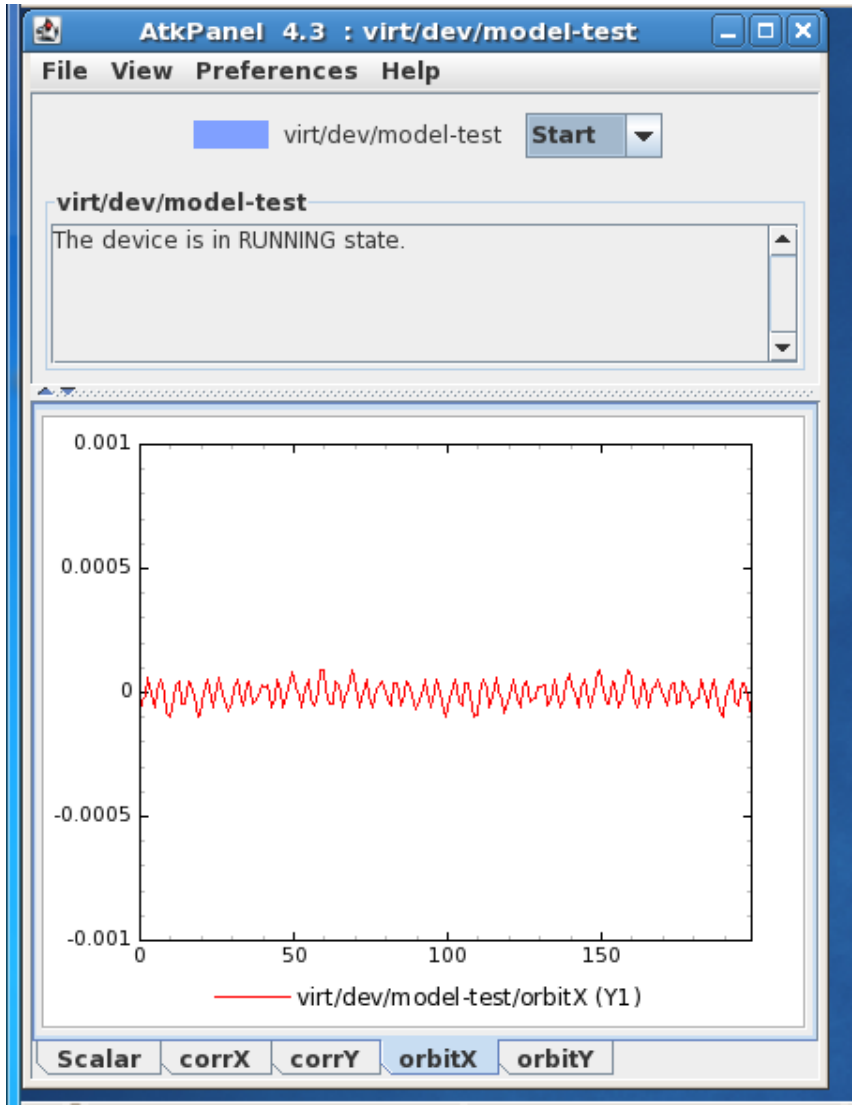
Property name	Value
DefaultJob	m4-cod
DynamicAttributes	dyn=DevDouble(1.1) corrX=DevVarDoubleArray(READ and (VAR('corrX' corrY=DevVarDoubleArray(READ and (VAR('corrY') orbitX=DevVarDoubleArray(READ and (VAR('orbitX orbitY=DevVarDoubleArray(READ and (VAR('orbitY
JobsPath	/home/tracy/ModelServerJobs
JobSystemGroup	tangosys
JobSystemUser	tracy
LatticeFileName	lattice.lat
OmmitLatticeFileExtension	True

JAGIELLONIAN UNIVERSITY IN KRAKOW

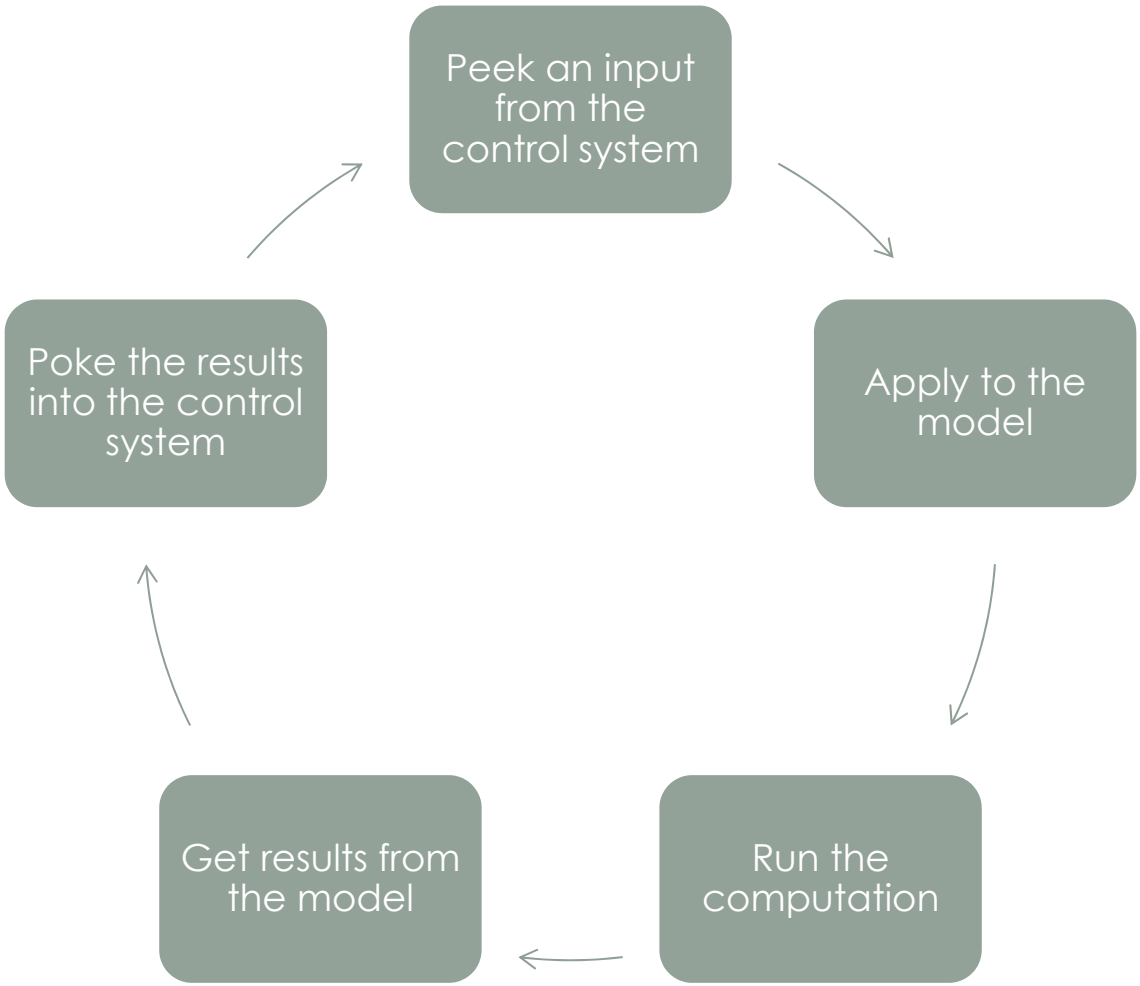
The Model Server

SOLARIS NATIONAL SYNCHROTRON RADIATION CENTRE





Jobs' rhythm


Peek an input from the control system
Apply to the model
Run the computation
Get results from the model
Poke the results into the control system

An Tracy example


```
double bpmsx[1000],bpmsy[1000],corrx[1000],corry[1000];

// read lattice
Read_Lattice(argv[1]); //sets some globval params

//get correctors' settings from tango
modelDevice=argv[2];
printf("Reading from TANGO...\n");

dim = 1000; //this is to denote size of our array
//read from tango
tango_read_spectrum_double(modelDevice,"corrX",corrx,&dim)
//apply to model
for (i=0; i<(unsigned int)globval.Cell_nLoc && j<dim; i++)
  if (strncmp(Cell[i].Elem.PName,"corr_h",6)==0) {
      set_dbnL_design_elem(Cell[i].Fnum, Cell[i].Knum, Dip, corrx[j], 0.0); j++;  }

// Do simulation / computations
Ring_GetTwiss(true, 0e-2); printglob(); //gettwiss computes one-turn matrix arg=(w or w/o c
globval.gs=ElemIndex("GS");  globval.ge=ElemIndex("GE");
getcod(0.0,lastpos);

// Provide output to TANGO
dim=0;
//get a calculated orbit
for (i=0; i< (unsigned int)globval.Cell_nLoc; i++)
  if (strncmp(Cell[i].Elem.PName,"bpm_m",5)==0 && dim<1000) {
      bpmsx[dim]=Cell[i].BeamPos[x_];
      bpmsy[dim]=Cell[i].BeamPos[y_];
      dim++;
  }
// write to TANGO
tango_write_spectrum_double(modelDevice,"orbitX",bpmsx,dim);
tango_write_spectrum_double(modelDevice,"orbitY",bpmsy,dim);
```

The Simple-Tango

- Access data from the Tango in a one line statement – for lazy dancers
- stango.h / stangolib.cpp => libstango.so.o

```
int tango_read_scalar_double(char *_dev, char* _attr, double* _data)
```

  - `int tango_read_spectrum_int(char *_dev, char* _attr, int* _data, unsigned int* _dim)`
  - `int tango_read_image_float(char *_dev, char* _attr, float* _data, unsigned int* _dimX, unsigned int* _dimX)`
  - `int tango_write_scalar_char(char* _dev, char* _attr, char _data)`
  - `int tango_write_image_short(char* _dev, char* _attr, short* _data, unsigned int _dimX, unsigned int _dimY)`
  - `int tango_run_command(char *_deviceName, char *_command)`

- stangoimpl.h / stangoimpl.cpp
  - Internal functionalities:
    - Templates providing proper casting
    - class STango – for keeping a list of DevicePoxy-ies (as a static dictionary member)

The Elegant case

1. Write a script that reads from the Tango and write to a SDDS file
2. Modify an elegant input file to load the SDDS file
3. Run the Elegant
4. Write a script to read from the output SDDS files and to send results to the Tango

Different jobs would need different input/output scripts

OR ?

The tango2elegant

The elegant input file is a list of commands to be executed – there are commands that enable runtime changes to a model:

```
&alter_element
   name = VC
   item = Kick
   value = .2
&end
```

tango2elegant – parse an input file searching for special commands: `&tango_input` and `&tango_output` (these are arbitrary defined by me)

If the `&tango_input` is found it is replaced with with proper elegant's commands to modify the model according to the values from the control system (attributes)

If the `&tango_output` is found it is not passed to the elegant. However, the script get values from output files and modify attributes in the control system

The tango2elegant


```
>tango2elegant max4.ele | elegant -pipe=in
```

Elegant input file:

max4-tango.ele [Read Only] (/home/tracy/ModelServe

File Edit View Search Tools Documents Help

max4-tango.ele

```
    clear_error_settings = 1
&end

&tango_input
  tango_input_mode=spectrum,
  tango_path=virt/dev/model-test,
  tango_attribute=corrY,
  name = VC,
  item= KICK,
  value = %value%,
  start_occurence = %index+1%,
  end_occurence = %index+1%,
  verbose=1
&end
```

Input from the Tango

max4-tango.ele

```
&tango_output
    tango_device=virt/dev/model-test,
    tango_attribute=orbitY,
    tango_output_mode=spectrum,
    element_type=MONI,
    column=Cy,
    file=../output/stdin.cen,
&end

&tango_output
    tango_device=virt/dev/model-test,
    tango_attribute=orbitX,
    tango_output_mode=spectrum,
    element_type=MONI,
    column=Cx,
    file=../output/stdin.cen,
&end
```

Output to the Tango

Status

- All modules ready
  - some new features required
  - some bugs revealed
  - documentation needs refinement
- Deployment on the PLGrid infrastructure (now in a test phase)
  - A virtual server with the Tango CS – it will be populated with the Solaris cs structure
  - The Tango libraries and tools available on a one of the clusters (ZEUS in the ACK Cyfronet in Krakow)
  - The ModelServer suffer from a bug in the *sudo* tool on the Scientific Linux 5 - migration to the SL 6 and parallel reimplementation to enable direct use any other 'starter' (like the Unicore ucc)
- Future
  - Deployment for the MAX IV linac and storage rings to test CS scripts and applications – this autumn
  - Improvements to the tango2elegant (dealing with images from SDDS files)
  - A GUI for the ModelServer
  - An event driven iteration starting
  - An educational deployment?
  - Lattice files management?

Thank you

References

www.tango-controls.org

MAX-lab

elettra

ALBA

DESY

SOLEIL SYNCHROTRON

ESRF

Forschungs-Neutronenquelle Heinz Maier-Leibnitz (FRM II)

PL GRID