Modern Accelerator Control Systems

Kazuro Furukawa, KEK
for KEKB Control Group
and Linac Control Group

<kazuro.furukawa@kek.jp>
Accelerator Controls at KEKB and Linac
Operational Software
Considerations on Accelerator Controls in General
Available Technologies
Adaptive Reliabilities
Summary
KEKB and Linac
Control Systems in KEK

- **Operational Presently**
  - Linac, PF, PF-AR, ATF, KEKB

- **Under Construction**
  - J-PARC, STF

- **EPICS**
  - KEKB, …
Increase of the Luminosity

**KEKB and Linac Accelerator**

**Modern Accelerator Controls**

**PAC 2007, Albuquerque, NM, US**

**Jan. 2007**

**Increase of the Luminosity**

**May. 2000**

**Dual Bunch e+**

**Apr. 2003**

**Feb. 2005**

**Continuous Injections**

**Now**

**Collision with Crab Cavities**

Kazuro Furukawa, KEK, Jun. 2007
KEKB Control System (Hardware)

◆ **GbE Fiber Optic Networks**
  - Single Broadcast Domain
  - Central Control Room and 26 Local Control Rooms

◆ **VME/IOC**
  - ~100 VME/IOC mostly with PowerPC CPU

◆ **Field bus**
  - ~200 VXI thru MXI for BPM Instrumentations
  - ~50 CAMAC for rf and Vacuum (inherited from TRISTAN)
  - ~200 ArcNet network segments for Magnet Power Supplies, and other field Controllers
  - GPIB for Instrumentations, RS232C, Modbus+ for PLCs

◆ **Host Computers**
  - HP-UX/PA-Risc, Linux/x86 Controls Server
  - 3 Tru64/Alpha with TruCluster
  - Several Linux
  - Many MacOSX
  - (Solaris/Sparc for VxWorks)
KEKB Control System (Software)

- **EPICS 3.13.1 and 3.14.6,8**
- **VxWorks 5.3.1 mainly, and 5.5.1**
  - Hope to upgrade EPICS/VxWorks Shortly
- **IOC Development**
  - CapFast, (VDCT) Perl, SADscript for Database Configuration
  - Oracle as a backend Database Management
    - Migration towards Postgresql
- **Operational Application Development**
  - MEDM(DM2k) for Startup
  - Python/Tk for Equipment Controls
  - SADScript/Tk for Beam Operation, etc
KEKBLOG and ZLOG

◆ KEKBlog/kblog Archiver is Used from the Beginning of the Commissioning
   - Just less than 2GB / day
   - Several Viewer Tools
     - Very often Used to Analyze the Operation Status

◆ Zlog Operation Log
   - Zope, Python, PostgreSQL
     - Most of the operation logs
     - In Mostly Japanese
     - Figure Storing Integration
       - ex. Screen shot of operational Panels
**Linac; Physical Structure**

◆ Multi-tier, Multi-hardware, Multi-client, …

![Diagram showing the physical structure of a linac system]

- X-Window Interface for Operation
- Touch Panel Interface
- MS-Window Interface
- EPICS Gateway to KEKB-Ring
- Central Network (Gb-Ethernet)
- Main Computer Systems (Unix)
- Interface to other Facilities
- Equipment Level Network (Optical Ethernet)
- VME’s (~30)
- PLC’s (~150)
- CAMAC’s (~15)
- VXI’s (~30)
- PC / GPIB / RS232C
- Beam Monitor Timing etc.
- RF Magnet Vacuum
- Timing
- RF Monitor
- Gun Beam Monitor others
Linac; Multi-tier Logical Structure

Advanced Beam Operations & Beam Study

Upper Level Servers

Engineering Operations

Middle Level Servers

Lower Level Servers

Network Based Hardware Controllers

Accelerator Equipment

Routine Operations

Electron / Positron Beams
Software Architecture

- **Base control software structure for Multi-platform**
  - any Unix, OS9, LynxOS (Realtime), VMS, DOS, Windows, MacOS
  - TCP - UDP General Communication Library
  - Shared-Memory, Semaphore Library
  - Simple Home-grown RPC (Remote Procedure Call) Library
  - Memory-resident Hash Database Library

- **Control Server software**
  - Lower-layer servers (UDP-RPC) for control hardware
  - Upper-layer server (TCP-RPC) for accelerator equipment
  - Read-only Information on Distributed Shared Memory
  - Works redundantly on multiple servers

- **Client Applications**
  - Established applications in C language with RPC
  - Many of the beam operation software in scripting language,
    - Tcl/Tk
    - SADscript/Tk
Operation
KEKB Commissioning Groups

◆ Formation of Commissioning Group (KCG)

- Linac Commissioning (LCG)
  - 7 from Linac
  - ~10 from Ring

- KEKB Ring Commissioning Group (KCG)
  - All LCG
  - ~20 from Ring
  - Several from Detector (BCG)

- Commissioning software base was formed during Linac Commissioning (1997~)
  - Tcl/Tk, Python/Tk, SADscript/Tk
SADScript

◆ Mathematica-like Language
  ◆ Not Real Symbolic Manipulation (Fast)
  ◆ EPICS CA (Synchronous and Asynchronous)
    CaRead/CaWrite[], CaMonitor[], etc.
  ◆ (Oracle Database)
  ◆ Tk Widget
  ◆ Canvas Draw and Plot
  ◆ KBFrame on top of Tk
  ◆ Data Processing (Fit, FFT, …)
  ◆ Inter-Process Communication (Exec, Pipe, etc)
    System[], OpenRead/Write[], BidirectionalPipe[], etc.
  ◆ Greek Letter
  ◆ Full Accelerator Modeling Capability
  ◆ Also Used for non-Accelerator Applications
  ◆ Comparable to XAL, but very different architecture
Virtual Accelerator in KEKB

◆ For Example in KEKB

❖ Most Beam Optics Condition is maintained in the Optics Panel
❖ Other Panels Manipulate Parameters Communicating with the Optics Panel

(Oide, Koiso, Ohnishi et al)

Tune Measurement/Changer

Optics Panel
Beam Optics Database

◆ Repository of Inputs to Simulation Codes?
◆ XSIF Extended Standard Input Format
  ❖ Many Simulation Codes utilize it
  ❖ SAD does not
  ❖ Currently a Conversion Tool is Used to for These Input Formats
  ❖ XSIF (LIBXSIF) inclusion in SAD?
◆ Yet another Generalized Input Format?
  ❖ Separation between Beamline Geometry (relatively static) and Beam Optics (more varying)
  ❖ Could be structured into XML
◆ Relational information to each Hardware Components
  ❖ We do not prefer complicated relations
Accelerator Controls
Accelerator Controls

Definition and goal

- Specified only after technical details of the accelerator is decided
  - Of course the final goal is the science achievement
- Often change after commissioning
  - Many prefer to flexibility as well as to robustness (depending on the purpose)
  - Should support rapid development to realize novel ideas immediately
- Unfortunately we don’t have general accelerator controls
  - We may have to make something
History

◆ Discussion of accelerator controls
  ❖ At ICALEPCS conferences
    ⊹ After some success of NODAL at SPS/CERN
    ⊹ Needs for more general software tools
  ❖ NODAL was chosen at TRISTAN
  ❖ SLC/SLAC used Micros + VMS
  ❖ Standard model
    ⊹ Field-network + VME + Unix + X11
  ❖ Software sharing
    ⊹ Definition of a Class to represent whole accelerator
      ♦ Which was impossible
  ❖ More common control system with extended API
    ⊹ ncRPC/CERN, TACL/CEBAF, ACNET/Tevatron, etc
    ⊹ EPICS got popular maybe because of the selection at SSC, APS, CEBAF, BESSY, …
  ❖ Then more object oriented software (naturally after RPC)
    ⊹ More computer aided development possible
    ⊹ CICERO/CERN, TANGO, CORBA+Java, CERN, …
    ⊹ Windows/Microsoft, …
No common controls yet

- Balance between many available technologies

- **Object-oriented vs. Channel-oriented**
  - **Object-oriented technology**
    - More support benefits from software engineering
    - Extendable, clearer definitions
    - Different people have different ideas on control objects
  - **Channel-oriented technology**
    - Flat (one-layer structure), simple, scalable
    - Not much support from software engineering
    - Easy to make gateways
More balances

- Compiled language vs. interpretive language
  - Two level languages
    - Interpretive language for rapid prototyping
    - Compiled language for established algorithms
  - After too much success of NODAL
  - Compiled languages programmed by expert
    - Documentation, maintenance, policy-driven
    - Manageable, then reliable
  - Interpretive/scripting languages
    - Rapid development
      - Realization of novel ideas in hours
    - Everyone attends the construction of operation environment
    - Another level of management/maintenance required
More balances

◆ Best & aggressive vs. moderate & conservative

- New technology is attractive
  - But can be a “fad”
  - Can we justify the choice?

- For longer life-span, which is better?
  - Life of accelerator is often very long compared with
    - User facilities
    - Commercially available software/communication technologies

- Operational performance continuously advances

- Accumulation of operation knowledge base
  - Stored mainly as software and database in the control system
    - Beam stabilization algorithms, hardware startup procedures, etc

- It is valuable treasure
  - There should be mechanism to keep such resources
    - With longer life-span
More balances

◆ International vs. de-facto standards
  ◆ International organizations pursue ideal solutions
    ✰ Sometimes they don’t become de-facto standards
    ✰ Selection of one of many standards is difficult
  ◆ Watching the market
    ✰ TCP/IP network, Unix/Windows operating system, VME boxes
  ◆ Advantages of de-facto standards
    ✰ Economical advantage to select products out of markets
    ✰ Save man-power avoiding proprietary development
    ✰ Solutions will be provided for the old standard in the next generation
    ✰ As a whole, it is good for long life-span
Available Technologies
Programmable Logic Controllers (PLC)

- Rule-based algorithms can be well-adopted for simple controls
- IP network for the both controls and management were preferable
  - Especially at KEK/Linac which has a policy of IP only field network
- ~150 PLCs at Linac since 1993, and also many at J-PARC
- Isolated/separated development becomes easy
  - Outsourcing oriented
- Equipment developer oriented
  - Many maintenance capabilities were implemented
- IEC61131-3 Standards
  - 5 languages, with emphasis on naming
  - Not so popular in Japan
  - Effort to make common development environment
  - XML representation of resources
  - Should be paid more attention
- Redundancy
Network with only IP/Ethernet

◆ The policy chosen when we upgrade Linac in 1993

- Make network management simpler
  - Faster switches, routing, network-booting, etc.
- Avoid Hardware failure and analysis effort with old field network
  - Home-grown field networks need much dedicated man-power
- Cost for optical Ethernet went down at around 1995
  - Linac has high-power modulator stations, noise source
- Nowadays many facilities have this policy with GbE
  - J-PARC controls basically followed this
- More and more intelligent network devices
  - ex. Oscilloscopes with Windows/3GHz-Pentium built-in
  - Even EPICS IOC, MATLAB, or others can be embedded
- Network components can be replaced one-by-one
- Security consideration will be more and more important
FPGA

◆ Another “everywhere” after IP network
  ❖ Digital circuit and software can be embedded in to one chip
    ✤ Even CPU core is embedded
    ✤ Flexible and robust, wonderful platform for local controls
      ✤ Sometime terrible source of bugs
  ❖ Nano-second level timing
  ❖ More and more gates, memory, pins, etc
  ❖ More software support
ATCA and μTCA

Advanced telecommunications computing architecture

- Accommodate several 100ohm serial buses
- GbE or PCI-express, 10GbE, etc
- Typically 14 slots in 19” and 12-unit height
- Shelf manager manages healthiness of the system
  - through Intelligent Platform Management Interface (IPMI)
- Many reliability improving facilities, redundancy, hot-swap, etc

MicroTCA

- More recently defined in 2006, based on AdvancedMC Mezzanine Card defined in ATCA
- Begin to have many facilities from ATCA
EPICS

◆ Now is a kind standard, but …

◆ Object-oriented design support
  ◆ Naming scheme, and/or design of new record
  ◆ More software-engineering support favored
    ☐ Several different efforts to provide better environment
      ◆ Java IOC (M. Kraimer), Control system studio (M. Clausen), Data access (R. Lange)

◆ Security mechanisms
  ◆ User, Host-based protection available
  ◆ More security
    ☐ Dynamic controls of security
    ☐ Access logging

◆ Dynamic configuration of database
  ◆ Dynamic creation / loading of records
  ◆ Dynamic removal of records
    ☐ Maybe some part of the codes can be shared with redundant-IOC project
Magnet Controls

◆ It is typical controls and still many things to do
◆ Many magnets and many power supplies
  ⊳ No one-to-one correspondence
  ◆ Which hardware interface to use
◆ Procedures
  ◆ Interlock status, on/off, analog with some precision, etc
  ◆ Energy, kick - field - current conversions
    ⊳ How to represent those conversion curves
  ◆ Timing synchronous operation
    ⊳ for tune change, orbit correction, etc.
  ◆ Standardization
Timing Event System

Present Timing System
- Provides ~3pico-second Timings to ~150 Devices
- Only 4 Events can be Distinguished
- VME(x6) and CAMAC(x10)

Diamond Event System
- Single Fiber can Transfer Clock, Delayed-Timings, Events (256), Data Buffers (2k-bytes)

New IOC
- MVME5500
- RTEMS (developed at BNL)
  - (May migrate to VxWorks if KEKB upgrades VxWorks)
- EPICS Driver/Device Support from SLS/Diamond/SLAC/LANL
Reliability
Reliability

◆ The end user expect rigid reliable operations

◆ Inner layers need flexibilities
  ✷ Because of daily improvement
  ❖ Compromise between
    ✷ Practical or ideal solutions
    ✷ Aggressive and conservative
    ✷ Under restrictions of
      ◦ Time, safety, budget, man-power
  ❖ Here we think about adaptive reliability
Reliability Increase without much Cost

◆ There should be “right way”
  ❖ We hope to have it some day, but for now we need interims

◆ Surveillance for everything
  ❖ Well-arranged system does not need this, but…

◆ Testing framework
  ❖ Hardware/Middleware tests just before Beam
  ❖ Software tests when installed

◆ Redundancy
  ❖ In Many Hardware/Software components
  ❖ Of course some of them are Expensive, but…
Surveillance for everything

◆ We have written too many pieces of software
  ❖ which assume certain circumstances unfortunately
    ▷ which will fail some day
  ❖ in scripting languages too rapidly and too easily
    ▷ without documentations

◆ We manage too many computers
  ❖ If only one, I’m almost sure I can make it stable
    ▷ But in reality even hostname can be mis-labeled

◆ We installed too many network components
  ❖ without good network database etc
    ▷ which sometimes has bad routing information, etc
Surveillance for everything

-if certain installation of (software/hardware) was not ideal-

-find out

-what is the most important feature of the installation?
-what is the easiest test for its healthiness?

-routine test is carried automatically

-by cron or continuous scripts

-if an anomaly found,

-alarm, e-mail to the author, make error log
-restart related software, if not critical
-report to the human operator, if critical

-not ideal, but effective under limited human resources
Software Testing

Moving operating environment

- For better resource performance
  - We tend to do it because of the pressure from budget restrictions
- May lead to malfunctions
  - We knew they may happen

Automatic software (hardware) tests preferable

- Under new environment (machine, compiler, network, etc)
  - Many kinds of important free software does them
  - Language systems, Linux Test Project

We do some tests

- But sometimes not enough
- More thoroughly prepared tests needed
Testing Framework

◆ When we introduce new environment
  ❖ Unit test
    ✂ We don’t do it much yet
    ✂ EPICS began to have it, “make runtests”
      ✪ Collecting existent test cases
      ✪ User can provide tests in Perl/Test framework
    ✂ Hope to have for SAD and SADscripts
  ❖ Regression tests
    ✂ We have something, but not thorough, not exhaustive
    ✂ Difficult to collect cases
  ❖ Stress tests
    ✂ We do it during operation (?)
    ✂ We know computers rarely fail, but network/network-devices do
      ✪ Find solution
      ✪ Development of surveillances
      ✪ Installation of failure-recovery or failover procedures
Testing Framework

◆ When we start new run
  ❖ New software/hardware
    ❙ We test unit by unit
    ❙ But not through operational tools prepared
  ❖ Maintenance works
    ❙ We often forget to restore/initialize cables, switches, variables
    ❙ Power-stop may bring another annoyance

◆ We need routine procedures which include
  ❙ Hardware tests
  ❙ Name/ID matching
  ❙ Database tests
  ❙ Software component tests
  ❙ Software/Hardware simulation tests

❖ Before beam operation
❖ We do it mostly by operator observations based on written procedures
❖ CERN did some efforts
Redundancy

Do we need redundancy?
- Redundancy may be the last-resort measure
- It may cost
- Centralized facilities are easier to manage
  - If I have only one server, my life is much easier
- But they become complicated monsters
  - Nobody understand everything

Especially useful for maintenance
- Not only for failure-recovery
  - Redundant systems of complicated system; (complicated)^2

Anyway we may have to prepare backups
- Then automatic failover is just around the corner
- And …
File server redundancy

- RAID and Mirror-disks are used everywhere now
- We began to use Cluster software before KEKB
  - DECsafe, TruCluster for Unix
  - LifeKeeper, Redhat-AS, Rose-HA for Linux
  - NetApp
- It works at least for Hardware troubles; but sometimes for Software troubles
- Maintenance and Scheduling became easier
Network Redundancy

- Mostly established technologies
  - Wide acceptance of Ethernet and IP
  - > 10 years ago
    - Redundant Transceivers
  - More recently Standards available
    - Hsrp or Vrrp and Rapid spanning tree
Redundant PLC’s

◆ CPU built-in redundancy is already used in several vendors
  ❖ Dual main memory with checksum at every-cycle
  ❖ ROM as well as flash memory
    ☐ Bad circumstances at field forced them to implement it

◆ We just started to evaluate redundant CPU’s

◆ Redundant PLC’s are used at CERN
  ❖ Siemens S7, slightly expensive

◆ Several possibilities in architecture
  ❖ Single vs. dual backplane
  ❖ Power-supply, CPU, Network-interface
  ❖ I/O (?)
Redundant EPICS IOC

◆ Redundant controllers are favorable
  - as in PLCs
  - The project was started at DESY (M. Clausen)
    - Redundancy monitor task (RMT)
      - Monitors healthiness of controllers
      - Manages primary redundancy resource (PRR)
    - Continuous control executive (CCE)
      - Synchronizes internal states
    - Modifications for several others PRR’s
      - Scan tasks, Channel access server tasks, Sequencer, Drivers
      - Possibly user tasks
  - KEK joined in for wider applications
    - Linux (OSI) port
    - Gateway applications
  - ATCA implementation possible
    - For ILC (?), microTCA (?)
Software redundancy

◆ EPICS IOC redundancy is slightly complicated
  ❖ Since it has name resolution facility
  ❖ More advanced

◆ Linac/KEK controls is simpler
  ❖ Normally we run several middle-layer control servers
    ♦ on separate machines
  ❖ For EPICS gateway
    ♦ We need redundant IOC technology

◆ Other existent servers
  ❖ Recently more careful in redundancy
    ♦ Like dchpd
    ♦ Redundancy and replications
Summary
Phronesis

◆ Aristotle’s view of wisdom.
◆ Contrary to Sophia; the ability to understand the universal truth
◆ Phronesis is the ability to find a way to achieve an overall goodness
Summary

◆ EPICS and SAD made KEKB a great success, but other accelerators have different criteria
◆ Accelerator controls design needs a balance between many aspects
◆ There are many good technologies waiting to be utilized
◆ Also more reliability features needed
◆ Share more experiences
◆ Phronesis
Thank you