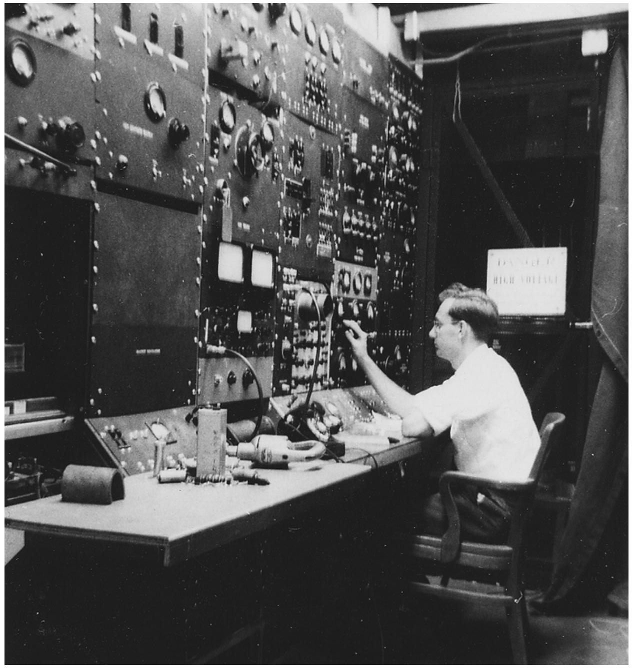
Status and Future Developments in Large Accelerator Control Systems

Karen S. White

Jefferson Lab

In the Beginning

- First accelerator control systems were hardwired arrays of dedicated hardware
- No computers ☹
- User interfaces
  - Knobs
  - Dials
  - Switches
  - Paper strip chart recorders



Karl Brown at the klystron controls for the Mark II accelerator at Stanford University. ~1950
Photo from Symmetry August 2005

Then There Were Computers

- By the early 1960's, technology provided affordable digital stored program computers
- Motivation for computer based control systems
  - Computing resources within reach
  - Success in industrial process control
  - Larger machines planned
  - Flexibility, expandability, efficiency
  - Desire for more sophisticated control

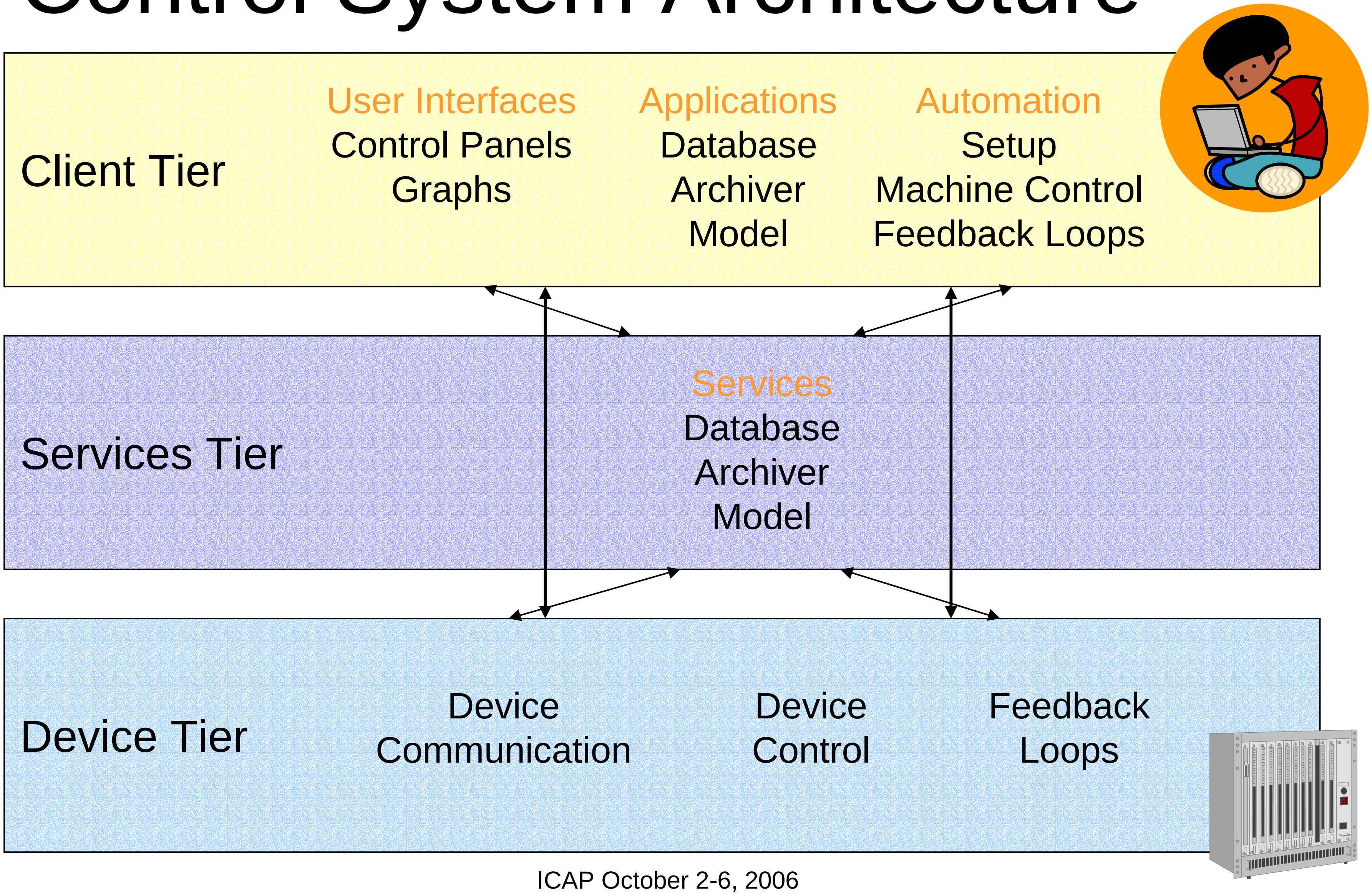
Early Computer Controls

- 1963 plans for Los Alamos Meson Physics Facility included computer controls
- Technology
  - 16-bit machine
  - 8K words of memory
  - Assembler language
  - Custom OS and fieldbus
  - Color CRT with knobs
  - Database for device information

Evolution

- By late 1960's
  - Existing machines introducing computers into their control systems
  - New machines planned with computer based control systems
  - Highly custom systems
- Today
  - Accelerator operations depend on sophisticated computer based controls with lots of commercial components

Control System Architecture



Control System Communication

- Volume of data has grown
- Data is more complex – includes attributes
- Communication models have evolved
  - From inefficient, not synchronized models
  - To efficient, synchronized models
- Further efficiencies
  - Send on change
  - Deadbands
  - Gateways

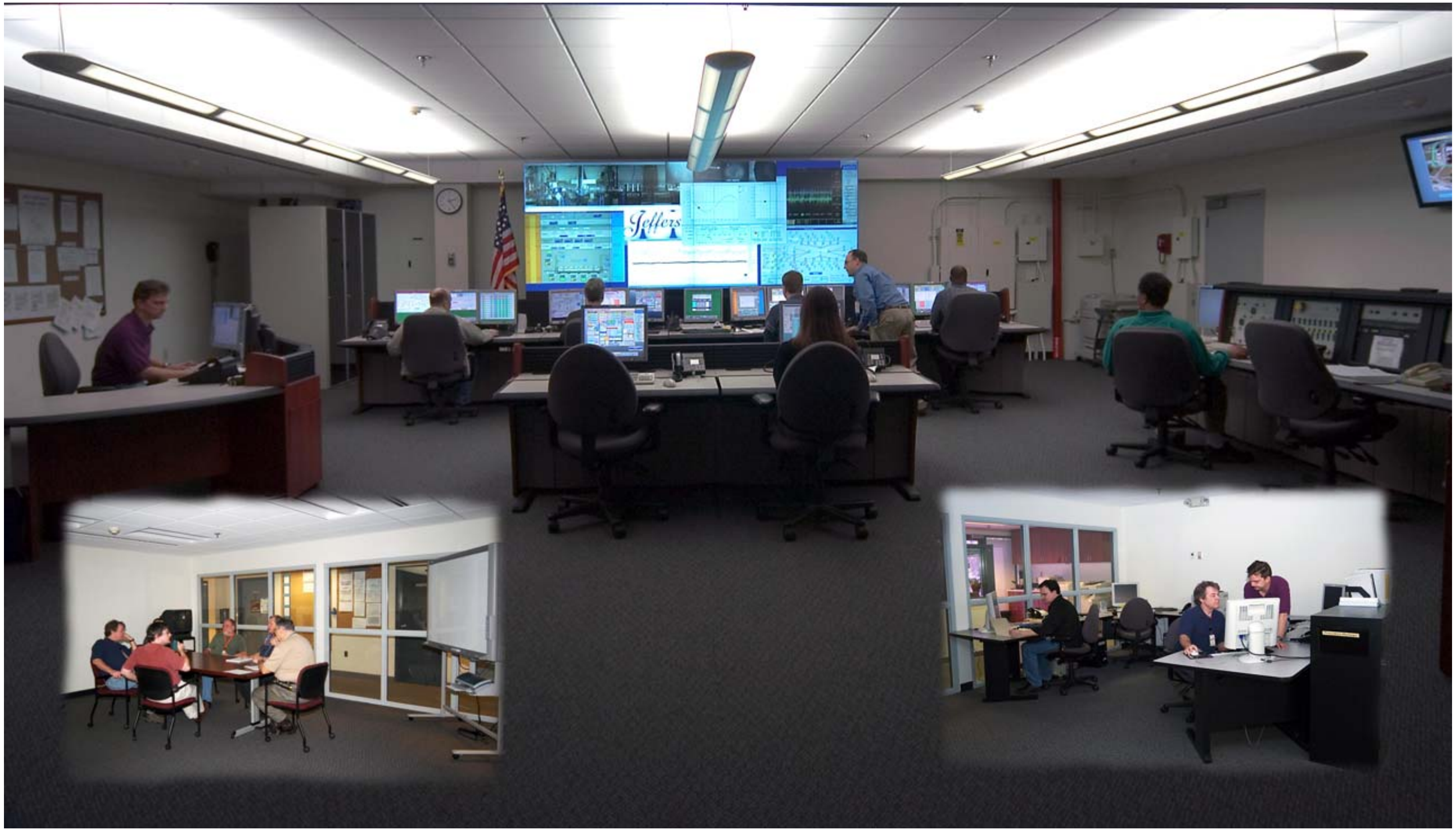
Trends

- Moving away from custom solutions
- Software Sharing/Reuse – most for device control and communications
  - Toolkits – EPICS (APS, JLab, SNS, others)
  - Frameworks – TANGO (ESRF, Soleil, others)
  - Commercial SCADA – (DESY, LHC, others)
- Limited success for high level applications

Trends

- Highly Distributed – moving towards embedded processors and network attached devices
- Automation Increasing
- HMI
  - Animated graphics
  - Large number of monitors, wall displays
  - Data overload

Modern Control Room



Jefferson Lab CEBAF Control Room 2005

Data Management

- Lots of room for improvement
- Some systems static data is distributed amongst the front end processors
- Some systems provide central DB for limited set of static data
- Effectiveness highly dependent on application use

What Next?

- Future machines will be bigger
  - 10x more devices
  - 100x more front end processors (one per device)
  - Larger geographic area
  - Large data stores
  - Greater need for automation

Technology

- New machines require more network bandwidth, storage capacity
- Technology is still evolving very rapidly
- Should be decided later in project
- Scientists and engineers have proven ability to incorporate new technologies to scale for larger systems

Other Challenges

- Global Project
- Reliability
- Maintainability
- Operability

Global Project

- Cost of future large machines implies a greater degree of global funding, development, maintenance and operations
- Cost considerations may drive outsourcing of large subsystems
- Implies networked operation which requires high level of security
- In order to succeed
  - Well defined, enforced nomenclature, standards and interfaces, including development processes, from the beginning
- Need better communication

Reliability

- Effect of 10x more device on MTBF
- Need to increase reliability
  - High availability components
  - Redundancy for key components
  - Reduce time to repair - dominated by time to diagnose
  - Design systems with uniform, built-in diagnostics reported to control system
- Control system needs to know every device is ready and operable
- Good SW engineering practices including extensive testing

Software Maintainability

- Biggest maintenance challenge is configuration control
  - Software must “discover” the machine configuration from central data repository and automatically adapt to changes
  - Depends on cooperation of all groups and use by all applications
- Difficult to identify dependencies to evaluate impact of a change
- Much better to get it correct in the beginning by rigorous use of requirements, reviews and testing
- Cost of fixing bugs is high compared to using good software engineering practices

Software Engineering Process is Important: Reliability and Maintainability

Compiled from multiple studies by Barry Boehm, USC and Victor Basili, U. of Maryland

- Finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase.
- About 40-50% of the effort on current software projects is spent on avoidable rework.
- About 80% of the avoidable rework comes from 20% of the defects.
- About 80% of the defects come from 20% of the modules and about half the modules are defect free.
- About 90% of the downtime comes from at most 10% of the defects.
- Peer reviews catch 60% of the defects.
- Perspective-based reviews catch 35% more defects than non-directed reviews.
- Disciplined personal practices can reduce defect introduction rates by up to 75%.
- All other things being equal, it costs 50% more per source instruction to develop high-dependability software products than to develop low-dependability software products. However, the investment is more than worth it if significant operations and maintenance costs are involved.
- About 40-50% of user programs enter use with nontrivial defects.

Source http://www.cebase.org/www/AboutCebase/News/top-10-defects.html

CeBase = NSF Center for Empirically Based Software Engineering

Operability

- Control systems moving towards a higher degree of automation – requires better modeling
- Complete automation of setup and operations needed for very large machines
- Controls interfaces more for status and information than for control
- Need consistently enforced standards for data presentation and GUI behavior
- Need to present information (rather than data) in useful context

Operability - Alarms

- Significant work remains to create useful alarm systems that only show conditions that operators need to know about, when they matter
- More a matter of thoughtful definitions than improved tools

Operability - Archiving

- Increasing requirements for huge amounts of data to be stored indefinitely
  - Follow's the Technician's Corollary to Parkinson's Law
    - No matter how big the data storage medium, it will soon be filled.
- Need better means to access and analyze logged data
- Need to think about what is really needed, how often and for how long

Summary

- Meeting the challenges for future systems enabled by initial adoption of
  - Project-wide standards
  - Rigorous engineering development processes
  - Consistent use of a central repository for machine configuration data
- Biggest challenge - getting a large number of people to do things the same way!