Three-Tiered Software Architecture at LANSCE

Author:
Eric E. Westbrook

Co-Authors:
Dr. Scott Baily, Greg DeLaTorre, Dr. Abraham Flores, Christopher Mendoza, Jonathan Quemuel, Dr. Tyagi Ramakrishnan, David Zimmermann

Los Alamos Neutron Science Center
Los Alamos National Laboratory, NM USA

LA-UR-26-23067

What is Three-Tiered Software Architecture?

- Presentation Tier
  - User interfaces
  - Communication between users and application tier
  - No direct data tier access
- Application Tier
  - Calculations, logic, access to data
  - Centralized functional implementations
  - Consistent APIs for all platforms
- Data Tier
  - Storage, organization, maintenance, and retrieval of all data
  - Single source of truth

Monolithic vs. Layered vs. Tiered Architectures

- Monolithic
  - Each program contains all functionality
  - No separation of concerns
- Layered
  - Programs refactored and well-organized
  - Conceptual separation of concerns
  - Still all functionality in each instance
- Tiered
  - Each tier runs on distinct infrastructure
  - Full separation of concerns
  - Ability to develop and deploy independently and rapidly

LANSCE Legacy Software Architecture

- Client / Server
  - Some separation of concerns
  - Not used consistently across software inventory
- RPC
  - Varying technologies in use:
    - SUN RPC
    - Java RPC
    - Custom protocols
    - File-based RPC (yikes)
    - REST (in limited but increasing use)
- (cont'd)

LANSCE Legacy Software Architecture (cont'd)

- Monolithic UIs and services
  - Complicated applications
  - Conflate presentation, calculation, and data storage functions
- Flat file data storage
  - Over the years, applications created their own one-off persistence
  - File locations, data formats, and naming conventions vary widely
- Multiple sources of truth
  - Over the years, applications have duplicated functionality
  - Additional sources of data were often created because of inexact fit

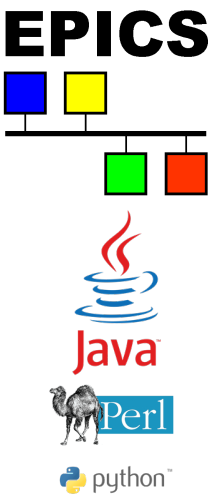
LANSCE Legacy Software Platforms

Applications

EDM

Tcl/Tk

Services



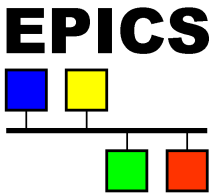
- UI Applications / Screens
  - EDM (~950)
  - Tcl/Tk (~164)
  - Java (~37)
  - Python (~20)
  - Web-based (~5)
- Services
  - EPICS
    - ~650 Hardware IOCs
    - ~25 Software IOCs
    - 6 EPICS Gateway Services (C/C++)
  - Perl etc. (~100)
  - Java (~24)
  - Python (~5)

LANSCE Future Software Platforms

Applications

Phoebus
(CS Studio)

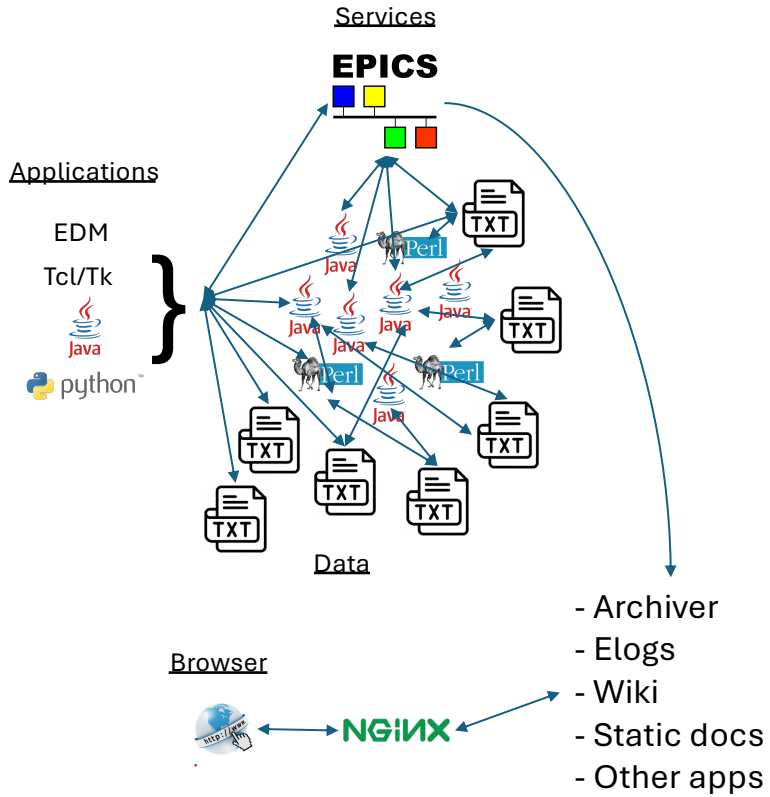
Services



- Migrate Urgently
  - Tcl/Tk
  - EDM
- Migrate Opportunistically
  - Java
  - Perl etc.

- Target Platforms
  - Phoebus (CS Studio)
  - Python
  - Web UIs / APIs

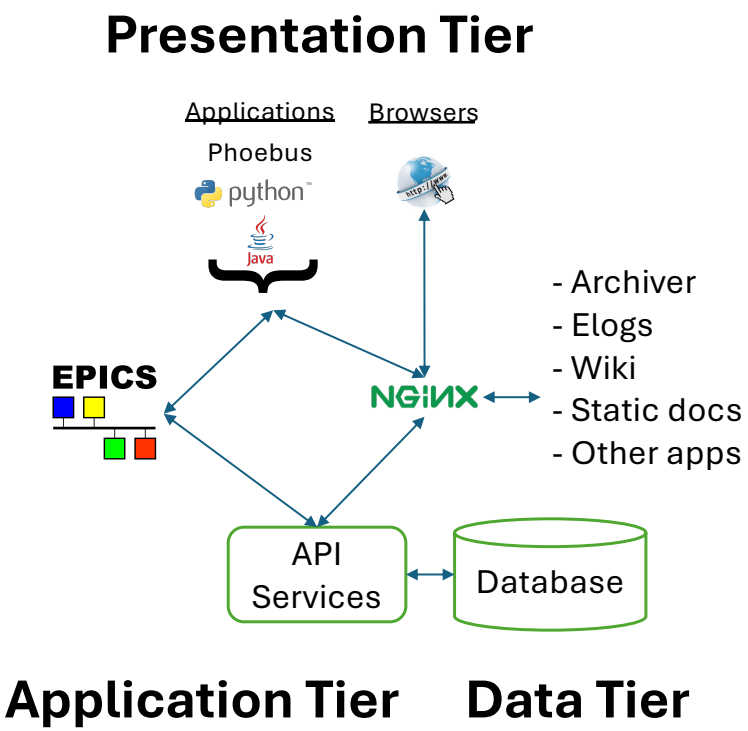
LANSCE Legacy Software Architecture



Technical Debt to Pay Down:

- Data redundancy
- Disparate RPC protocols
- Multiple sources of truth
- Over-engineered solutions
- Fragile/brittle code
- Inconsistent implementations
- Scattered documentation

LANSCE Future Software Architecture



- Presentation Tier
  - Web, native, or CLI user interface applications
  - REST and CA interfaces to application tier
- Application Tier
  - Access to calculations and data for presentation tier via REST APIs
- Data Tier
  - Applications and logic are relieved of storage particulars
  - Single source of truth
- NB: EPICS serves multiple tiers, yet still has sufficiently good separation of concerns

Thank you!