Implementing SOA at Duke University Health System
The Journey and How It Enabled Big Wins

Boyd Carlson
Sr Director Application Development, Integration
Duke University Health System
Duke University Health System

- Large academic medical center
- 2 community hospitals
- 80 outpatient clinics, 60 locations
- Supports patient care, research, and education
- 800,000 patients seen each year
- 60,000 inpatient visits/year
- 1,600,000 outpatient visits/year
Duke Strategy and Architecture

• Best of breed shop, about 40 interfaced systems, across 3 hospitals

• Some key applications are home grown
  – Clinician result display, clinical data capture, disease management, transcription, Patient Portal, etc.

• Partnerships with vendors for development

• Integration architecture it key to application strategy
  – Clinical Data Repository (CDR)
  – Interface Engine, interfacing standards
  – Software layering
  – SOA
Duke Patient Portal
Start to Production in 3 Months

• In 3 months we:
  – Partnered with IBM, purchased WebSphere
  – Delivered a Patient Portal, in production with:
    • On-line registration for patients
    • View bill details, pay your bill, request payment plan
    • Print billing summary for tax itemization
    • Update demographics and insurance
    • Request an appointment

• This is typically a 12-24 month project because of integration work
Duke Patient Portal
The Next 4 Months

• New functions brought into production
  – Proxy access for children’s information
  – View future and past scheduled appointments
  – View lab, radiology and allergy information

• Required integration with our CDR and additional IDX functions

• It took 2 days to display the patient’s entire medical record in the Patient Portal using existing CDR web services
Duke Patient Portal
The Architecture

• All transactions and queries go through centralized web services

• Built the actual Patient Portal with WebSphere

• Existing web services
  – Duke Clinical Data Repository
  – Hyland Onbase workflow management system

• New web services were built on top of
  – IDX Healthcare Objects
  – Siemens Novius Data Warehouse
Duke Patient Portal
What’s Up Next

• Coming projects that use the current SOA and new services we’ll add to it
  – Patients make their own appointments on-line
  – Advance Registration – Patients fill out forms, clear demographic and insurance questions, and pay their co-pay on-line (eligibility in the background)
  – Kiosks for on-site checkin, way finding, etc.
  – Referring Physician Portal
  – Patient Education

• Deep integration enabled by IDX Health Care Objects is critical to Advance Registration, Appointments on-line, and Kiosks
RHIO Demonstration Project with IBM

- Used 1 man month of effort
  - Implemented register report, query for full report paradigm with the Community Hub (CH), for all of Duke’s EMR data
  - Historical report registration with the CH at patient registration
  - All new reports registered with CH real-time
  - Created new CDA report formatter (no CDA previously)

- All of the above was done within existing architecture and services, while implementing a few new features

- Other sites simply hooked up their Interface Engine without CDA, or didn’t participate at all because of the level of effort
Using Old Mainframe CICS Functions from Web Applications

• Used existing mainframe functions via a service

• Started with 25 year old CICS transactions, layered Remote Procedure Calls on them, and layered ODBC on top of that

• Used for physicians to drive HIS functions
  – Pend a patient for discharge
  – Change physician responsible for a patient

• It took about a week’s effort to build this facility
  – (we had a good tool)
Duke Architecture
1992

DHIS
Homegrown HIS
Mainframe - IMS, CICS, PL/I

Proprietary Interfaces

Labs - Cerner
Scheduling, Prof Billing - IDX
Goals – The Architecture Vision

• Build technology that:
  – Supports a demanding academic medical center
  – Is agile
  – Can be adapted to new standards and technologies
  – Supports a best of breed environment
  – Facilitates replacement of major systems with minimal effort
  – Allows us to build our own applications
  – Fosters reusing transactions and data

• It’s a distributed architecture with independence between elements
  – Do not wind it up too tightly
The Approach

• Develop an architectural vision that fit our organization

• Create organization structure and processes to ensure the vision is followed

• Use what we have
  – Use the systems and technology you have in place
  – Avoid major rewrites, layer on top of old technology

• Do it incrementally
  – Evaluate each new project for opportunities to build new architecture and services
The Approach cont’d

• Do not try to design/build for too far in the future
  – No one will fund a big application infrastructure project

• Use standards if available
  – Build it so we could use standards as they evolved

• Generalize functions/technologies we build, within reason

• Get all transactions flowing through common infrastructure so future projects can readily use them
  – If you need something once, you’ll probably need it again
Duke Architecture
1992

DHIS
Homegrown HIS
Mainframe - IMS, CICS, PL/I

Proprietary Interfaces

Labs
- Cerner

Scheduling,
Prof Billing
- IDX
Duke in 1992

• The situation
  – Duke’s previous monolithic strategy was in the way of quickly developing new functionality
  – Bring in 25 best of breed systems quickly
  – Physicians need better access to data

• The strategy
  – Adopt HL7
  – Build a repository
Duke Architecture - 1994

Integration Broker

Billing
Scheduling Registration
Labs
Pharmacy
DHIS

Result Display (Mainframe)

Retrieve Results

Communication Layer
Data Representation Layer
Business Object Layer
Data Access Layer

HL7
HL7
HL7
HL7

Post Results

CDR

HL7 Asynchronous Transactions (about 15 systems)
Duke in 1995

• The situation
  – We installed many systems, interfaced with HL7
  – We built a repository
  – Many physicians refused to use the mainframe 3270 result display system we built for the repository

• The strategy
  – Install PC’s on all the inpatient units
  – Create a GUI result display client – the Browser
  – Create layered repository access middleware to support the new Browser
Duke Architecture - 1996

Integration Broker

Billing
Scheduling Registration
Labs
Pharmacy
DHIS

Result Display
(C++ Client)

Transcription System
(Lotus Notes)

HL7
COM
Communication Layer

HL7
Data Representation Layer

Business Object Layer

Data Access Layer

HL7 Asynchronous Transactions
(about 25 systems)

CDR

Other Systems and Databases

Synchronous Transactions
Duke in 2005

• The situation
  – The Browser was wildly successful, converted to web
  – The repository had turned into the medical record
  – We needed to decompress our call center, be more efficient managing patients, involve patients in their care more

• The strategy
  – Build the Patient Portal with WebSphere
  – Transactionally connect the Portal to operational systems
  – Use web services between the Portal and everything else
  – Use IDX Health Care Objects
  – Use billing data in the Siemens Novius Data Warehouse
  – Send Portal requests to call center with Hyland Onbase
Key Architecture Points

• Get data and transactions out on a common street so everyone can use them – SOA, Interface Engine

• Build software layers – they enable change, hide complexity
  – The business objects in the middle rarely change, but communication, data representation, and data access do change

• Look for simplifying technologies
  – ODBC isn’t sexy, but everyone can do it – It’s a service

• Be consistent
  – There must be a VERY good reason to step outside of your architecture

• Centralize and hide your complexity, don’t make everyone deal with it
What Did We Have to Correct?

• Problem: We were too strict with our standards, and bent vendors to our will
  – We made vendors commit unnatural acts which they took too long to do, and could not support

• Correction: We adapted to what vendors did naturally
  – We bought a new interface engine that made it easier for us adapt to vendors
  – We built new middleware with business logic in it to deal with vendor shortcomings (the Order Manager)
What Did We Have to Correct?

• Problem: Some of our technology and standards were too sophisticated and heavy
  – Some departments went around us
  – Some projects took too long

• Correction: We built some lighter technology – Think about providing a service to your customers
  – We put ODBC views over some transactions to give departments simpler technology
  – Started using web services when they became popular
  – Adoption of Web Services and XML lowers the bar
Summary

• Our architecture has enabled several leaps forward in the functionality we provide our customers

• SOA is a key part of it

• SOA does not have to be exotic

• Don’t worry about old technology, layer on top of it

• New tools like Web Services and XML are making it easier
Summary

• Magic happens if you continually build standard methods to move data and expose functions

• It's an architecture vision – revisit it regularly

• It's about balance
  – The most sophisticated architecture is worth nothing if its not used or too expensive
  – 3rd Normal Form is nice, but don’t let it get in your way

• Don’t make it too big, do it along with new projects incrementally

• Push vendors to open up their systems