Service Oriented Architecture

Brown Bag Lunch Presentation, August 23, 2007

From	Michael Neuman <neuman@georgetown.edu>
Sent	Thursday, August 23, 2007 9:19 am
To	uis-all@georgetown.edu , CNDLS <cndls@georgetown.edu>
Cc	
Bcc	
Subject	Reminder: Piet and Charlie on SOA

Just a reminder that today at noon in Harris 2005, Charlie Leonhardt and Piet Niederhausen will give a Brown Bag presentation on Service Oriented Architecture. The talk is a version of their presentation at this summer's Snowmass Seminars on Academic Computing.

Piet and Charlie write:

This session will review service oriented architecture (SOA) from a number of perspectives. We will define SOA, discuss its value to higher education, and present use cases on how it can enhance our scholarly and enterprise systems. Challenges in organizing for and implementing SOA will be highlighted including a discussion of using SOA in enterprise open source development as well as distributed environments.

Mike

--
Michael Neuman, PhD
Senior Associate for Scholarly Information Initiatives
University Information Services
Georgetown University
3300 Whitehaven Street N.W.
Suite 2000
Washington, DC 20007
202-687-6283
202-687-1505 (fax)

Georgetown UNIVERSITY

Service Oriented Architecture

A Brown Bag Lunch Presentation

University Information Services, Georgetown University

August 23, 2007

Publisher's Note

- This presentation is based on a presentation given Tuesday, August 7, 2007 at the EDUCAUSE Seminar in Academic Computing, Snowmass, Colorado.
- The third presentation at the Snowmass Conference made by Jens Haeusser, University of British Columbia, is given here by Mr. Leonhardt.

Georgetown UNIVERSITY

Part I: Service Oriented Architecture

Seminars on Academic Computing

Directors Leadership Seminar, August 7, 2007

Charles F. Leonhardt, Principal Technologist, Georgetown University

leonhardt@georgetown.edu

Overview

1) What is SOA?
2) Why is it important and why should we care about it?
3) How can it be used to enhance the services we provide in Scholarly Systems?

Some Truth in Advertising

- Not an expert in SOA and don't write code used within an SOA….
- Work in a computing environment without a fully developed SOA
- Do believe that SOA can bring tremendous benefits to all parts of our IT infrastructure:
  - The Infrastructure Itself
  - Business Process Improvement
  - Better Scholarly, Enterprise, and Distributed Services for our customers

A Quick Show of Hands

Before we even define SOA…..

How many of you think a Service Oriented Architecture is in place and being used successfully in some way on your campus?

Georgetown UNIVERSITY

1) What is SOA?

Gartner says:

> "A style of multi-tier computing that helps organizations share logic and data among multiple applications and usage modes."

….. But with what level of probability? :)

What is SOA?

Wikipedia (the font of all human knowledge) says:

> “SOA is an evolution of distributed computing and modular programming. SOA provides a modularity of logic that can be presented as a service for a client (as in client-server architecture) and at the same time function as a client for other services.”

What is SOA?

IBM says:

> “An application architecture within which all functions are defined as independent services with well-defined invokable interfaces which can be called in defined sequences to form business processes.”

What is SOA?

SOA for Dummies says:

> “A software architecture for building applications that implement business processes or services using a set of loosely coupled black-box components orchestrated to deliver a well-defined level of service.”

Service Oriented **Architecture**

I think…… It's Emphasis on all of the words:

ar·chi·tec·ture

- a fundamental underlying design of computer hardware, software, or both
- the structure of anything, e.g. the architecture of a novel.

- NOT software or a specific tool

Service **Oriented** Architecture

or·i·ent·ed

- To align or position with respect to a point or system of reference
- adjusted or located in relation to surroundings or circumstances

Georgetown UNIVERSITY

Service Oriented Architecture

serv·ice

- We all know what services are: we seek them and consume them and provide them often every day!

Some Service Examples: Staying at the Silver Tree Hotel

- Reserve a Room
- Check Credit / Charge Deposit
- Send Reservation Confirmation
- Bellman: Pickup and Deliver Baggage
- Park Car
- Check In / Assign Room
- Issue Room Key
- Charge Hotel Account for Service x
- Issue Bill
- Charge Amount Due

How does we isolate unique services?

- By going through an existing business process…. or performing business process reengineering to get at a better, improved process (with the cheerful and patient collaboration of functional and technical staff),….. we can isolate well defined processes that can be reused.
- Process oriented design is a critical component of SOA

Attributes of SOA

- An Applications **Architecture**
- Multi-tier / distributed computing environments
- All functions are well-defined as independent **services** with invokable interfaces
- Modular / distributed logic in loosely coupled black box components which may be reused… and, therefore, used by multiple applications
- Black box components hide specific technologies or technology barriers……
- Process oriented design

2) Why is SOA important and why should we care?

- Most of us have been held hostage to heterogeneous computing environments that were or are:
  - Built without an applications or software architecture at all or, worse yet, competing architectures
  - Proprietary
  - Dependent on specific technology tools and / or vendor/system specific interface specifications
  - Using code bases where functions/services are not isolated and code is not reusable
  - Very dependent on hundreds of point-to-point interfaces

Consequences of being “held hostage”

- It takes “too long” and it “costs too much” to integrate new systems because code is not reusable and modularized
- User frustration at this lack of efficiency, particularly beyond central IT, causes people “who want to get something done” to
  - create shadow systems replicating data and/or business process
  - use standalone interfaces that requires care and feeding and likely doesn’t contain the “right” data
- Innovation is stifled
- Service levels and user satisfaction are lower than they *should* be`

Scholarly Systems

Scholarly Systems are those systems that directly support our core business:

- Course Management Systems
- Synchronous Learning Environments
- Rich Media Services
- Assessment Engines / Course & Faculty Evaluation
- Digital Repositories and Digital Libraries
- Digital Notebooks
- ePortfolios
- Academic Portals
- Wikis / Blogs / Tagging in pedagogy

Georgetown UNIVERSITY

3) SOA and Scholarly Systems

A well implemented SOA will deliver modular, reusable services directly applicable to Scholarly Systems:

- Get student name…. School / Class / Major / Status
- Get Enrollment for a Course
- Validate Enrollment in a Course
- Get seats available
- Dozens more……..

Georgetown UNIVERSITY

Benefits of SOA in Scholarly Systems

- Integration of new services will be much more efficient as new applications or services use modular service agents to get, process, or store needed information
- Decentralized, departmentally based applications AND external partners could use business process and data (with the appropriate data security controls, of course) obviating the need for shadow systems or redundant interfaces
- Innovation is encouraged
- Service levels and user satisfaction are where they should be
- Delivery of very high quality user experiences at lower long term costs

Georgetown UNIVERSITY

Introduction to SOA
Part II: SOA in the enterprise

Seminars in Academic Computing, Directors Leadership Seminar, August 7, 2007

Piet Niederhausen, Web & Data Architect, Georgetown University

Overview

1) How will we bring SOA into our institution?
2) How will we manage and govern SOA?
3) What SOA infrastructure capabilities will we need?

1) How will we bring SOA into our institution?

a) Services use cases
b) SOA technologies vs. SOA strategies
c) “Big SOA” vs. “little SOA”

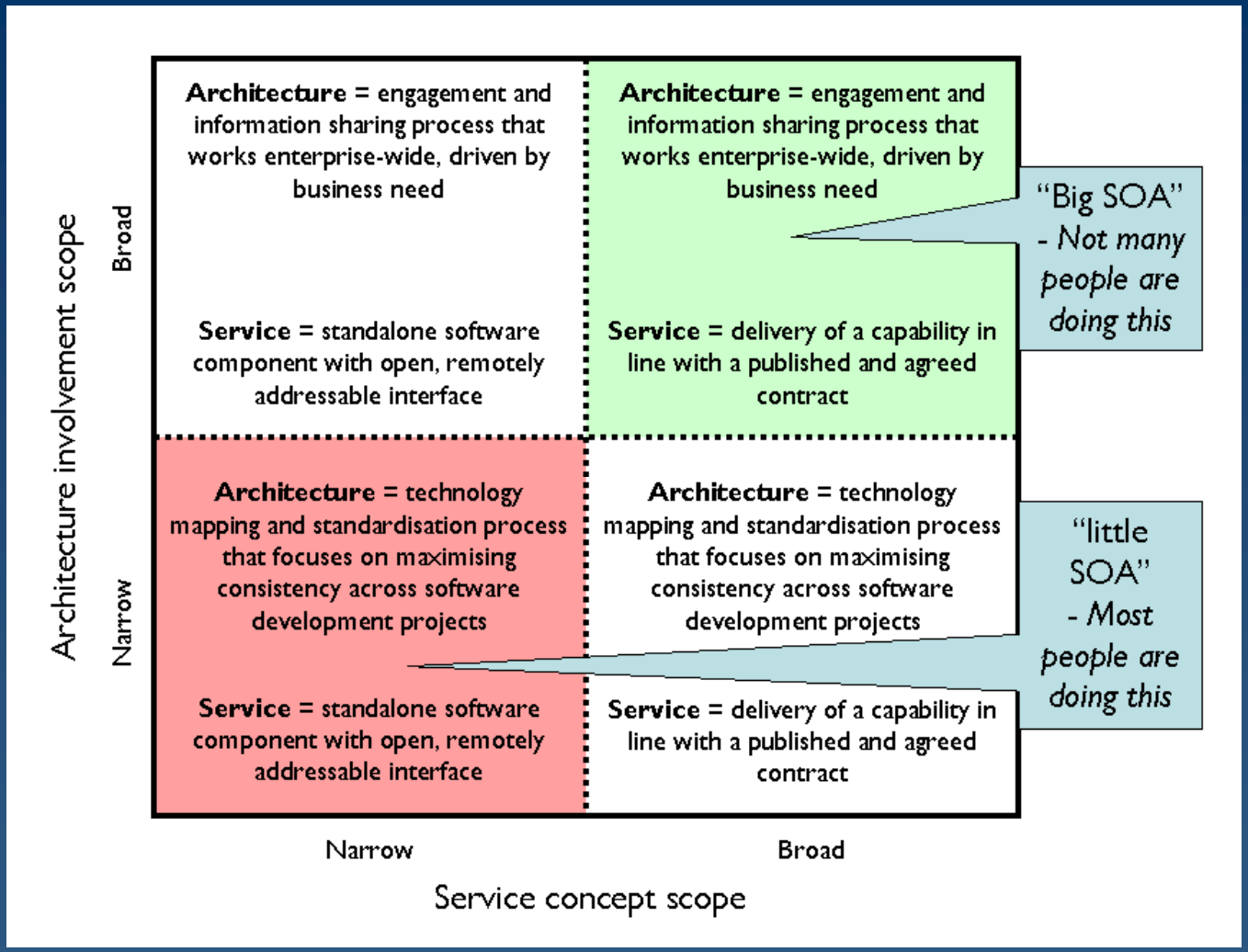
Services use cases

- Application to application messaging or data exchange through services
  - e.g., between an in-house application and a hosted application
- Portlets based on services
  - e.g., institutional portal with portlets based on services from student system, courseware, etc.
- Web 2.0 user interfaces and mashups using services
- Applications consuming shared enterprise services
  - e.g., shared application services for authorization, identity management, CRM, etc.
- Services used within the design of a large application
  - e.g., between modules of an enterprise business system

SOA technologies vs. SOA strategies

- You probably already have some use of Web Services or other SOA-related technologies in your enterprise
- Without an SOA strategy, these are just another tool for application integration, with all the usual potential risks
  - Tightly coupled, poorly documented dependencies between applications
  - Poorly understood security implications
  - Probably not re-usable
- What will be your approach to making implementations of SOA-related technologies part of an enterprise architecture?

“Big SOA” vs. “little SOA”



Used with permission of the author:
Neil Ward-Dutton, Research Director, Macehiter Ward-Dutton

“Little SOA” or “bottom up”

- Service = A software component with an open interface that can be used remotely
- Architecture = Technology standards that promote consistency within and across application projects
- Outcomes
  - Complete specific development or integration projects
  - Develop useful skills in Web Services and related technologies
  - Provide great showcase projects for wider adoption of SOA
- Risks
  - Create a network of poorly understood and poorly managed service interdependencies
  - Miss opportunities to make services widely re-usable; gains in agility are limited to specific applications or domains

“Big SOA” or “top down”

- Service = Delivers a result according to a published, enforceable contract; designed for re-use
- Architecture = Enterprise wide communication and governance; tied to business strategy
- Outcomes
  - Documented understanding of enterprise business domains and processes
  - Create services that are re-usable across business processes
  - Create institutional governance for services and data
  - Create enterprise infrastructure for managing services
- Risks
  - Take on a very large planning and design process that may overreach and fall short on the IT side, business side, or both
  - Implement a complex infrastructure with insufficient organizational expertise or resources to manage it

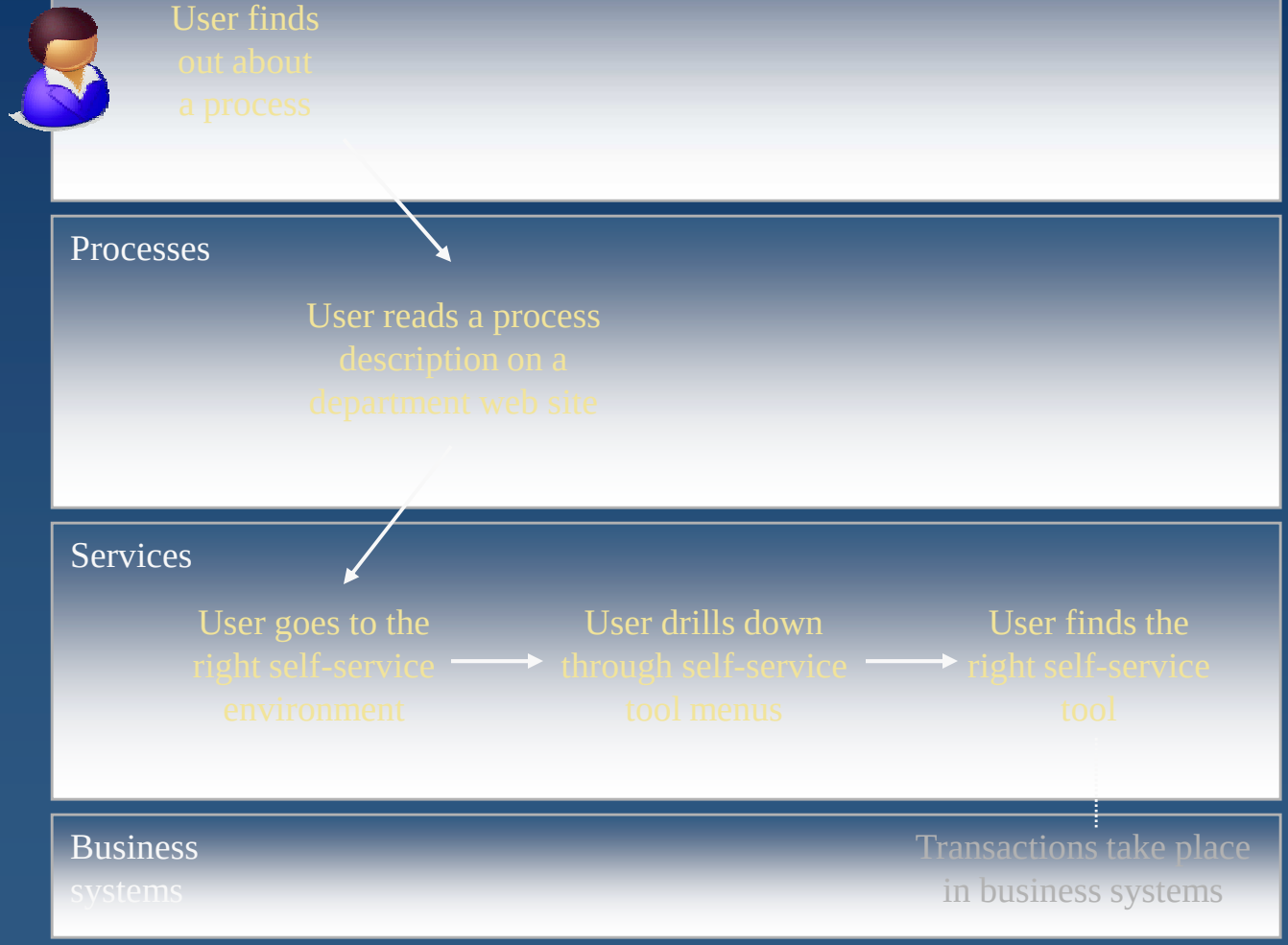
Hybrid or “middle-out”

- Recognize SOA as a long-term element of your enterprise architecture
  - Related to data administration; application architectures; integration standards
- Set up governance early and grow its sphere of influence over time
  - Facilitate communication and collaboration to make short-term projects more likely to support long-term needs
- Identify achievable projects that provide opportunities
  - Create re-usable services (along with some that aren’t)
  - Build SOA-related skills
- Identify long-term infrastructure needs and incrementally build your way there

Example: Enterprise self-service

- Self-service: The ability for end users to conduct their business with the support of enterprise systems
  - Check their status, access data or content, conduct transactions
- Currently this is done with a mixture of vendor-provided and home-grown web front ends for enterprise systems
- Future self-service will be unified, crossing domains for any given task or inquiry
  - Fully realized user life cycle
  - Processes that cross domains and systems
- But for the foreseeable future, self-service incorporates a variety of resources:
  - Vendor-provided self-service environments
  - Vendor-provided APIs, some with Web Services
  - Home-grown self-service applications, and their APIs
  - Web sites and web content management systems
- SOA can bring structure and discipline to this diverse environment
  - Focus on processes
  - Think of applications as sets of services, even if they aren't Web Services yet

Current web self-service



No unified communication across web sites.

Each department's ability to communicate is fragmented.

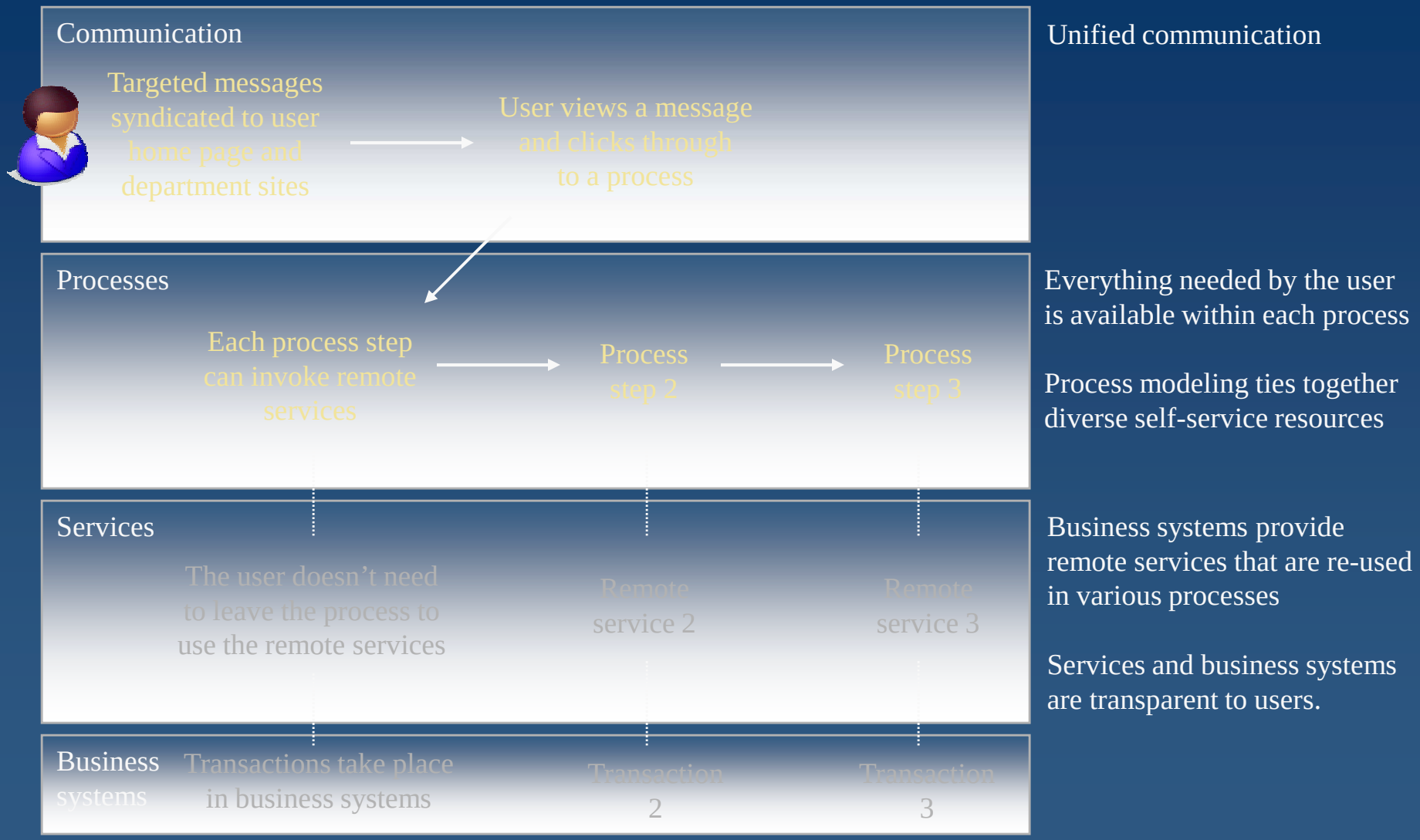
No automated process steps.

Each department's process documentation is unique.

Self-service environments are "tool boxes" without process context.

Each domain's self-service environment is unique.

Future self-service with SOA



Unified communication

Everything needed by the user is available within each process

Process modeling ties together diverse self-service resources

Business systems provide remote services that are re-used in various processes

Services and business systems are transparent to users.

2) How will we manage and govern SOA?

a) Service life cycle
b) SOA governance

"It's really easy to build a Web Service, it's really easy to consume a Web Service, but it's really hard to manage a Web Service.“

– Sri Muthu, VP, Wells Fargo Inc.

Service life cycle

- Request a new service
  - Compare with existing services; decide whether to adapt an existing service, compose existing services into a new business service, or create a new implementation service
- Create a new service
  - Development
  - Testing, including impact on any existing services being re-used
- Implement the new service
  - Change control (new use of an existing implementation service is also a change)
  - Add to services registry; add relevant materials to repository
- Operate the new service
  - Monitor
  - Maintain required levels of service
  - Support re-use of the service
- Retire the service

SOA governance

SOA governance is “the creation, communication, enforcement, maintenance and adaptation of policies across the SOA lifecycle of design time, runtime and change time.”

– Miko Matsumura, VP, webMethods

- Governance secures the institution’s long-term investment in SOA
- Governance is institution-specific, and needs different approaches even within parts of one institution
  - More centralized administrative services vs. less centralized academic services
  - Mandates vs. carrots
- As SOA grows in our institutions the people governing SOA are often also the people championing and supporting it
- The capacity to govern SOA is as much of a constraint as any of the technical, resource, or business challenges of SOA

SOA governance tasks

- Provide an interoperability framework
  - Identify supported standards and protocols
- Guide the creation of services in the context of the institution's SOA
  - Architectural review of proposed services
  - Incentives for reuse of available services
  - Disincentives for development of redundant services
- Ensure creation of service contracts and Service Level Agreements (SLAs)
- Enforce contracts and SLAs
  - Keep services reusable and reliable
- Govern the use of institutional SOA infrastructure
  - Requirements for use of shared services registry, enterprise service bus
- Coordinate with related governance efforts
  - Security
  - Change control and operations
  - Data administration and data governance
- Facilitate communication and collaboration

3) What SOA infrastructure capabilities will we need?

Depending on your need for various capabilities, these may be found in a single “Enterprise Service Bus” (ESB). An ESB provides a layer of abstraction and integration middleware between applications providing and consuming services.

- Basic capabilities
- Additional capabilities
- Management capabilities

Basic capabilities

- Service mapping; resolution of service requests
  - Services registry
- Routing of messages between services
- Protocol transformation
- Transformation and enhancement of message content
- Adapters
  - Designed to connect to specific legacy systems

Additional capabilities

- Service orchestration
  - Turn a request for a business service into multiple requests for implementation services
- Process choreography
  - Execute business logic (usually BPEL) to turn a request for a business service into a process involving multiple business services
- Quality of service
  - Security, including application-independent authorization
  - Message state, assured delivery
  - Transaction management (limited)

Management capabilities

- Monitoring, troubleshooting
- Logging, auditing
- Enforcement of policies
  - e.g., required security protocols
- Administration consoles
- Documentation repository

Summary

1) Consider how SOA will come into your institution and what form it will take
   - » Many potential use cases
   - » Different scales of SOA efforts
2) Consider how you will manage and govern SOA in your enterprise
   - » Managing the services life cycle
   - » Governing the creation and use of services
3) Consider what SOA infrastructure capabilities you may need

Open Source, Community Source, and SOA

Seminars in Academic Computing, Directors Leadership Seminar, August 7, 2007

Jens Haeusser
Director, Strategy
IT, UBC

Agenda

- Why Open Source?
- Open Source SOA Building Blocks
- Higher Education Services - Fedora
- SOA Project - Kuali Student

Open Source and Community Source

- **Open Source:**

  Software whose source code is published and made available to the public, enabling anyone to copy, modify and redistribute the source code without paying royalties or fees.

- **Community Source:**

  Hybrid development model that blends directed development by community members with traditional open-source practices.

Why Open Source for SOA?

- **Similar Aims**
  - Increased business flexibility
  - Reuse of code/services
  - Reduced costs
  - Avoiding vender and product lock-in
- **Open Standards**
- **Forrester Survey on Enterprise Open Source**
  - 77% Improving Efficiency
  - 71% Consolidating IT Infrastructure
  - 59% Migrating to SOA

SOA Building Blocks

- **Emerging set of open source projects address the complete SOA stack**
  - Portal Framework: Jetspeed, uPortal
  - Web Service Framework: Axis2, JBossWS, XFire
  - Message Broker: ActiveMQ, JBoss Messaging
  - ESB: JBoss ESB, Mule, ServiceMix
  - Service Registry: jUDDI
  - Workflow Engine: JBoss jBPM, Kuali Workflow
  - Rules Engine: JBoss Rules, OpenRules
  - Development Environment: Eclipse, Netbeans

Higher Education Services - Fedora

- **Developed by Cornell, University of Virginia**
- **Digital asset management architecture**
- **Collection of services for managing and delivering digital content**
  - Digital library
  - Multimedia authoring system
  - Institutional repository
- **Implements Open Standards**

Higher Education Project - Kuali student®

- **Next generation Student System**
- **Community Source project**
  - UBC, Berkeley, Florida, Maryland, San Joaquin
  - MIT, Carnegie Mellon
- **Student centric system**
- **Service Oriented Architecture**
  - Enables integration at diverse institutions
  - Allows schools to implement *their* practices

Kauli Vision - A Next Generation Student System

- To provide person centric services that support students and other users by anticipating their needs and reducing the time it takes to complete administrative tasks.
- To support a wide range of learners and learning activities in a wide range of institutions by using a flexible, configurable, data model.
- To support a wide range of academic and related business processes, including those that cross departments, systems and institutions, in ways that work best for each institution, by using rules based business logic, and configurable processes.
- To ensure a modular design by using a Service Oriented Architecture implemented using Web Services.
- To achieve sustainability through community source distribution and wide spread adoption.

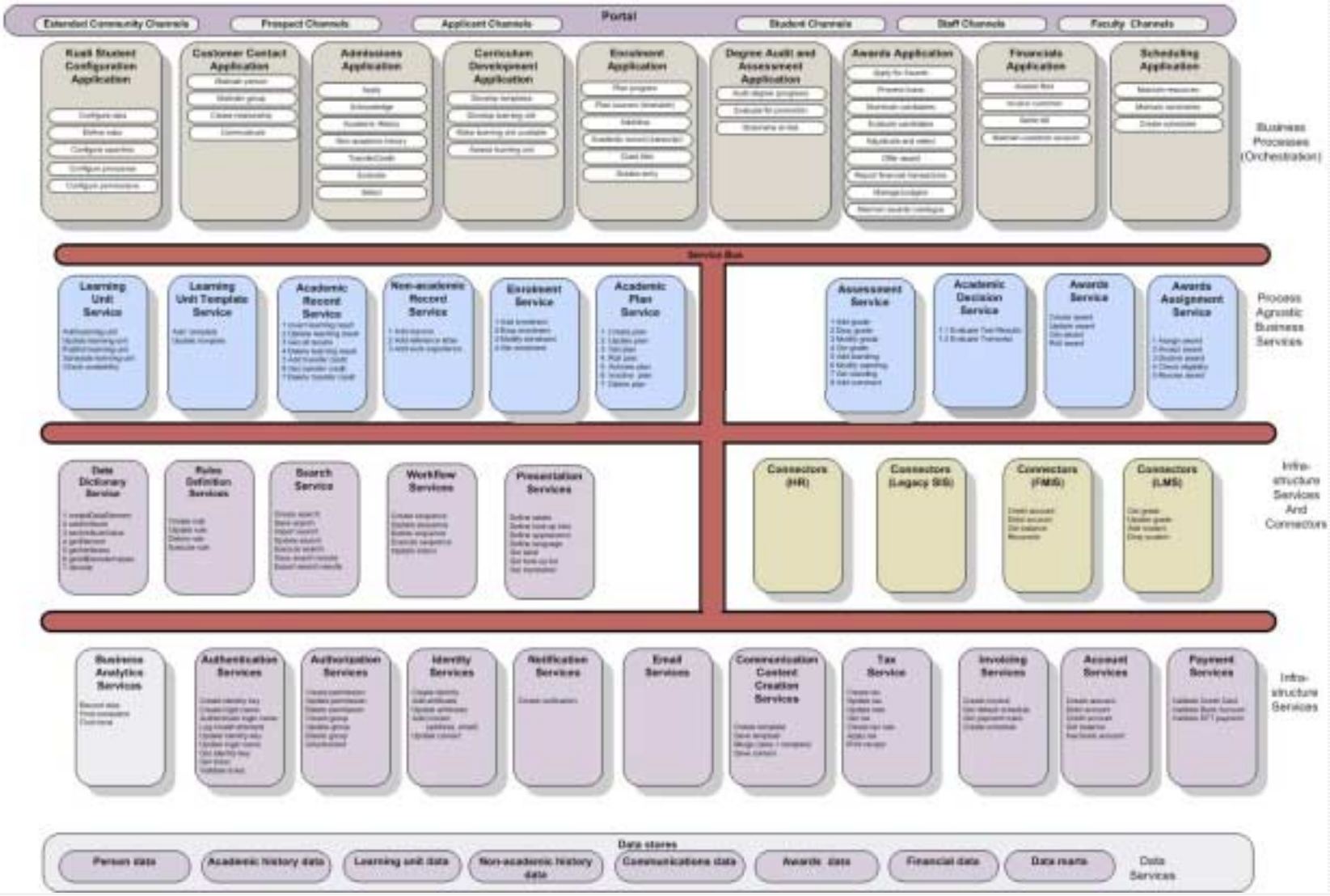
Architectural Principles

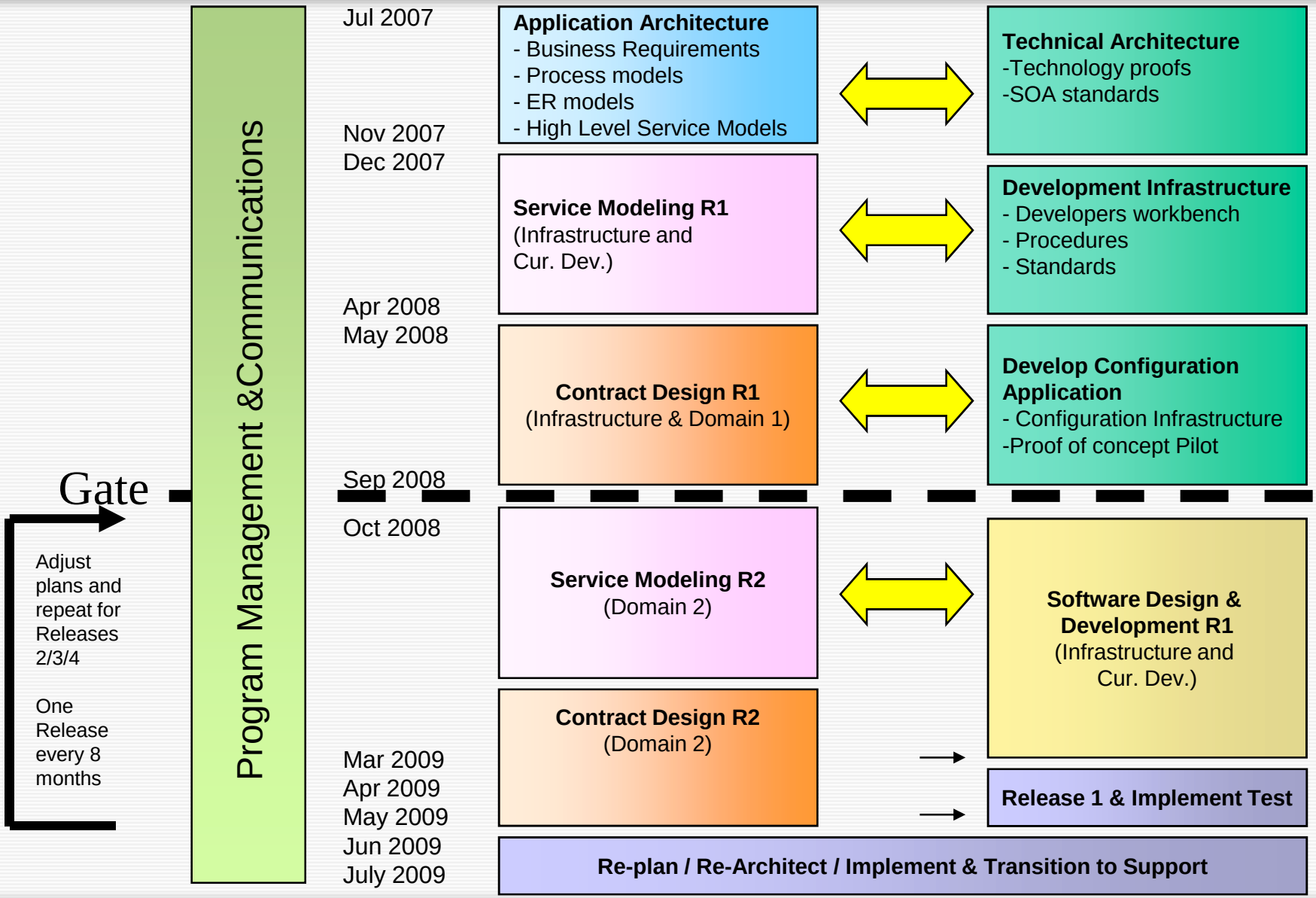
- **Service Oriented Architecture**
  - SOA Methodology
    - reuse, autonomy, loose coupling, up-front design
  - Web Services
    - SOAP, WSDL, XML
  - Standards Based
    - WS-*, IMS, PESC
  - Separate Governance for Service Contracts
    - Core assets of SOA system

Architectural Principles part 2

- **Component Abstraction**
  - Abstraction of Business Processes and Business Rules
  - Abstraction of Presentation Layer and use of a Portal
  - Abstraction of the Data Layer
- **Leveraging Open Source**
  - System will be built entirely on Open Source software stack
  - Infrastructure will use existing Open Source products
- **Development**
  - Java as the language and platform of choice

Kuali Student Service System Application Architecture




Functional
(Users & IT Functional)
Technical
(IT Architects & Developers)
Program Management &Communications
Jul 2007
Application Architecture
- Business Requirements
- Process models
- ER models
- High Level Service Models
Technical Architecture
-Technology proofs
-SOA standards
Nov 2007
Dec 2007
Service Modeling R1
(Infrastructure and
Cur. Dev.)
Development Infrastructure
- Developers workbench
- Procedures
- Standards
Apr 2008
May 2008
Contract Design R1
(Infrastructure & Domain 1)
Develop Configuration Application
- Configuration Infrastructure
-Proof of concept Pilot
Gate
Sep 2008
Oct 2008
Service Modeling R2
(Domain 2)
Software Design & Development R1
(Infrastructure and
Cur. Dev.)
Contract Design R2
(Domain 2)
Mar 2009
Apr 2009
May 2009
Jun 2009
July 2009
Release 1 & Implement Test
Re-plan / Re-Architect / Implement & Transition to Support
Adjust plans and repeat for Releases 2/3/4
One Release every 8 months

Conclusion

- Open Source software is a key building block for a Service Oriented Architecture
- Higher Education specific services are emerging, as are service contracts and standards
- Service Orientation dramatically effects how software projects are architected and governed

Questions?

leonhardt@georgetown.edu

niederhp@georgetown.edu

jens.haeusser@ubc.ca