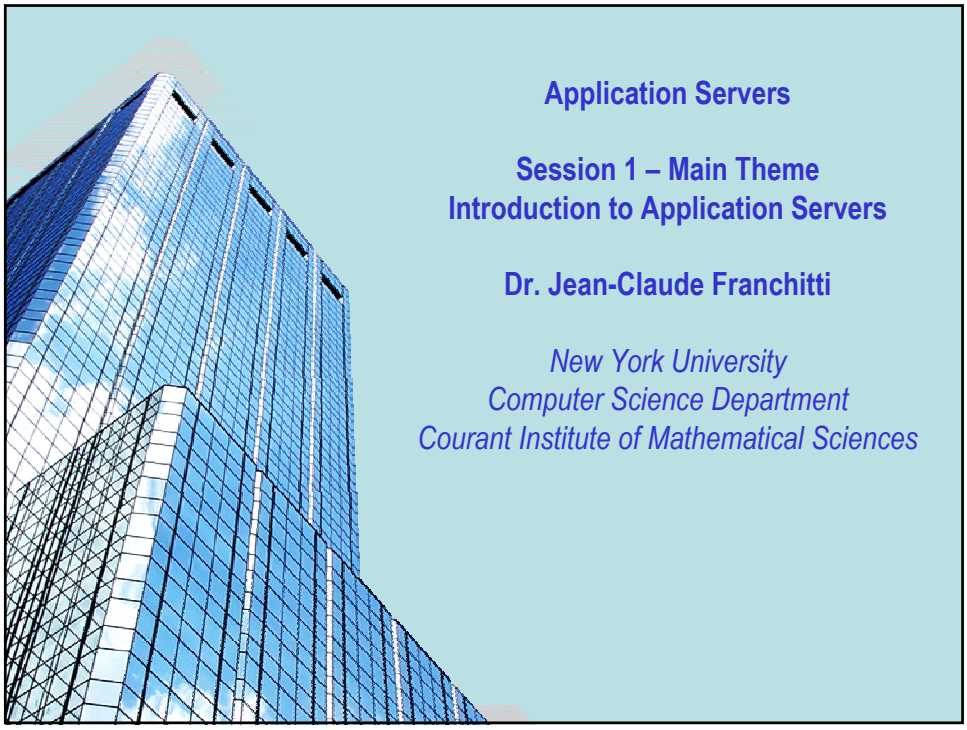


Application Servers

Session 1 – Main Theme
Introduction to Application Servers

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

Icons / Metaphors


Information


Common Realization


Knowledge/Competency Pattern


Governance


Alignment


Solution Approach

Who am I?

- Profile -

- 26 years of experience in the Information Technology Industry, including twelve years of experience working for leading IT consulting firms such as Computer Sciences Corporation
- PhD in Computer Science from University of Colorado at Boulder
- Past CEO and CTO
- Held senior management and technical leadership roles in many large IT Strategy and Modernization projects for fortune 500 corporations in the insurance, banking, investment banking, pharmaceutical, retail, and information management industries
- Contributed to several high-profile ARPA and NSF research projects
- Played an active role as a member of the OMG, ODMG, and X3H2 standards committees and as a Professor of Computer Science at Columbia initially and New York University since 1997
- Proven record of delivering business solutions on time and on budget
- Original designer and developer of jcrew.com and the suite of products now known as IBM InfoSphere DataStage
- Creator of the Enterprise Architecture Management Framework (EAMF) and main contributor to the creation of various maturity assessment methodology
- Developed partnerships between several companies and New York University to incubate new methodologies (e.g., EA maturity assessment methodology developed in Fall 2008), develop proof of concept software, recruit skilled graduates, and increase the companies' visibility

How to reach me?

	Cell	(212) 203-5004
	Email	jcf@cs.nyu.edu
	AIM, Y! IM, ICQ	jcf2_2003
	MSN IM	jcf2_2003@yahoo.com
LinkedIn	LinkedIn	http://www.linkedin.com/in/jcfranchitti
twitter	Twitter	http://twitter.com/jcfranchitti
skype	Skype	jcf2_2003@yahoo.com

What is the class about?

- Course description and syllabus:
  - http://www.nyu.edu/classes/jcf/g22.3033-003/
  - Web site will be replaced by a new wiki shortly
- Textbooks:
  - TBA

Knowledge Required

- Programming Languages (g22.2110)
- Operating Systems(g22.2250)
- Programming for the WWW
- Ability to program in Java and/or C#
- Some exposure to XML and associated technologies

Other Useful Knowledge

- Web server configuration and the HTTP protocol
- Scripting languages (e.g., JavaScript, Perl, TCL, etc.)
- Database theory (normalization rules)
- Web publishing
- Enterprise applications design

Agenda

Understanding Application Servers

- Wikipedia Definition:
  - "An **application server**, in an n-tier software architecture, serves an API to expose business logic and business processes for use by third-party applications"
  - However, not all application servers expose APIs today?!
- Application Server vs. Legacy Servers
  - Database server and transaction processing monitors are degenerated application servers
  - However database servers and transaction processing monitors pre-date application server technology
    - Why?
- Role of Application Servers
  - Manage non-functional requirements so that developers can focus on functional requirements

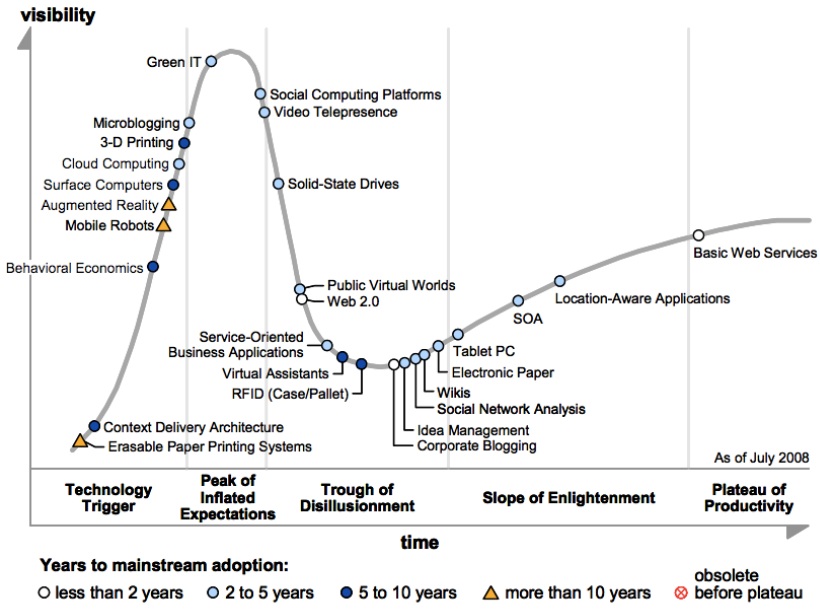
Application Servers Evolution

- Traditional client-server technology
- CGI frameworks
- Page-based extended HTML environments
- Distributed object computing platforms
- Java-Based
- Object Management Architectures (OMAs)
- Component-based computing environments
- Web Services platforms
- Next generation application servers (reflective, multimedia- and agent-enabled, MDA-compliant, etc.)

Application Servers Generics

- Modern Application Server Properties
  - Rich/portable software
  - Middleware between pervasive devices and back-office systems (OMA-compliant)
  - Platform independent programming interface
  - Support legacy applications integration (EAI/B2Bi)
  - XML-enabled
  - Web-services-enabled
  - SOA-compliant
  - etc.

Application Servers and Gartner Hype Cycle



Years to mainstream adoption:

○ less than 2 years ○ 2 to 5 years ● 5 to 10 years △ more than 10 years ⊗ obsolete before plateau

Source: Gartner (July 2008)

Agenda

Pattern Categories and Framework

- Model View
  - Reference Architectural Style and Element(s)
  - Architectural Style
  - Architectural Pattern
  - Design Patterns
- Implementation View
  - Reference Implementation Style and Element(s)
  - Implementation Style
  - Implementation Pattern
  - Idiom

Architectural Styles

- An architectural style is a description of component types and their topology
- It also includes a description of the pattern of data and control interaction among the components and an informal description of the benefits and drawbacks of using that style
  - Architectural styles are important engineering artifacts because they define classes of designs along with their associated known properties
  - They offer experience-based evidence of how each class has been used historically, along with qualitative reasoning to explain why each class has its specific properties

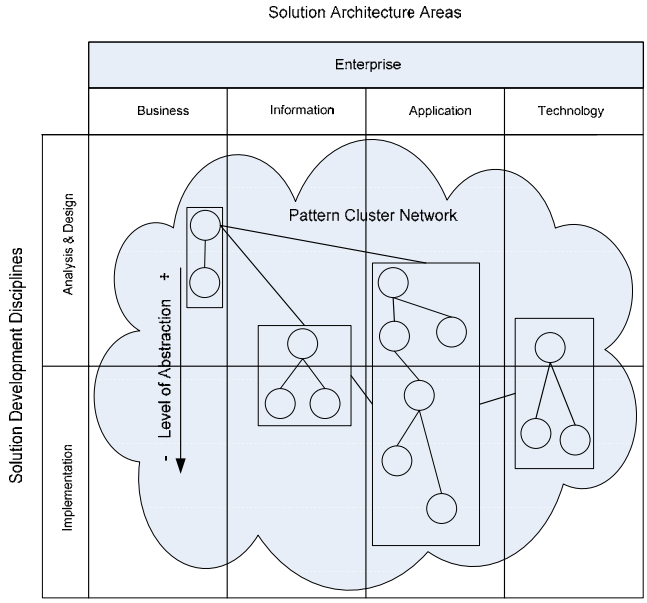
ABASs

- Attribute Based Architectural Styles (ABASs)
- ABASs build on architectural styles to provide a foundation for more precise reasoning about architectural design by explicitly associating a reasoning framework (whether qualitative or quantitative) with an architectural style
- These reasoning frameworks are based on quality attribute-specific models, which exist in the various quality attribute communities (such as the performance and reliability communities)

Pattern Clusters

- A **Pattern Cluster** is a set of patterns which are involved in the generic solution of a given problem type
- There may be more than one generic solution to a given problem type
  - e.g., Enterprise Solution Architectures typically provide combinations of pattern clusters of one or more known applicable Reference Architectures
- Identifying applicable patterns typically involves a two step process
  - Step 1: Identify applicable existing Reference Architectures
  - Step 2: Identify applicable pattern clusters based on these existing Reference Architectures

Pattern Clusters Catalog



Sample Business Architecture Patterns (1/3)

- Business Process
  - A long running set of actions or activities performed with specific business goals in mind
    - Business processes typically encompass multiple service invocations
      - e.g., *Initiate New Employee*, *Sell Products or Services*, and *Fulfill Order*
  - In an SOA context, a business process consists of a series of operations which are executed in an ordered sequence according to a set of business rules
    - The sequencing, selection, and execution of operations is termed service or process *choreography*
      - Typically, choreographed services are invoked in order to respond to business events

Sample Business Architecture Patterns (2/3)

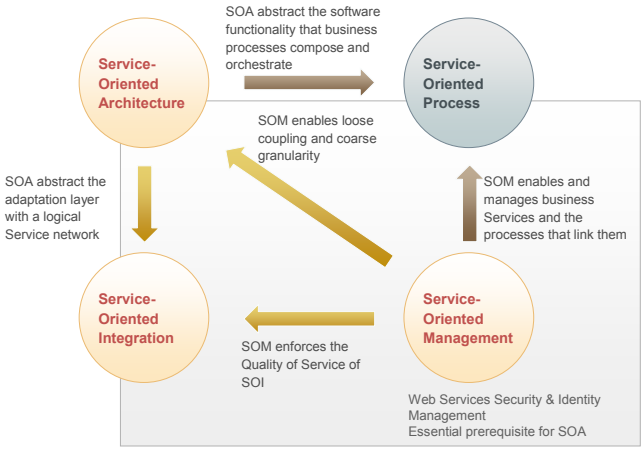
- Choreography
  - A choreography is the observed sequence of messages exchanged by peer services when performing a unit of work
  - Services do not need to be orchestrated to perform a unit of work (this is a concept that emerged and should have stayed in the last century)
    - This is a very common misconception, actually most units of work are accomplished by a series of "orchestrated services" performing a choreography
    - There are several industry efforts in the area of choreography languages, such as BPML (defined by BPMI.org), BPSS (defined by ebXML), IBM's WSFL, Microsoft's XLANG, and IBM/Microsoft/Oracle-BEA's BPEL4WS and their companion specifications WS-Coordination and WS-Transaction, etc.

Sample Business Architecture Patterns (3/3)

- Orchestration
  - An orchestration is a generalization of composition that sequence services and provide additional logic to process data that does not include data presentation
  - The same language can be used to perform a complex unit of work achieved by invoking a series of service operations
  - Any given orchestration is not forced to expose a service interface
  - If it does, it is a composition
- An orchestration is executed by an orchestration engine
  - BPEL is an orchestration programming language

SMA Reference Architecture

SOI-Driven Approach



SOA – Service Oriented Architecture
SOP – Service Oriented Process
SOI – Service Oriented Integration
SOM – Service Oriented Management

SOM Capabilities:

- Encapsulation and Composition
- Web service instance management
- Dynamic routing
- Transport protocol translation
- Web services caching
- Synchronous / asynchronous conversion

SOA Entry Points

Vertical slicing - Identify and re-engineer Business Processes across Presentation to Infrastructure layer using SOA paradigm.
Horizontal Slicing - Approach SOA from Infrastructure, Information, Integration, Application and Portal perspectives.

Entry Point	Description	Value
BPM	Model, Simulate, Optimize and Implement Business Processes leveraging existing assets and new services	Optimize Core and Out Source Ancillaries – Produces measurable business value
Infrastructure As Service	Consolidate Servers, Leverage, Virtualization, Grid and Utility, Strategize Provisioning Approach for IT H/W	Reduce TCO and Improve manageability and dynamic scalability
Information As Service	Build Services for Business Entities, Abstract underlying Data Sources	Business Aligned Data Managed Strategy
Integration / ESB	Use ESBs to integrate COTS, SAP, Legacy Applications and Web Services	Integrates IT systems across enterprise / businesses in a standardize way improves connectivity
Application	Build SOA infrastructure using SAP and Domain Frameworks as the basis	Reuse existing assets and COTS solutions
Portal	Consolidate delivery channels through portals and other technologies	Improves Collaboration and Standardizes Service Delivery

Pattern Frameworks References and Related Material

- IBM Patterns of eBusiness
  - http://www.ibm.com/developerworks/patterns/
- Microsoft Enterprise Architecture Framework (Blog Notes)
  - http://www.mikethearchitect.com/2009/03/microsoft-enterprise-architecture-framework.html

Agenda

Underlying Facilities

- (Network) Communication Protocols
  - e.g., TCP/IP, HTTP, RPC, GIOP/IIOP, RMI, XML, XML-RPC, SOAP/DIME/ROPE, UDDI/DISCO, WSDL
- Client-Server Technology
- Distributed Object Computing
- Component Models and Frameworks
- Secure Messaging Infrastructures
- etc.

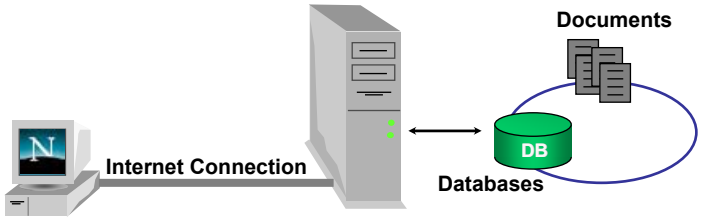
Old Client/Server Model

- **Connection**
  Customers (and partners) were required to have dedicated lines, leased lines, dialups, or some other access to a company.
- **Network Protocol**
  Customers had to use the same network protocols as the company they wanted to communicate with, TCP/IP, IPX/SPX, NetBUI, etc.
- **Hardware Requirements**
  Customers had to meet specific machine requirements: microprocessor speed; screen resolution; RAM; hard disk space; modem speed; etc. Customers were required to make a huge investment in hardware.
- **Operating System Requirements**
  The customer had to run a specific Operating System and version.
- **Software Updates**
  Application updates via floppy disks or CDs had to be infrequent. The customer was required to perform the costly installation or update.

Web 1.0 Business Evolution

- **First it was Online Publishing**
  - The World Wide Web - a Global Information Network Emerged (The Information Superhighway)
  - The Web Browser Provided Platform-Independent Access to Information
  - People Could View the Same Content Anywhere in the World
  - There was Explosive Growth in the Number of Business Web sites
- **Then it became Online Services and Web Applications**
  - Businesses are Building Relationships with Web-based Customers
  - Value-Based Services are Ensuring a Steady Flow of Web-based Traffic
  - Overhead is Reduced with Automated Online Services
  - A New Global Marketplace is Emerging - Web Applications/services are Available from Anywhere in the World 24x7
  - Web Application Updates Occur Instantly and Universally
  - Doing Business is now Cheaper, Faster, and Easier

Basic Web Applications



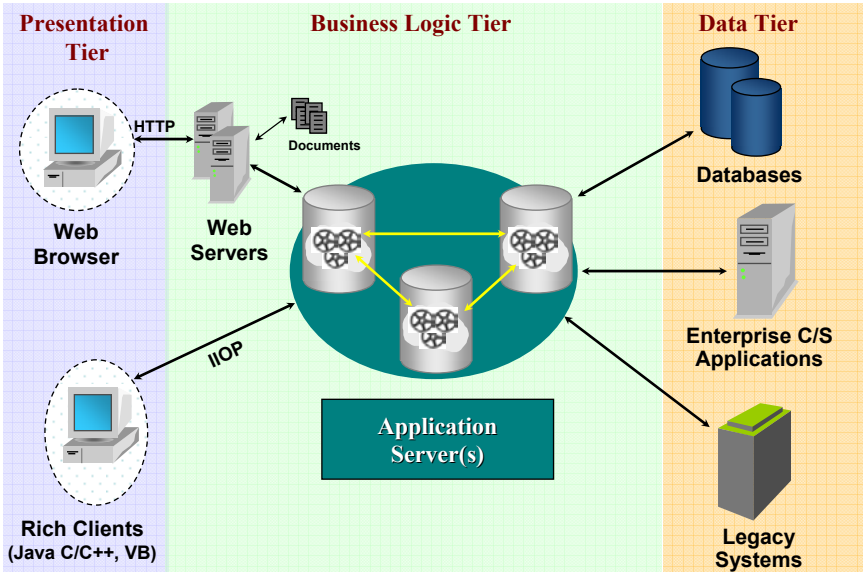
Client Browser

Web Server

CGI, SSJS, NSAPI, ISAPI, WAI, Etc.

Enterprise Data

Robust Web Applications



Mission Critical Requirements

- **High Performance and Scalability**
  - Benchmarks demonstrate 6000+ concurrent users, 12,000 TPM on a 4-CPU Sparc
- **High Availability & Reliability**
  - Customers like E*Trade & ISN demand 24x7 reliability with software & hardware fault tolerance
  - eBay on WebSphere
    - 30+ billion transactions per day
    - Over 8,000 tps!
  - Countrywide Insurance on WebSphere
    - Over 20,000 tps!

Mission Critical Application Requirements (continued)

- **Rapid Development through Pre-built Application & System Services**
  - Proven that large-scale enterprise applications can be built in half the time with equivalent resources
- **Enterprise Integration**
  - Need for high-performance integration to databases, legacy systems, client/server applications and ERP applications
- **Open & Extensible**
  - Need for standards-based, cross-platform supporting Windows/UNIX, JAVA/C++, CORBA/IIOP, RMI/IIOP, and .Net/COM+

Application Server Selection Criteria

- Usability
- Scalability
  - Concurrency
  - Extensibility
- Security
- Manageability
  - Fault tolerance, auto-deployment, communications, development environment, monitoring tools
- Reusability
- Support
- Skills

Application Server Categories

- Legacy technology
- Page-based extended HTML environments
- OMA-based
- Web Services platforms
- MDA-based
- Next generation
- Sample Classification:
  - » http://en.wikipedia.org/wiki/Comparison_of_application_servers

Application Servers Examples

- CGI-Perl custom environments
- ColdFusion 8, PHP 5, ASP .Net, JEE JSP
- WebSphere Application Server V7
- Oracle WebLogic 11g
- Red Hat JBoss Enterprise Application Platform
- etc.

Application Server Packages
(e.g., SAP Software Solutions, Oracle/PeopleSoft, Infor's Baan)

- ERP, and B2Bi Suites
- Human Resources
- Sales Automation
- Financial/Accounting
- Retail/Point of Purchase
- Manufacturing/Inventory
- Supply Chain Management
- etc.

Agenda

Sample Features

- **High performance & scalability**
  Create applications that deliver data quickly and scale to hundreds and thousands of concurrent users.
- **Maximum availability (24x7)**
  Create applications that are available 24 hours a day, 7 days a week, even when while being updated!
- **Client Independence**
  Access applications using web browsers or rich Java/C++ clients.
- **Rapid application development (RAD)**
  Develop applications quickly and easily with pre-built system and application services, application builder, extension builder, and a variety of third-party tools.
- **Enterprise Application Integration**
  Connect to backend databases, existing client/server applications, and existing legacy systems.

Value Proposition

What Customers are Building with it...

Customer Self-Service

- on-line credit card
- customer care & billing
- portfolio management
- benefits administration

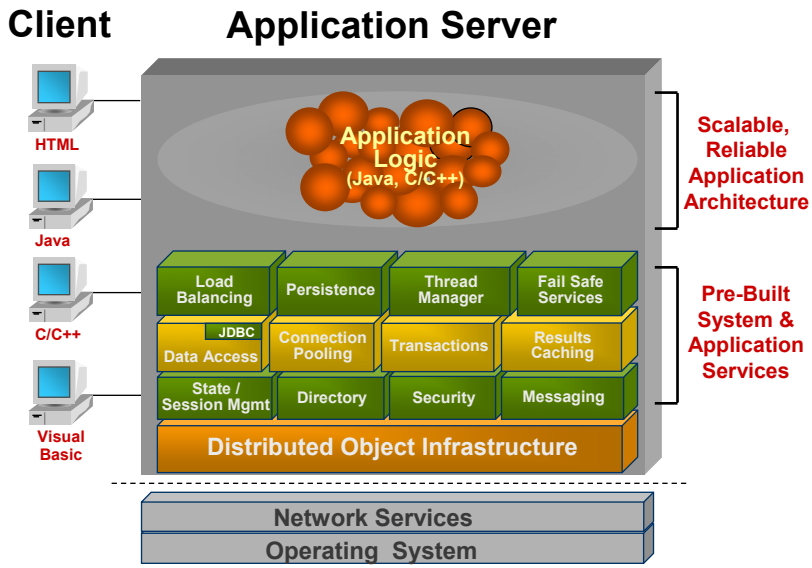
Business-to-Business Efficiencies

- package tracking
- claims processing
- supply chain management
- sales automation

Revenue Expansion

- on-line retailing, on-line trading
- loyalty programs
- travel and entertainment

Internal Architecture



Services

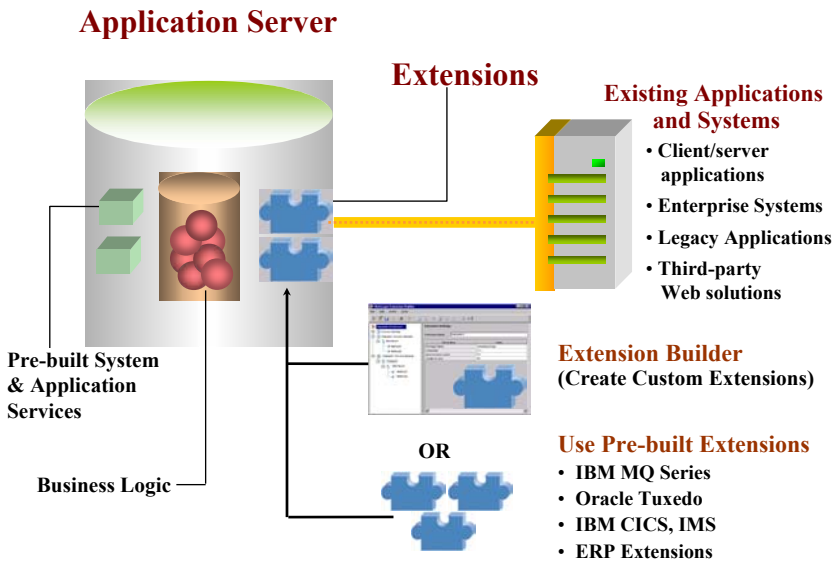
- **Key Application Services**
  - Java, C/C++ Client Support
  - Rich client Support
    - e.g., Adobe Flash/Flesh/Air, Microsoft Silverlight
  - State/ Session Management
  - Database Request Management
  - Transaction Management
  - Connection Cache
  - Results Cache
  - Dynamic Content Generation
  - Streaming
  - Security

Services (continued)

- **Key System Services**
  - Multi-process, Multi-threaded
  - Dynamic Load Balancing
  - Application Partitioning
  - Asynchronous Processing
  - Event Logging & Tracking
  - Kernel Services
  - Directory Services
  - E-mail Messaging
- **Key Administration Services**
  - Application Management
  - Server Management

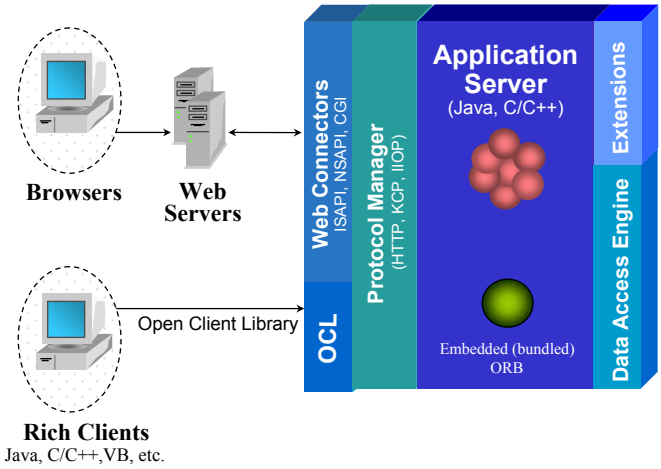
Extension Builder

Extending Application Servers Functionality



Technical Architecture

Open and Extensible



Extensions
- TP Monitors
- Legacy
- Client/server

Data Sources
- RDBMS
  - Oracle
  - Sybase
  - IBM Informix
  - IBM DB2
  - SQL Server
- OODBMS
  - ODI
- ODBC & JDBC

Platforms
- Oracle Solaris
- HP-UX
- SGI Irix
- Windows XP and Server 2008

Agenda

Sample Application Server Development Methodology Support

Develop, Deploy & Manage Business-Critical Applications

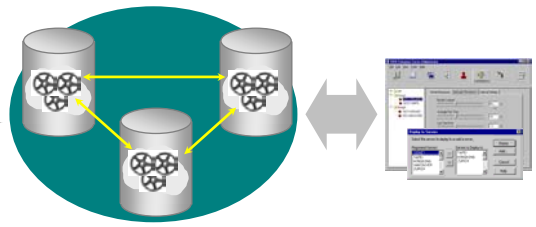
Develop | **Deploy** | **Manage**

Application Builder

Develop & deploy business applications in Java/C++

Extension Builder

Develop & deploy 'Extensions' in Java/C++ for enterprise application integration



Application Servers

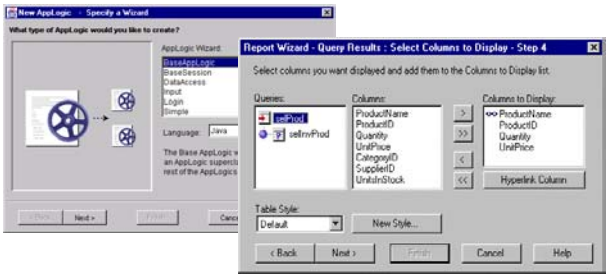
- High Performance & Scalability
- High Availability
- Client-Independence
- Rapid Development of Business Applications
- Enterprise Integration

Administrator

- Application Administration & Management
- Server & Cluster Management

Application Builder

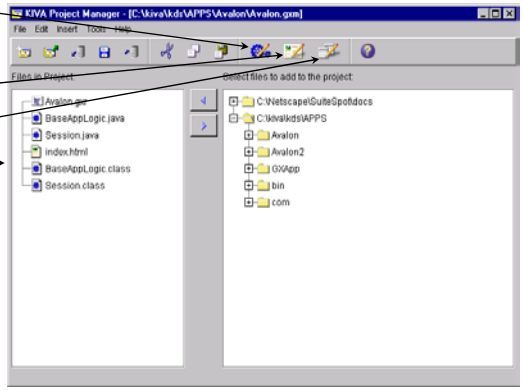
- **Rapid Development of Business Applications**
  - Wizards for Point-&-click Development
  - Pre-built Application Services and Class Libraries (Java, C/C++)
  - Reusable Application Components
  - Distributed Deployment of Application Components
  - Distributed Development Using Three-Tier Programming Model



Wizards for Point-&-click Development

Application Builder (continued)

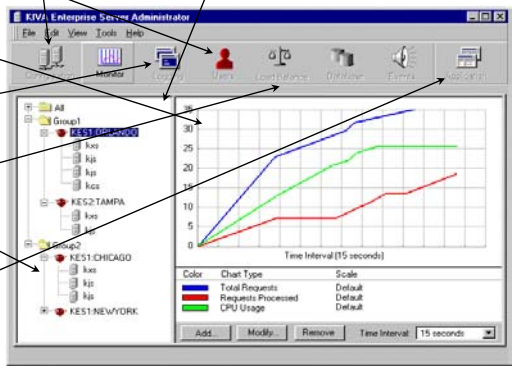
- **Robust Application Development Tools**
  - Application Logic Designer
  - HTML Designer
  - Query Designer
  - Project Manager
  - Third-party Tool Support



Project Manager

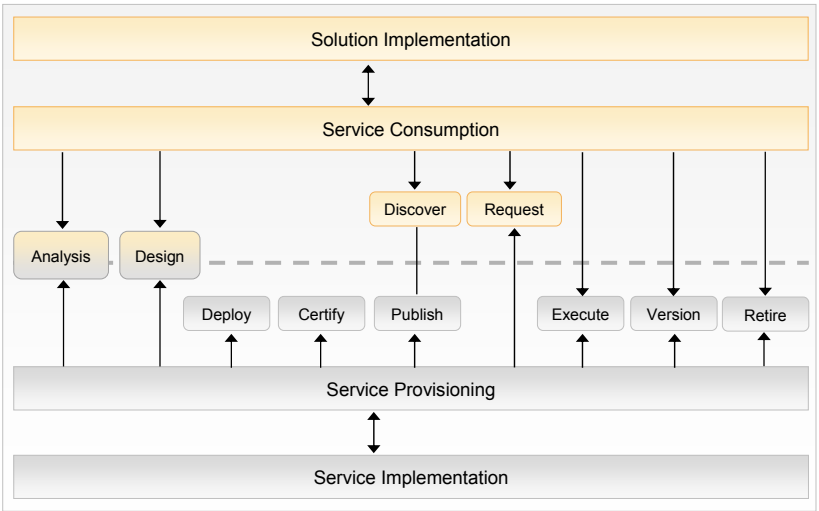
Server Administration

- Advanced cluster management
- Enterprise-wide views of all servers and processes
- Event monitoring & alerts
- Multi-view graphical performance monitoring
- ACL, User & Group Management
- Load balancing customization
- Application administration
- Application partitioning
- Integrated deployment

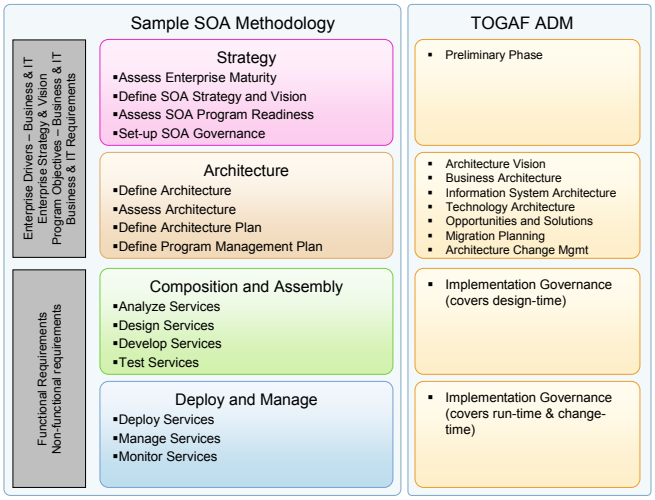


Performance Monitor

SOA Service Lifecycle

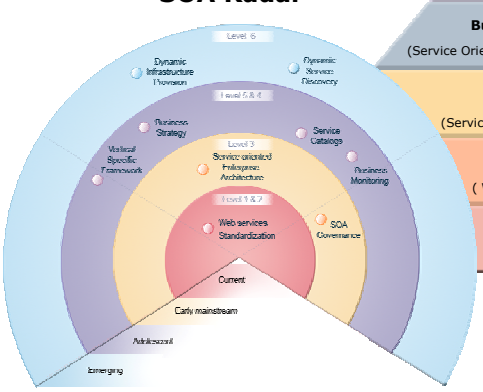


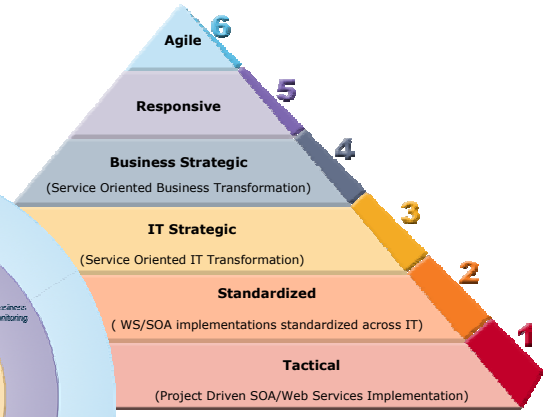
Sample TOGAF-Compliant SOA Implementation Methodology



Sample Out-of-the-box methodology & mapping to TOGAF

Sample Specialized Maturity Model for SOA

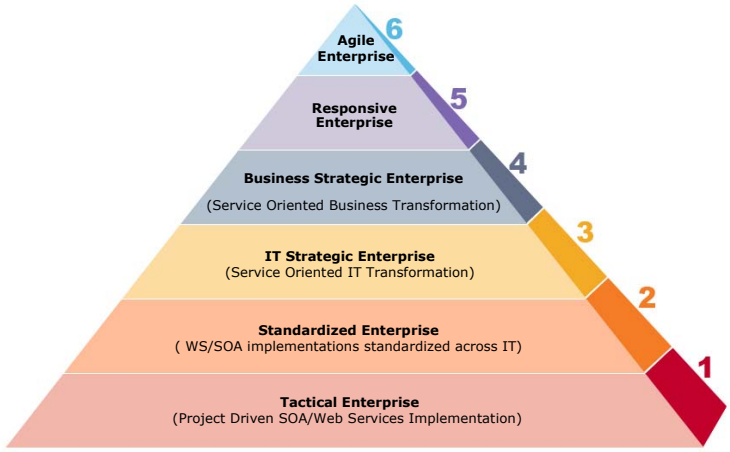




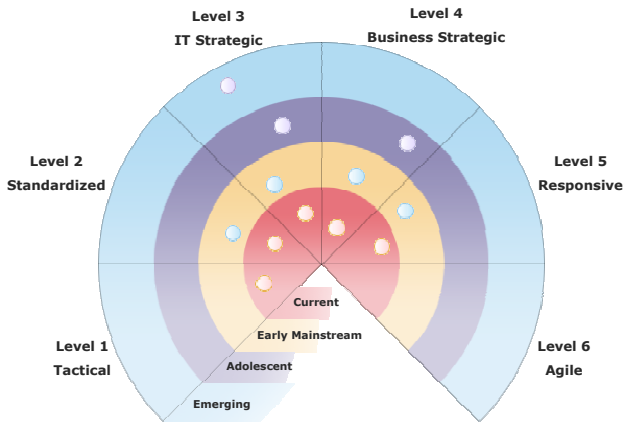
Defined Service Maturity Model for an Enterprise to establish benchmarks and standards to measure SOA adoption and maturity based on business impact

Sample SOA Maturity Model Explained

Defined Service Maturity Model for an Enterprise to establish benchmarks and standards to measure SOA adoption and maturity based on business impact

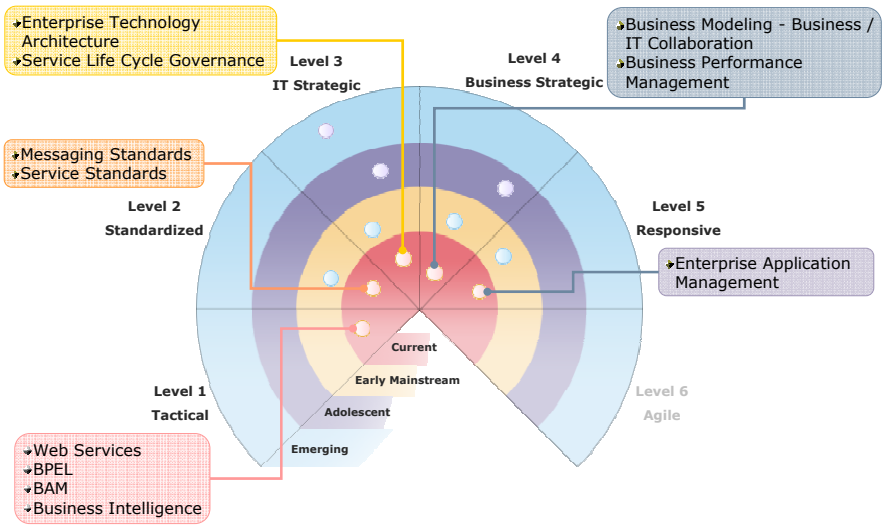


SOA Radar Explained

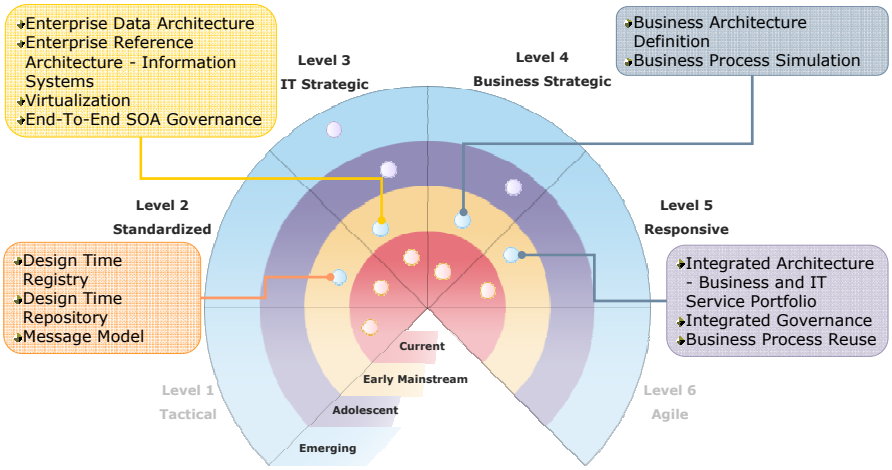


In order to get the most out of SOA paradigm, We need to understand our "Current State" and decide on the "To Be States" by taking in to consideration IT enablers

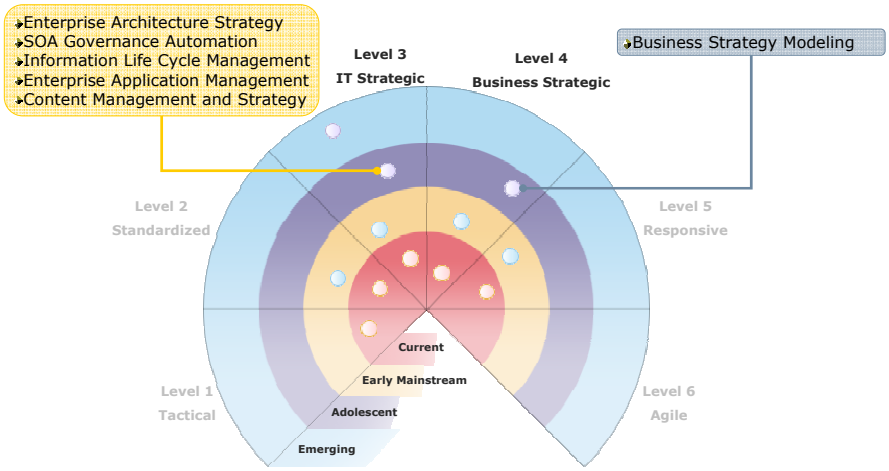
SOA Radar Explained



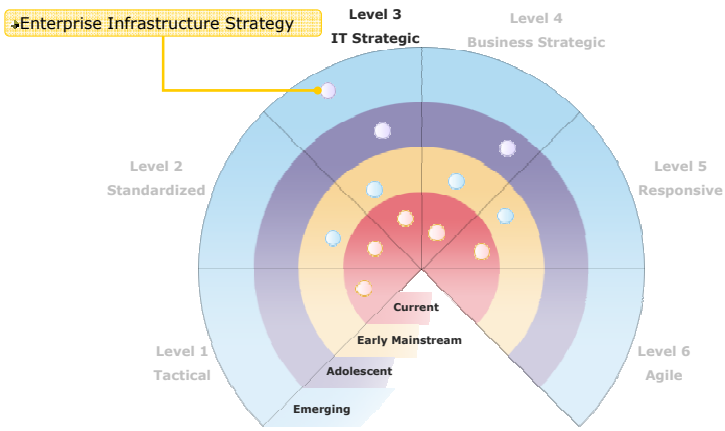
SOA Radar Explained



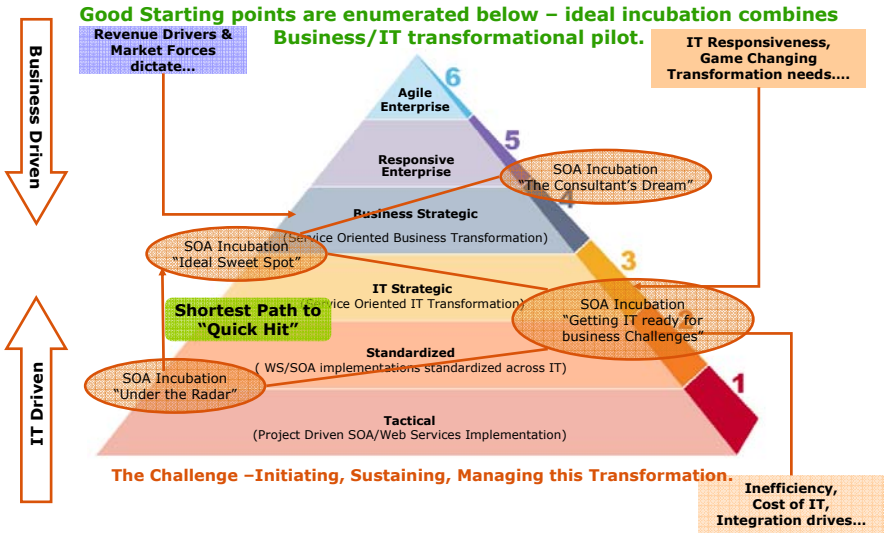
SOA Radar Explained



SOA Radar Explained



SOA Incubation – "Where to start?" "How to grow?"



Governance Components Detailed (SOA Specific)

Agenda

Summary – Key Application Server Objectives

- Enable Rapid Development of Business Applications
- Provide Industry Leading Performance & Scalability
- Provide High Availability & Reliability
- Enable Enterprise Application Integration
- Allow Client-Independence (HTML, Java, C++, VB, etc.)
- Provide Open & Extensible Architecture

Next Session: Architectural Mapping