

A large, stylized sun graphic in a lighter shade of blue, positioned on the left side of the slide. It features a circular center with several pointed rays extending outwards, and a wavy line at the bottom representing the sun's base or a shadow.

Why a Service-Oriented Architecture?

Jason Bloomberg
Senior Analyst
ZapThink, LLC

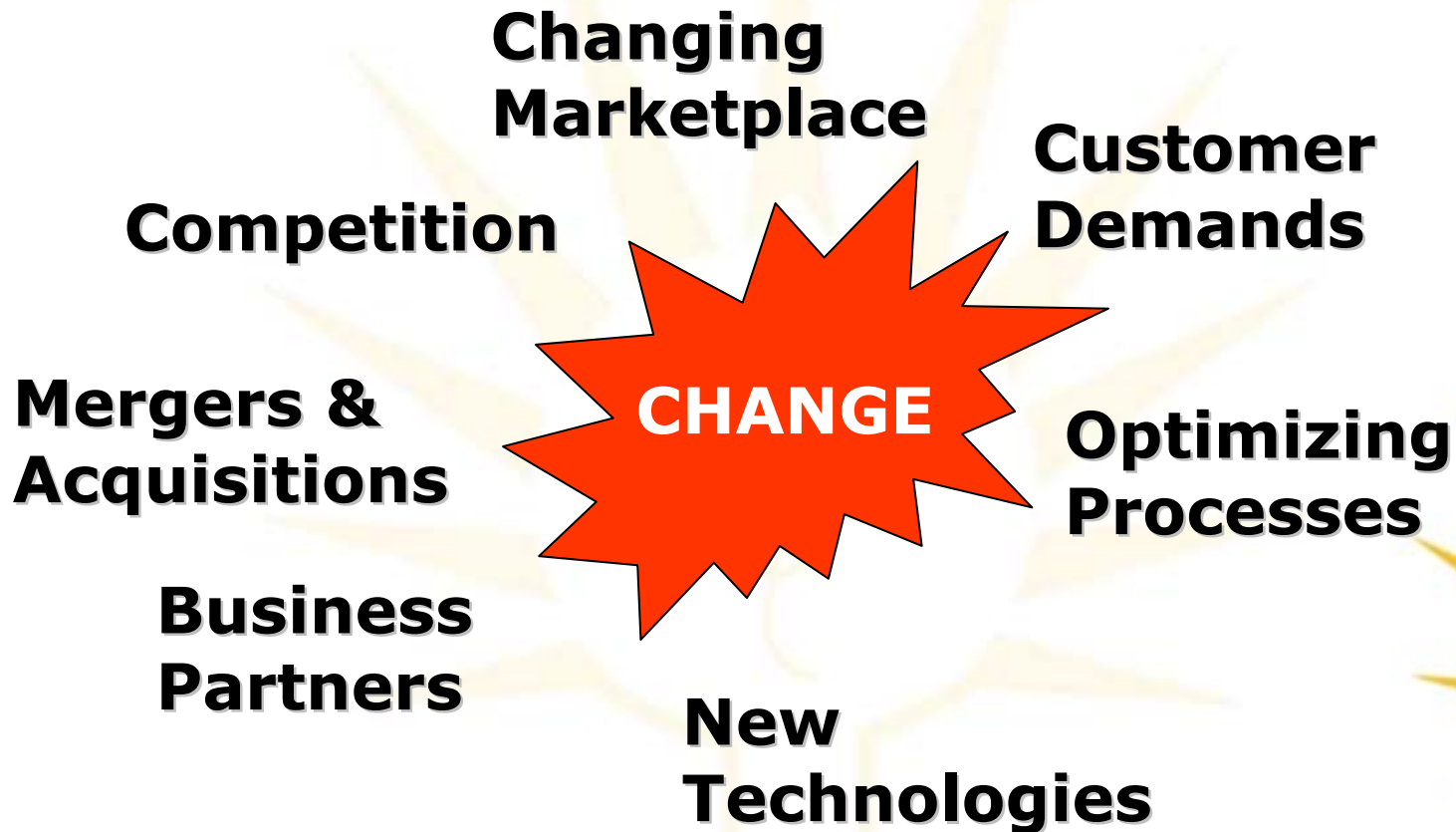


Agenda

- **What are the problems that SOAs solve?**
- **Why haven't these problems been solved before?**
- **Just what is SOA, and how is it related to Web Services?**
- **What steps should you take to implement an SOA?**
- **What are the key requirements for SOA?**
- **How might SOA transform your organization?**



Business Constant: Change



A Business is Never STATIC



IT: Fulfilling Business Requirements

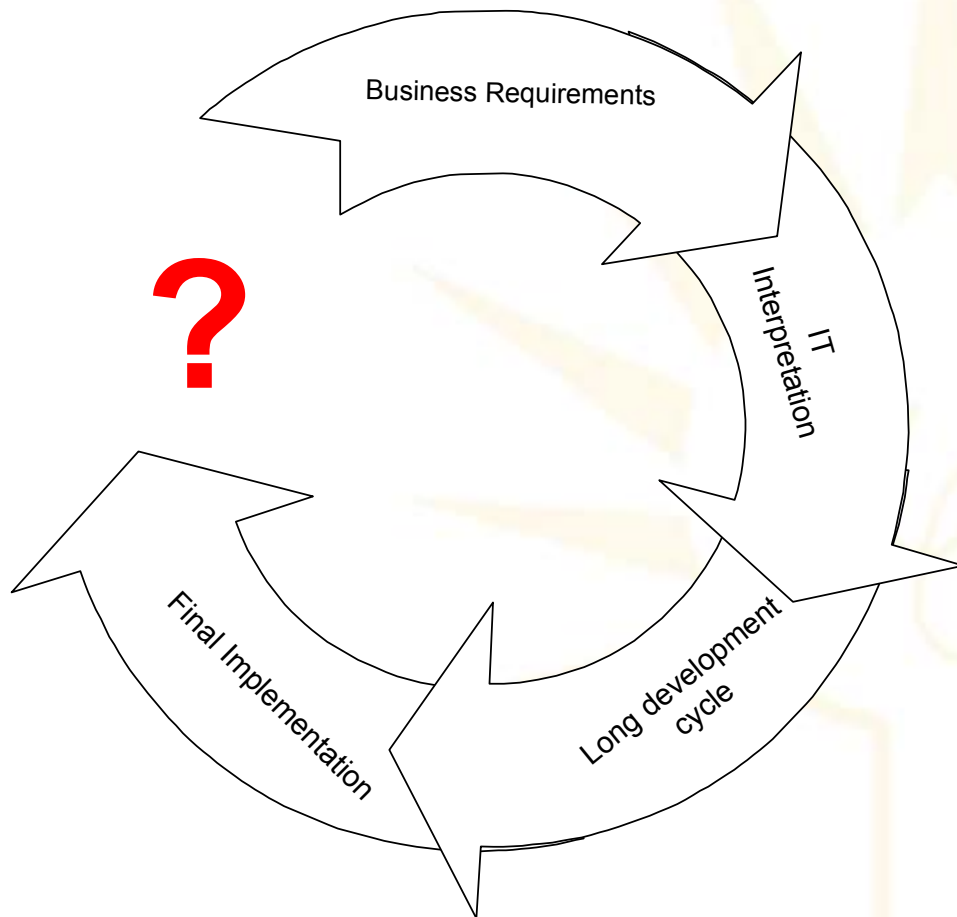
Business Requirements

- Service Customers
- Manage Operations
- Increase Worker Productivity
- Communicate with market
- Ensure reliable and secure operations
- Develop new products and services
- Respond to new business drivers

IT Capabilities

- Implement CRM Systems
- Implement ERP Systems
- Manage desktop environments
- Manage server environments
- Manage email systems and web sites
- Manage network and storage operations
- Develop applications

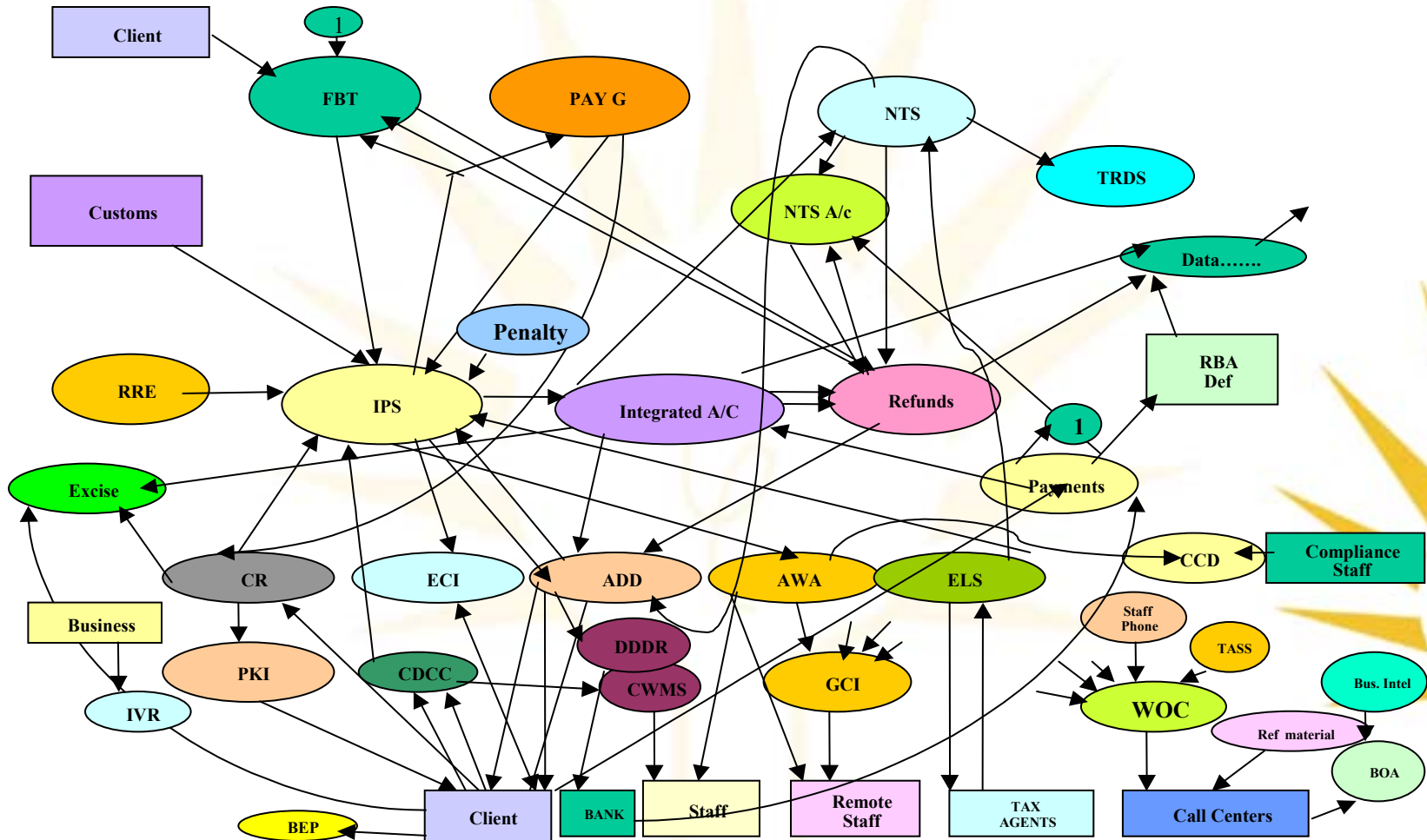
However, it rarely works that way...



- Requirements change
- Interpretations often inaccurate or limited
- Lengthy development cycles impervious to change
- Implementations "cast in concrete"

Result: IT that places limitations on Business

The Integration "Rat's Nest"





Integration Approaches of Yesterday

- Custom Integration: Coding to Interfaces
 - APIs: COM, Java, COBOL, Assembly?
 - Distributed Computing?: DCOM, CORBA
 - Screen-Scraping and Emulation (3270 and HTML)
 - Message-Queues
- EAI and B2Bi Middleware
 - Automating interface-level integration
 - Bus or hub-and-spoke architecture

Fundamentally *brittle* approaches to integration



What is a Service-Oriented Architecture?

- Access software via *Services* that are easy to find and connect to
- Web Services provide a *standard* way of building and accessing Services
- Users can build applications out of Services



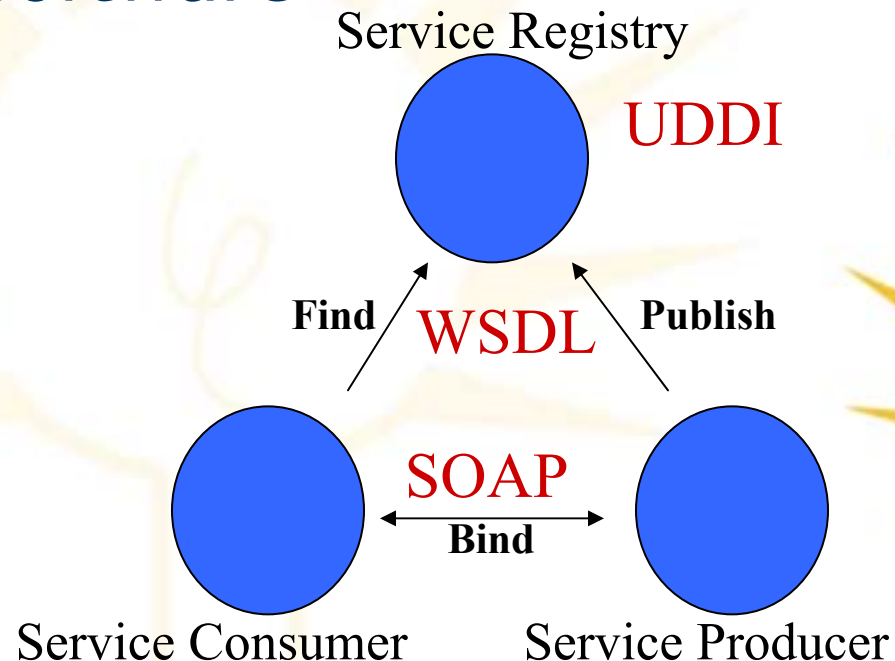
Have We Been Here Before?

- Service-Oriented Architectures have been around a while
- CORBA (Common Object Request Broker Architecture) and DCOM (Microsoft Distributed Component Object Model) two familiar examples
- What's new this time?



The Difference is Web Services

- *Standards-based* interfaces to software functionality





Web Services are the Trees....



Service Orientation is the Forest

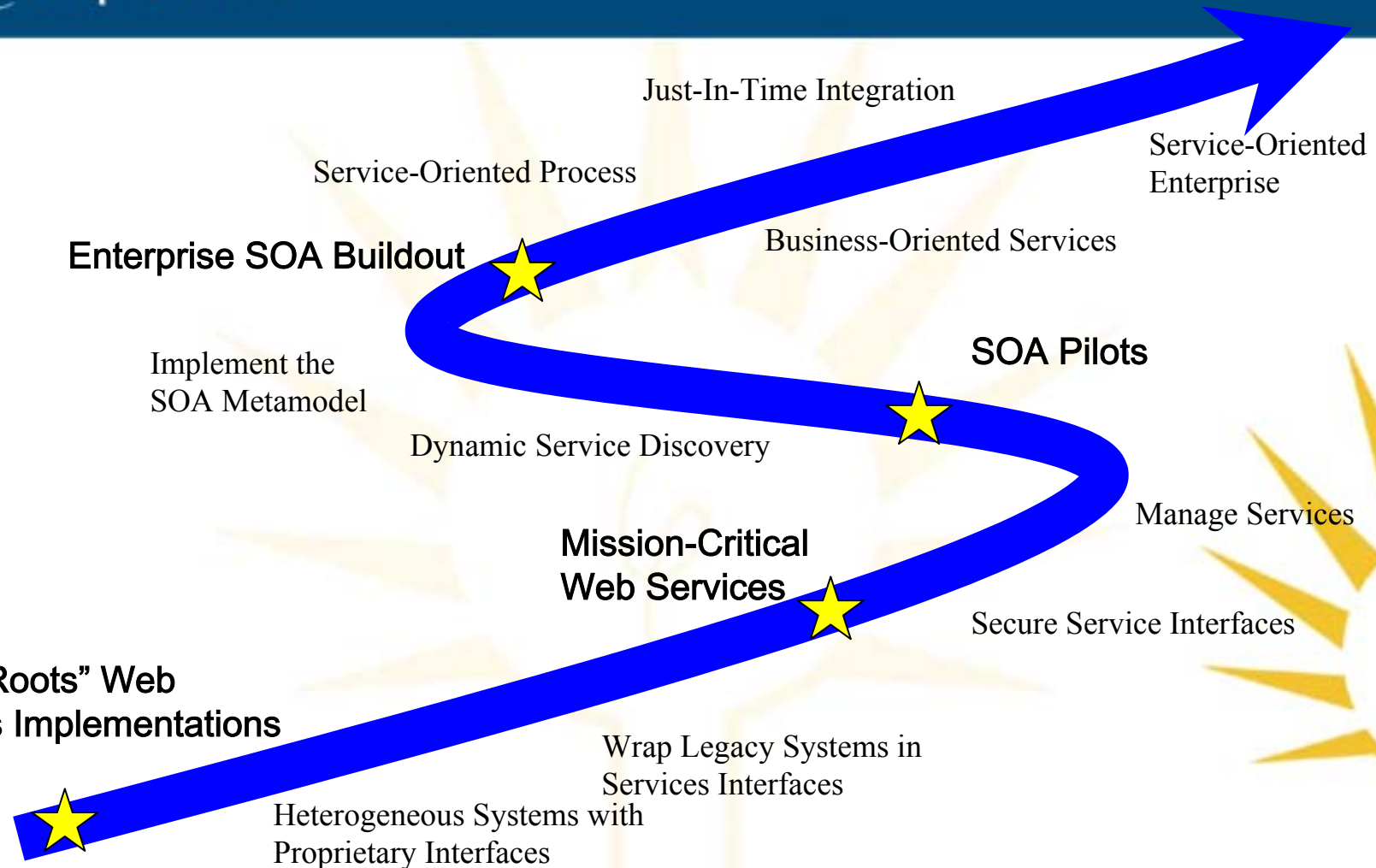


The Next Big Thing?

Approach	Timeframe	Programming Model	Business Motivations
Mainframe timesharing	1960s –1980s	Procedural (COBOL)	Automated business
Client/server	1980s-1990s	Database (SQL) and fat client (PowerBuilder, Visual Basic)	Computing power on the desktop
n-Tier/Web	1990s-2000s	Object-oriented (Java, COM)	Internet/eBusiness
Service orientation	2000s	Service-oriented (SOAP, WSDL, UDDI)	Business agility



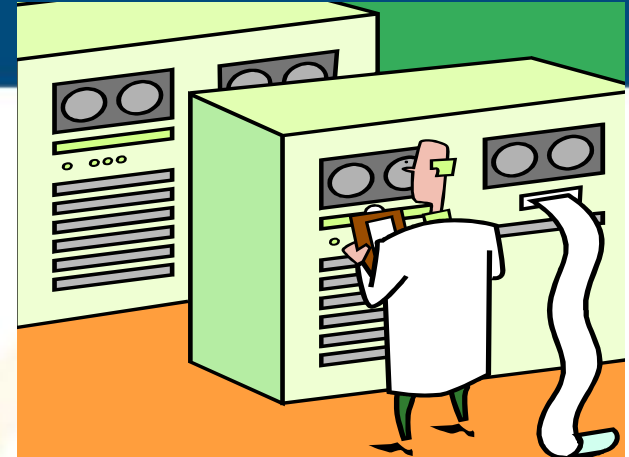
The SOA Implementation Roadmap





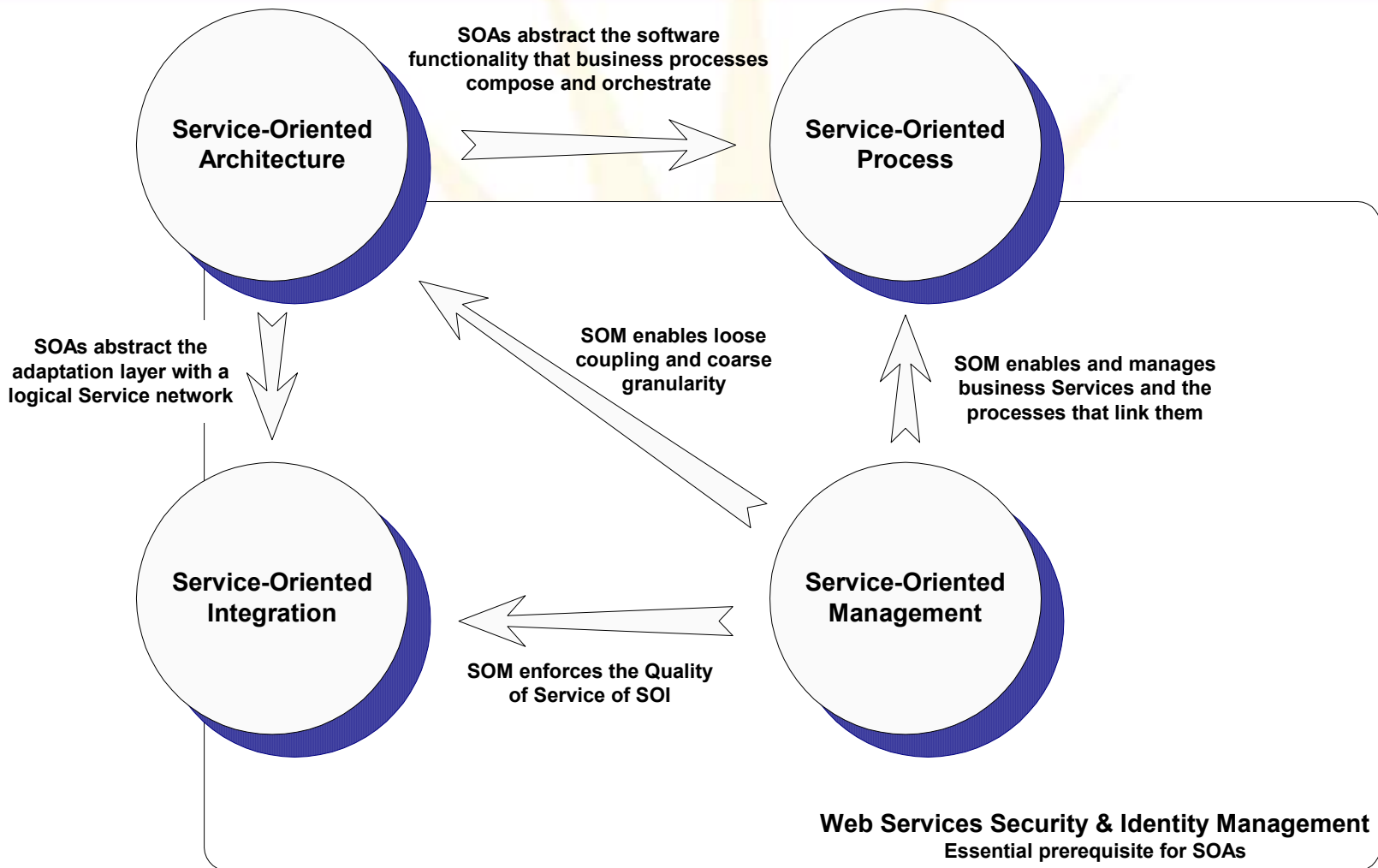
Getting Started: Web Services & Legacy

- What is "legacy"?
 - Host-based systems...
 - SCM, CRM, and other business apps...
 - Anything that's in use...
- Legacy systems enable a significant amount of mission-critical functionality
- Rip-and-Replace vs. Maintain-and-Extend



The first step to extending functionality: abstracting the implementation – aka "Service Wrappers"

Requirements for SOA



Requirement #1: Security

Traditional Distributed Computing

Network (Packet) Level

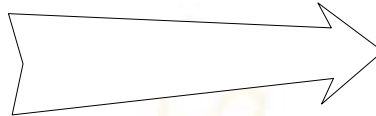
TCP/IP packet inspection
Perimeter-based

Closed Systems

Proprietary, closed APIs
Tight coupling
Single administrator
Localized users

Fragmented Control

Islands of security
Maintained as separate units



Service-oriented computing

Application Level

XML-based
Content aware
Application control

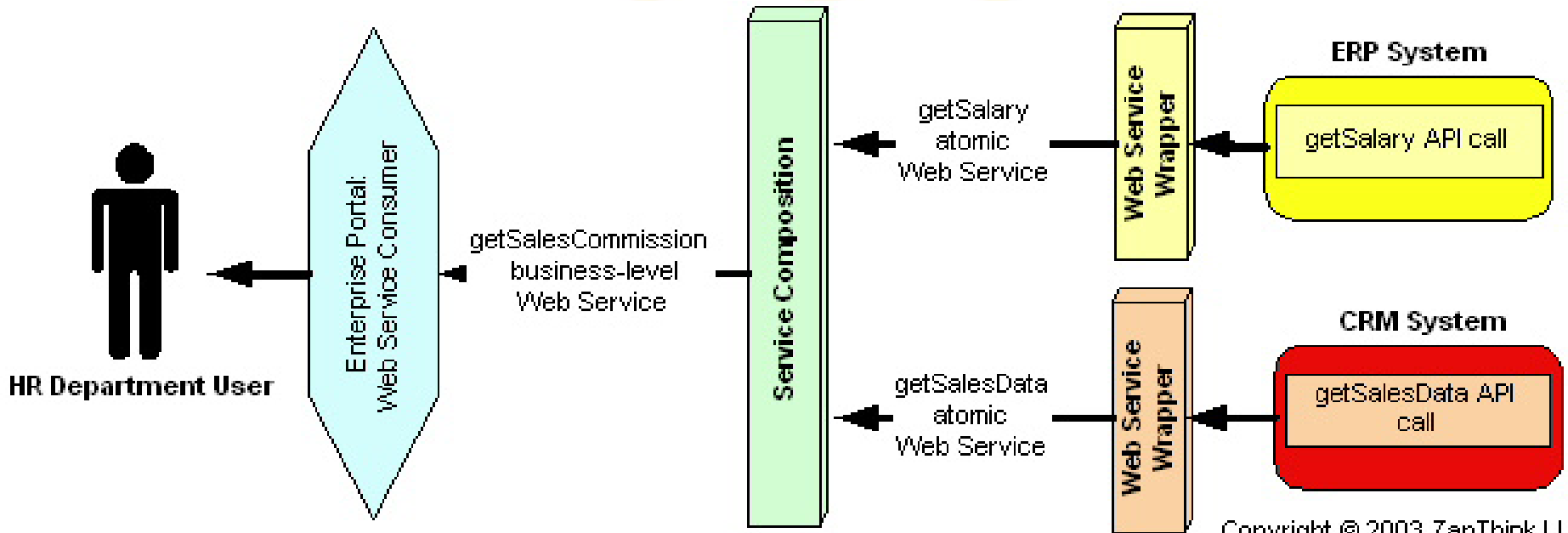
Open Systems

Open, accessible APIs
Loose coupling
Multiple administrators
Distributed users

Managed Security

Whole network policies
Tiered administration

The Problem of Context



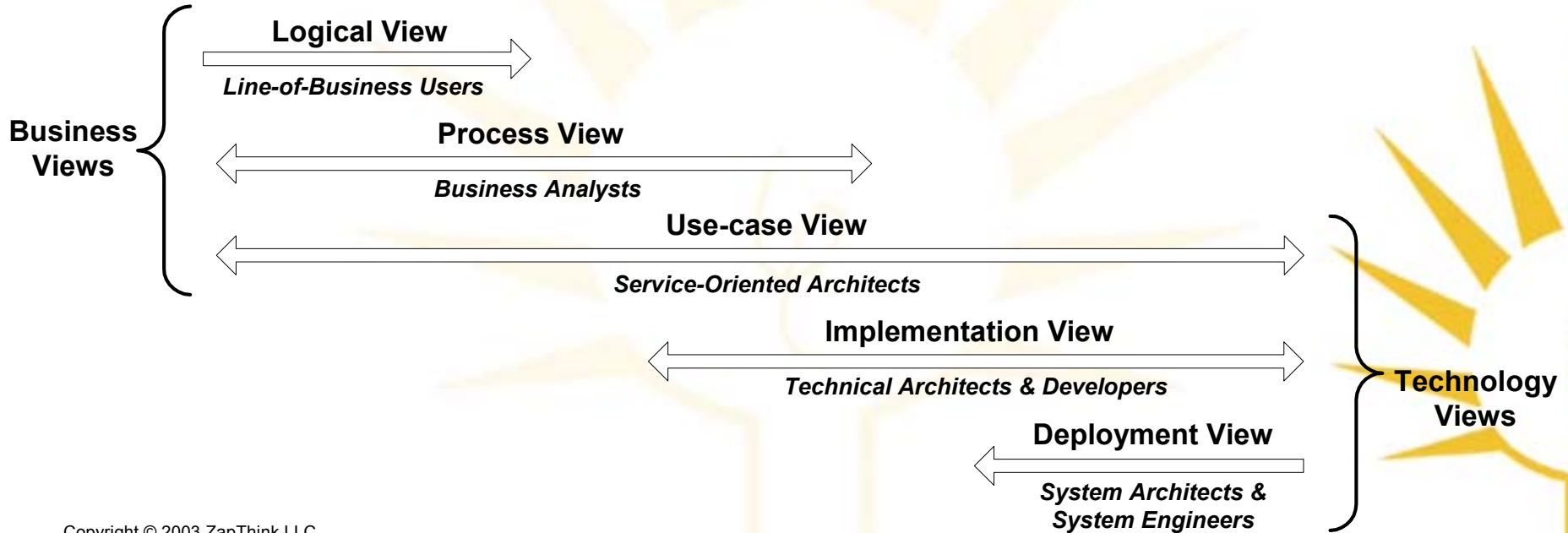
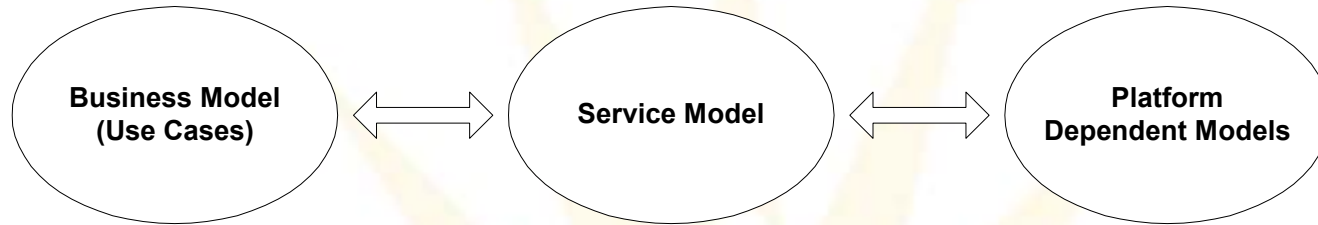
Copyright © 2003 ZapThink LLC



Requirement #2: Management

- Are your Services up and running?
- Are the right consumers accessing the right Services?
- How do you keep consumers & producers of Services loosely coupled when Services change?
- How do you fix things when something goes wrong?
- Are you providing the required quality of Service?

Requirement #3: Architecture





Requirement #4: Process

- Processes that are *coarse-grained*: composed of Services and *exposed* as Services
- Processes that are *loosely coupled*: a change to a process flow, activity, subprocess doesn't effect other processes
- Processes that are *asynchronous*
- Processes that are dynamically discoverable

***SERVICE-ORIENTED PROCESSES ARE
PROCESSES THAT CAN RESPOND TO
CHANGE***



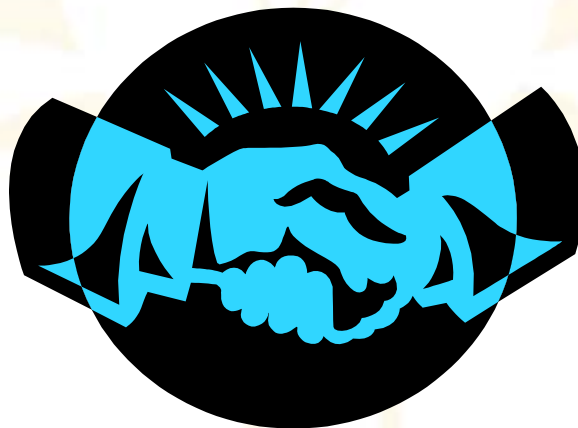
Managing SO Processes

- Achieving Loose Coupling
- Managing Atomic Services
 - Making sure the activities work!
 - Processes *are* Services
- Managing End-to-end Processes
- Keeping the abstraction in place



Business-Oriented Services

- Complete the abstraction layer, hiding implementation details from line-of-business
- Typically are SO Processes
- Location independent
- Updated and modified without breaking the loose coupling with consumers





The Service-Oriented Enterprise

- IT resources are available on demand to businesses as Services
- The Service-oriented abstraction layer enables companies to run their operations and conduct business with each other in a dynamic and automated fashion
- Business drives IT, and agile IT enables agile businesses

Thank You!



ZapThink is an industry analysis firm focused exclusively on XML, Web Services, and Service-Oriented Architectures.

Take Credit for attending this presentation!

- Go to www.zapthink.com/credit and enter the code **SOACORD**.
- Download a digital copy of the presentation
- Sign up for our ZapFlash newsletter
- Get a ZapCredit good toward free research and ZapGear!



Jason Bloomberg

jbloomberg@zapthink.com