© Copyright 2003, Productivity Solutions, Inc.

Patterns Are for More than Code: Building a Real-life Web Service-based Application

David Moskowitz

Productivity Solutions, Inc.

© Copyright 2003, Productivity Solutions, Inc.

Important Note

- I should change the name of the presentation to reflect the real nature of the application.
  - Did it have a Web-based component?
    - Yes!
  - 90% of the code had nothing to do with the Web
- Web-based was/is only a "component" of the overall application
- Service-oriented architecture includes Web-based applications.
  - The architecture "tricks" are in the notion of "service" not "Web."

© Copyright 2003, Productivity Solutions, Inc.

Agenda

- The challenge
- The initial questions
- The process
- Patterns: Make it duplicate-able

This is a companion presentation to "J2ME Case Study". See that one for the economic results

Based upon a real project. What's here is what worked!

We can talk about mistakes at the fireplace.

© Copyright 2003, Productivity Solutions, Inc.

The Challenge

- Management committee decided the company had to be more competitive
  - Losing share in some markets, not gaining enough in some, and stagnating in others.
- Question: How do we do that?
  - Start by asking the right additional questions
- Bottom line: They decided that "information everywhere" was the new corporate mantra.

© Copyright 2003, Productivity Solutions, Inc.

This Is Really a Story about…

- What can we do to take new and solid technology, …
- … apply it to real business problems…
- … and deliver results?
- What can we do that is on the edge, and not risk your job/company?
- At the time of project inception, they couldn't begin to think about it without Java
  - Impossible without standards

© Copyright 2003, Productivity Solutions, Inc.

The Starting Questions

- What does it take to build Web services that really work?
- What's really going on with B2B, B2E, B2C transactions?
- What makes a project successful?
- Are there patterns, outside the context of code, that contribute to project success?

Mike Brown: "Intro to Grid Computing with Globus Toolkit V 3"

© Copyright 2003, Productivity Solutions, Inc.

Web Services=>Distributed App

- Solve the business problem
- Address security considerations
- Provide high performance optimized for distributed (multiple) platforms
- High availability
  - Resilient for the inevitable problem(s)
- Easily managed: deploy & troubleshoot
- Easily maintained at each "touch point"

© Copyright 2003, Productivity Solutions, Inc.

Problem Approach

- Current state
  - Which applications & databases are already linked?
  - 20 to 30 databases, over 150 applications
- Needed to do some data remodeling
  - Different data stored in different formats and locations
- Created a suitable backbone
  - Do the math – 150 apps means 149 connections for each
- Broadcast messages internally, not directly
  - To get the disparate applications talking to each other would have taken more 11K adapters
  - Publish & Subscribe messaging
- This is old data! More databases and applications

© Copyright 2003, Productivity Solutions, Inc.

Cross-boundary Problem

- Integrate systems across department and organizational boundaries
  - Forced a service-based approach, in part, because distributed objects can be very complex
- Services can be built on different platforms, by different teams, on different schedules, and maintained independently.
  - The critical piece is to define the service interfaces (somewhat similar to Java Interface).

© Copyright 2003, Productivity Solutions, Inc.

Service & Service Integration

- The term ***Service*** is applied to any external software component that provides a business (model) related service.
- Use messages to provide communication between service components
- Services expose a ***service-interface*** that receives inbound messages.
- The set of messages that must be exchanged for the service is a ***contract***.

© Copyright 2003, Productivity Solutions, Inc.

Service-oriented Programming

- Internal implementation of the service is irrelevant to overall design.
  - Assumes (a) service meets published expectations for business functionality and (b) communication follows the contract.
- Internally, services typically contain traditional object-oriented components

© Copyright 2003, Productivity Solutions, Inc.

Key Points

- Services should be designed to communicate with each other with minimal coupling
  - Message-based communications helps decouple availability and scalability
- Each service is "self-contained"
  - Has its own data sources, business logic, manages its own transactions and, if appropriate, UI.
  - May have same internal design as traditional n-tier application.

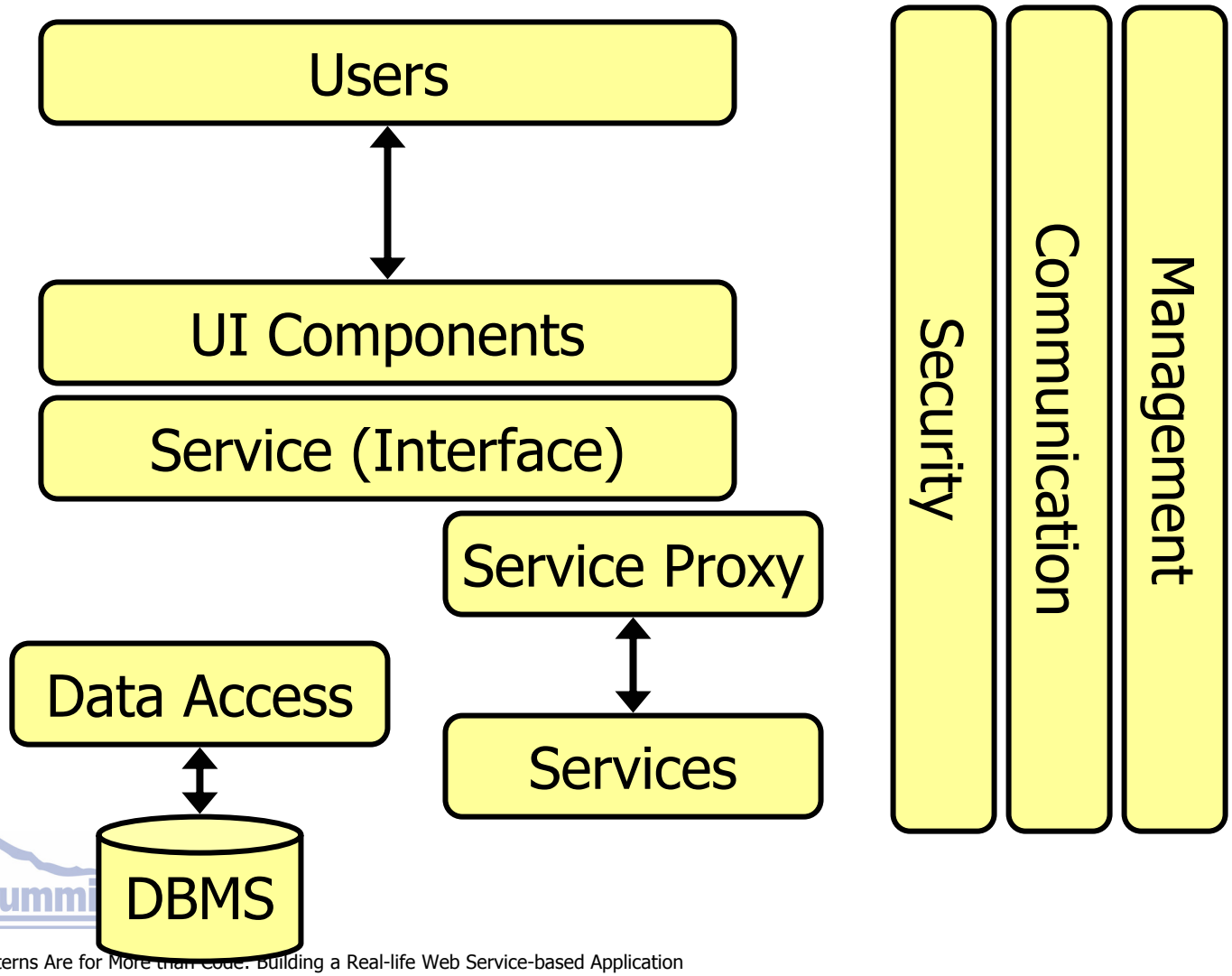
© Copyright 2003, Productivity Solutions, Inc.

Key Points

- A given service is only implemented once
  - Services are reusable
- Service types:
  - Application
  - Management
  - Deployment
- Services must be loosely coupled
  - Service, service interface, contract
- The contract must include QoS

© Copyright 2003, Productivity Solutions, Inc.

Generalized Architecture



© Copyright 2003, Productivity Solutions, Inc.

Success Patterns

- Identify service types
  - Some applications don't need certain components/services
- Design all services of a particular type to be as consistent as possible
  - Use a small set of design models

© Copyright 2003, Productivity Solutions, Inc.

Success Patterns

- Know how the services communicate with each other before choosing physical distribution boundaries
  - Keep coupling low & no chatty interfaces
- Minimize the number of data formats
  - Using XML strings, serialized objects, ResultSet, *etc.* in the same application radically increases difficulty to develop, maintain, enhance
    - Pick a VERY small subset of possibilities: 1 or 2???

© Copyright 2003, Productivity Solutions, Inc.

Success Patterns

- Abstract "policy" code from the "main" app
- Before you start, determine the type of layering you will enforce.
  - With very strict enforcement, layer A cannot directly call services or components in layer C
- Don't RYO; use existing standards whenever and wherever possible!!

© Copyright 2003, Productivity Solutions, Inc.

Design Presentation Layer

- You have to consider multiple platforms
  - Composed of UI and UI specific process logic
    - Process logic handles state management within UI for predictable UI interactions
  - Both OS and display device
    - Web browser
    - Mobile device
    - Smart device
- Consider two "Webs" eyeball and transaction

© Copyright 2003, Productivity Solutions, Inc.

Separate Process from Input

- This should be obvious
  - Everyone knows this at some level.
  - Caused major problems for folks
- The point is simple: You have to be religious about this
  - Particularly for handheld mobile devices
- What was good practice before, is rigorously required, now.

© Copyright 2003, Productivity Solutions, Inc.

Separate Process from Input

- Identify business process(es) that UI will trigger or accomplish
  - Sequence diagrams help understand how user sees the flow
- Identify the data needed by the business process(es) that must be collected by the UI
  - Remember the goal, everything everywhere
  - Used the user issues to include legacy systems
- Identify additional state information that must be maintained

© Copyright 2003, Productivity Solutions, Inc.

Separate Process from Input

- Finally, design the visual flow of the UI and the control interfaces (w/ platform in mind)
  - This is the only part that has any degree of "platform dependence"
  - Wrapper device controls behind your own code.
    - If you don't – the resulting multiple code paths could drive you nuts.
- Obvious: Designing Web service-based applications is different.

© Copyright 2003, Productivity Solutions, Inc.

On- and Off-line Connectivity

- Server must accommodate loss of mobile connectivity.
  - Another reason for message based design
  - Needs policy to determine when to abandon partial transaction if communications interrupted
  - Something not usually considered on server, easy to understand on the mobile/remote
- Design remote as an off-line app
  - "Replicate" when connection restored/established

Colorado Software Summit

© Copyright 2003, Productivity Solutions, Inc.

Something to Consider

- Up to this point it might appear that we've focused on a single design application.
  - Talking about service-based application
- Need to develop a set of services that integrate disparate systems.
  - This isn't trivial
  - Requires understanding and management of business workflows
    - Need people who either understand or coach big-picture understanding of the pieces.

© Copyright 2003, Productivity Solutions, Inc.

Something Else to Consider

- The real issues have absolutely nothing to do with technology
  - One set of problems will be caused by bureaucracy, organization culture, turf wars, and "legacy discoveries."
  - Can you guess the other source?
- **The challenge: learn how to reuse and commoditize existing information assets.**

© Copyright 2003, Productivity Solutions, Inc.

Oh, by the way…

- Security isn't an afterthought!
- Is a Web services security layer *really* required?
- There already is a set of widely accepted transport-layer security mechanisms for message based architectures
  - SSL and TLS
  - Why add another?
    - Kelvin Lawrence: "Making Web Services Secure"

© Copyright 2003, Productivity Solutions, Inc.

Security

- Workflow is the primary application paradigm for dynamic integration.
- Security services evolve into core elements of secure application workflow.
- A Web service security model must support:
  - Protocol-independent declarative security policy
    - Web service providers enforce;
  - Descriptive security policies attached to the service definitions
    - Clients use to securely access services.

© Copyright 2003, Productivity Solutions, Inc.

General Security Principles

- Use tested, proven security systems *vs.* RYO
- Never trust external input
  - Validate everything
- Assume external systems are insecure
  - Assume cleartext information is compromised
- Enable only needed attributes
  - Minimal permission
- Security by obscurity, isn't!!

© Copyright 2003, Productivity Solutions, Inc.

Management

- Monitor – instrument applications to allow admin staff insight into:
  - Application/service health
  - Scale/capacity management
  - Expectations
  - Efficiency

© Copyright 2003, Productivity Solutions, Inc.

Management

- Business monitoring: identifies service bottlenecks, *etc.*
- User process monitoring: duration, pause points (depends upon message definition), started and not finished, *etc.*
- Health monitoring: business services, components, workflows, *etc.*
- Data access monitoring: long running disk I/O, connection use, *etc.*

© Copyright 2003, Productivity Solutions, Inc.

Management

- Exceptions
  - UI components
    - Retry the operation
    - Expose "the issue" to the user
    - Stop, restart, or continue with UI app flow
  - Business process services
    - Technical (*e.g.*, failed DBMS connection)
    - Business (*e.g.*, violation of biz rule or constraint)
- Configuration (used J2EE provisioning)

© Copyright 2003, Productivity Solutions, Inc.

Communication Policy

- Define communications synchronicity, format, and protocol
- Remote communication using J2ME Wireless Toolkit (multiple versions) with SSL and TLS
  - Are one-way or two-way authentication mechanisms needed?
  - Sign messages
  - Encrypt sensitive parts or entire message

© Copyright 2003, Productivity Solutions, Inc.

Communication Policy

- Intra- different from inter- application
  - Both need to be defined
  - Only one is "global"
  - Sometimes use same message-based for both
- Services accessed using messages (WS-I)
- Using the same communications (bus) between tiers and services…
  - … made the application more modular
  - … high level of team / platform independence

© Copyright 2003, Productivity Solutions, Inc.

Tiers as Services

- Compelling long-term, but poses challenges
  - Business layers may rely on context (security) which may be unavailable when trying to invoke some logic
  - Communication must support all requirements of intra-application commo including:
    - Transaction flow, throughput, latency, & exception
    - Standards still evolving in these areas
  - Commo between UI and biz layers must be robust and resilient – should be standards-based

© Copyright 2003, Productivity Solutions, Inc.

Asynchronous Messages Advantage

- Scalability and availability
  - Utilize hardware resources better
  - Isolate application from software or infrastructure failures
- Location transparency
- Transport agnostic
- Similarity to business model
- SLA/Contract isolation

© Copyright 2003, Productivity Solutions, Inc.

Asynchronous Messages Disadvantages

- Deterministic outcome – need extra states with no return message is received
  - Means managing conversation state in addition to everything else
- Message correlation
  - Need to invent correlation mechanism that identifies a specific message that identifies a specific instance of a business conversation

© Copyright 2003, Productivity Solutions, Inc.

Asynchronous Messages Disadvantages

- Message delay – messages can arrive late
  - Implement business logic to handle messages that never arrive
  - Make sure message is still valid when it arrives
  - Need drop-dead time after which the order / request / transaction won't be processed
    - Underlying data or biz rules might have changed
- Transaction flow – means different transaction model
  - Can't send transaction & get response in one atomic transaction

© Copyright 2003, Productivity Solutions, Inc.

Asynchronous Messages Disadvantages

- Repeated messages – have to handle the special case where messages arrive more than once
  - Require transaction ID to be supplied as part of the message or specify both old and new data
- Message sequence – messages could arrive out of sequence
  - Handle in conversation state or business logic
  - Force sequencing in business logic with ACK
    - Defeats some of the advantages – use very sparingly

© Copyright 2003, Productivity Solutions, Inc.

Summary

- Competitive advantage was huge.
- Mobile enablement of the workforce was a clear business differentiator.
- It's a different development space with many pitfalls if you assume that old development patterns work
  - Change is required
- Business operates on repeatable processes
  - Source of resistance to change

© Copyright 2003, Productivity Solutions, Inc.

A Final Word

- This is also a story about change
  - Change is what must happen
  - Experimentation is how to accomplish it.
- Means a vocabulary change:
  - We had to drop these words: change, different, improved, new, and replace.
  - Replaced with these words: experiment, create, participate, and contribute.
- It's not a word game, it's a way of thinking and creating *versus* decreeing and imposing.

© Copyright 2003, Productivity Solutions, Inc.

Questions & Thank you

- For more information

David Moskowitz
Productivity Solutions, Inc.
147 Ashland Avenue
Bala Cynwyd, PA 19004
Voice: 610-664-7726
Fax: 610-667-1798
E-mail: davidm2@usa.net