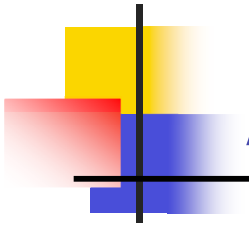
Patterns & Anti-Patterns in SOA

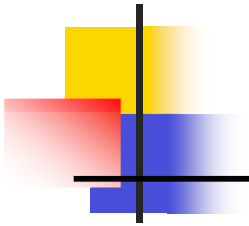
David Moskowitz

Productivity Solutions, Inc.



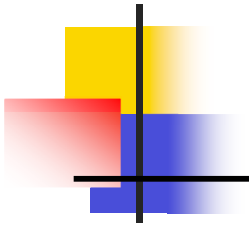
Agenda

- Common vocabulary
  - Sounds simple, but it can be part of the problem
- Identifying SOA patterns and anti-patterns
- Patterns to emulate
- Anti-patterns to avoid
- Drivers: Technology or (people and process)



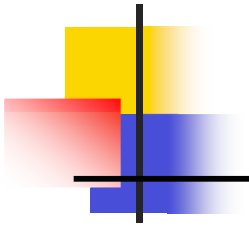
Common Vocabulary

- Service: reusable bit of software that can be invoked *via* a published interface
  - Service contract
  - Has a recognizable business function that plays a clear role in multiple applications
- Service reuse: the major payoff point of SOA
  - Purpose: decrease time to market, reaction time, and…
  - How do you define reuse and reusability?
- Service broker(ing)
  - Part of reuse
  - Publish/advertise service contracts



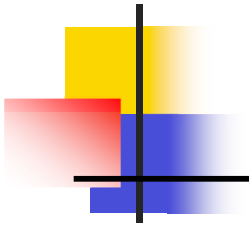
Software Architecture

- Defines system components & how they interact (100K ft level)
  - Components aren't any specific type of object
  - Abstract "modules" deployed as one or more units on one or more servers
- Defines the externally visible properties and their relationship(s).
  - Structure, relationships **and** patterns
  - Defines expectations expressed as a contract



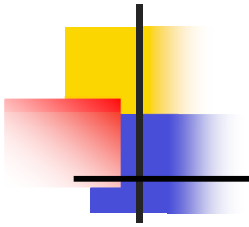
Service Oriented Architecture

- SOA formally separates interface & implementation
  - Service-consumer views service as an endpoint
  - Supports contract with defined request & response msgs
  - Dictates a Find Bind Execute (FBE) paradigm
  - Web Services take this one step further – define a formal mechanism for the FBE process
    - WS is something that can be described by WSDL
- SOA expresses the definition of loosely coupled software business service components/modules that map a business process or problem space



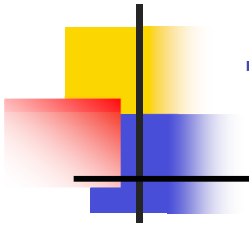
SOA Beginnings

- SOA: not a technology: how, not what
  - Defines the **starting point** as business value
- SOA: built on standards-based protocol stack
  - … but not limited to the stack
- SOA is also about business re-engineering
  - SOA isn't about applications, it's about identifying business processes that can be expressed as one or more (connected) services.
- Build, connect/deploy, manage, orchestrate
  - Ultimate goal: More agile and adaptive IT



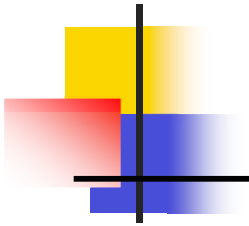
Success 1: Planning

- Conceptualize the big picture, BUT start with something manageable
  - Classic advice: TBSS
- Business processes: you can't know them all.
  - Collaboration and whiteboard
  - John Soyring's comments: lack of BP documentation
  - Map the bigger picture and then select the right pilot
- Take inventory! What already exists?
  - Reverse engineer code to derive business process
  - Is reuse possible? Can you leverage what is?
    - It's more than software, people, process, skills, ...
  - This step is easier to describe than it is to do.



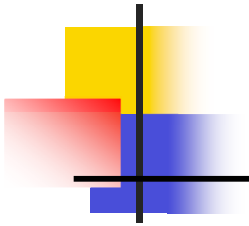
The Inventory Is Critical

- Identify repeated/similar processes that cross direct business lines
  - SOA is really a way to craft software based on **business process** oriented architecture
  - Identify initial data sets and applications
  - Consider both inside and outside the firewall
- Identify existing technology
  - Tools, messaging infrastructure, distributed applications or objects
- The more you know, the easier the transition



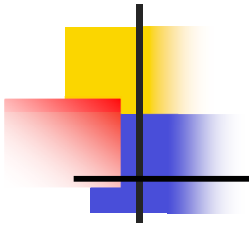
Success 2: Start Small

- Start with a limited pilot project
  - Based upon the repeated processes
  - Pick the right project team
  - Pick the right tools
    - Don't forget repository, directory/registry, lookup tools, message infrastructure, **standards**, …
  - Map to business process/service
    - Major anti-pattern is to map to either an application or an object layer
  - To Web service or not…???
    - Small packet messages or not time-critical: maybe WS



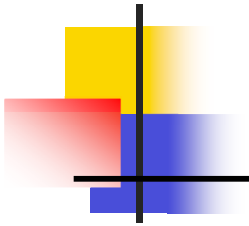
Considerations

- Use customers' key scenarios to help prioritize service implementation order
  - Focus on services that support the greatest number of scenarios
  - Focus on services that provide the most impact
    - That's why we suggested evaluation of repeated business processes
- Identify and **standardize** on a common set of infrastructure services



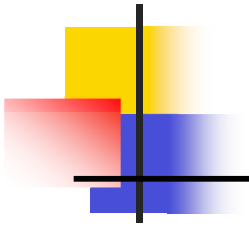
Success 3: Registry/Repository

- Repository == where, registry == what
  - Make sure they are standards based to assure interoperability
    - Test it, before committing money!!!
- Deploy service discovery facility (registry)
  - Helps prevent duplicate effort
  - Start as intranet site with manual discovery
    - The key is an enforced mechanism for discovery
  - As number of services increases commercial/open source tool that supports description and discovery (UDDI, DOAP, *etc*.)



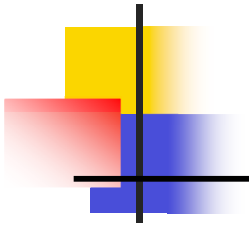
Registry Selection Is Critical

- Registry is more than a descriptive container
- Center for SOA governance!
  - Human responsibility chain, version, enterprise compliance requirements, and more…
- SOA is a new paradigm that maps code to business services, NOT an application. The registry is a critical piece of the puzzle and part of the new thought process.
  - Not link to, but FBE.



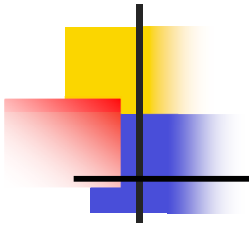
Success 4: Governance

- Success with SOA is inexorably linked to governance
- Governance is not just about code, it's also about communication and process.
  - Good governance is also about distributed responsibility to teams that have budget, authority, and capability to maintain and modify each service!
  - You can guess the anti-pattern



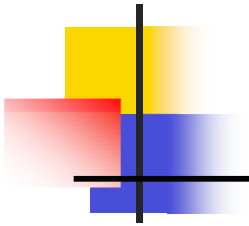
Governance

- Who, how, and what
  - Who is involved & what do they do?
    - Who has rights to make decisions?
    - … with appropriate entries in the registry
  - How do they communicate, make decisions, maintain oversight, assure regulatory compliance?
  - What happens: stuff doesn't work or breaks?
    - When do you "version"? Do you "version"?
- What impact will this have on communications, structure, and management?
  - Is enterprise infrastructure support sufficient?



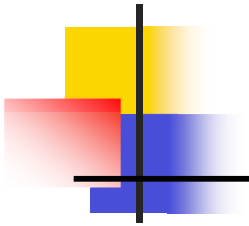
Success 5: Security

- It's Security Time – sooner not later!
  - Trust: zones, domains… within an enterprise
    - Or closely related enterprises
- It's too late to worry about security, if you only think of it in the context of external connections.
- Look at WS-Security sooner!
  - Also SAML, or WS-Trust, WS-SecureConversation, WS-SecurityPolicy
- Have people responsible for standards



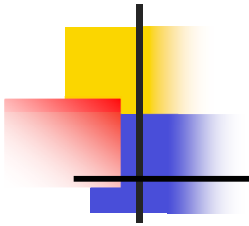
Security

- First thoughts about SOA are usually internal
  - What can we do to…???
- Quickly discover collaboration with partners is a natural evolution.
  - More than 40% of the time it ends up providing the most ROI
  - It's an Internet-based **connected** world!!!
- Perception: inside is easier to MMAC
  - Outside is usually when the first hard thoughts about security occur.



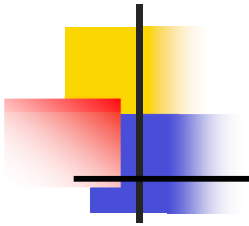
Security: Tail or Dog?

- What is your organization's position?
  - You connect to them…
  - They connect to you…
  - Mutual and balanced
- Don't assume!
  - If you're connecting to them, you're not going to be the one to decide how things work.
- Many security standards aren't fully baked
  - Have people responsible for standards



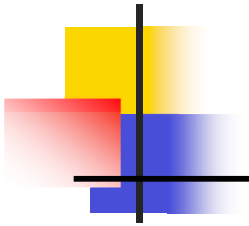
Success 6: Messaging

- SOA requires that services communicate and are triggered by messages.
- Initial messages may be pure SOAP-based (or some other direct mechanism)
  - Consider scalability
- Next critical decision: Message infrastructure
  - If you've already got one (*e.g.*, MQ Series), keep it.
    - Including Web services standards
  - Don't switch for the sake of the switch!
  - WS-ReliableMessaging



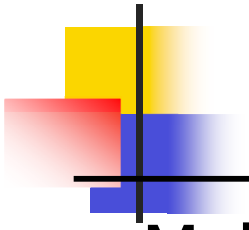
Success 7: Service = Resource

- No Ifs, ANDs or BUTs: **Services are network resources**
- There are tools to help monitor network health, network performance, network fail-over, network exceptions…
  - Substitute the word "service" for "network" and you'll get the idea
  - Instrumented with a package of services to monitor the health of your service resources.
- Service assets/resources have a life cycle



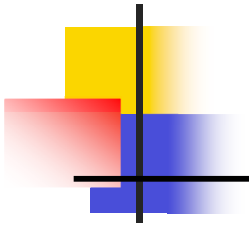
Service Management

- Standards coming
  - *e.g.*, WSDM, WS-Management, ITIL
- It's not a problem until it's a problem and then it might be too late.
  - With one or two services it's less of an issue
  - As the number of services increases and their interactions grow and…
- At what point did you need network management tools?
  - Similar concept, thought process, and planning



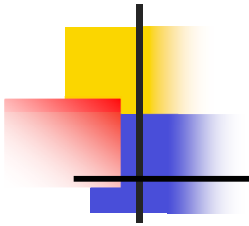
Success 8: Coordination

- Make things play nice. Pick your buzz word:
  - Coordination or Orchestration
- Either way it's about combining services
  - BPEL doesn't really account for human interaction ➔ BPEL4People
  - WS-Coordination
    - http://www-128.ibm.com/developerworks/library/specification/ws-tx/
- How does a complete business process happen?
  - If it's a single service (response sent to requestor), done
  - If the answer involves multiple services then it's time to think about orchestration (replyTo)



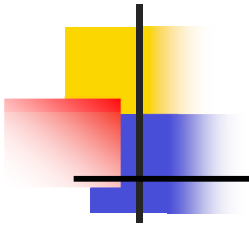
Orchestration *vs.* Choreography

- Choreography is an anti-pattern
  - If services know about each other then you've destroyed the notion of business process
- Orchestration leaves services independent
  - It's the independence that provide the most business flexibility
- Remember: Business Process-oriented architecture



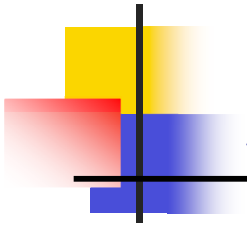
SOA Challenges

- Limited vision (usually from executive mgmt)
- Limited visibility (for value proposition)
- Limited skills (in-house)
  - IT staff
  - Business analysts or SMEs
- Limited understanding that SOA is BPOA
- Limited or no understanding of best practice
- Too much perceived complexity
  - Implementation of SOA
  - Existing processes/interaction are too complex to identify or separate – so never get started.



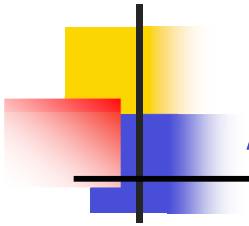
SOA/BPOA Challenges

- Make no mistake: SOA/BPOA requires new testing tools and new processes!
- It's a new paradigm for development
  - Business process is everything!!!
- What doesn't work:
  - Think in terms of applications…
  - Think in terms of distributed objects…
  - Everything isn't SOA, it is BP



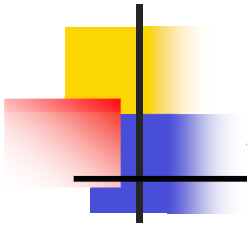
Anti-pattern Causes

- Unclear understanding of what SOA really is
  - Confusing SOA with Web Services
  - Confusing SOA with SaaS
  - Confusing SOA with Client-Server
  - SOA isn't about re-useable components, it is about reusable business services.
  - SOA is about architecture, not implementation



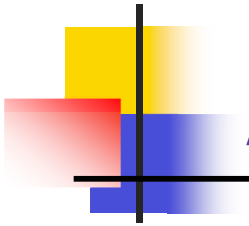
Anti-pattern Incorrect Assumptions

- You don't do SOA to get re-use. You get re-use because you're doing SOA properly
  - Does not mean that all services are or should be re-useable – agility (constant or frequent requirements changing) usually overrules re-use.
  - Composition might look good on paper (for components) but could limit usability.
- SOA is "designed" for business solutions
  - Not games, modeling, graphics, *etc*.



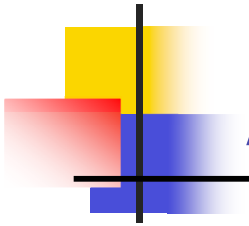
Anti-patterns

- Adoption
  - Bandwagon, everyone's doing it
  - What's new (lack of understanding what SOA is and how it's different than past development paradigms)
  - Big bang
- Identification & design
  - WS == SOA
  - Silos-R-Us (same service identified by different groups with different names)
  - Messed up Registry (duplicate, unclear ownership) –> governance nightmare



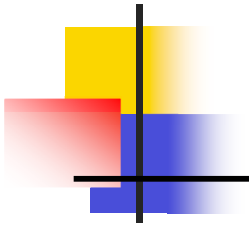
Anti-patterns

- Implementation
  - Chatty services
  - 1:1 connected services
  - Duplicated services (see Silos-R-Us)
  - Monolithic services (akin to monolithic objects)
  - Everything is a service (not really, it's about business process)
  - SOA development == application development
  - IT driven SOA



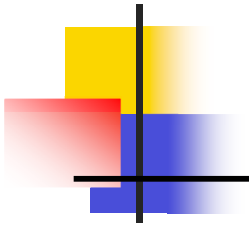
Anti-patterns

- Forcing top-down or bottom-up architecture
  - Top-down may only identify services that aren't cost effective to build
  - Bottom-up alone usually produces redundant and/or ungoverned services
  - Best practice, iterate between the two approaches



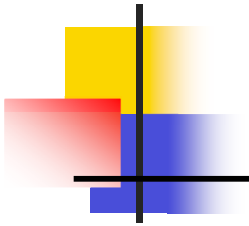
Myths and Misconceptions

- SOA is buggy – try SOA 2.0
  - SOA is not buggy nor is it client-server.
  - It's an **architecture**, not application
- SOA slow
  - SOA is an architecture.
  - Performance is a non-functional requirement that provides guidelines for implementation choices.
    - See chatty services anti-pattern



It's About the Business!!!

- Is the current state of (or plan for) SOA adoption described in clear, straight-forward terms?
- Have you investigated the benefits and advantages you can access right now with your existing integration infrastructure?
- It is possible to achieve greater business flexibility with greater adoption of Services Oriented (Business Process oriented) approaches?



Summary

- SOA is business process architecture
- Learn from the mistakes of others
  - Study the challenges and anti-pattern causes and then don't do that
- Study the success factors.
  - They're not a Chinese menu, there's no selection
    - Do them all!
    - Note: It's not about tools; it's about process!
- Develop a plan before you start
  - Ready, fire aim doesn't work

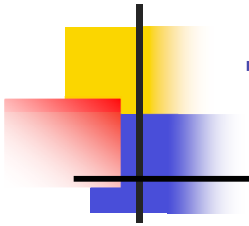
Questions ? ? ?

If you don't ask, who will?

There aren't any dumb questions.

If not now, when?

The only dumb question is the one not asked!



Thank You

For more information:

David Moskowitz

Productivity Solutions, Inc.

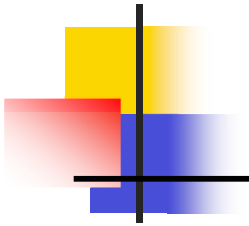
147 Ashland Avenue

Bala Cynwyd, PA 19004

+1-610-726-9925

davidm2@usa.net

SkypeID: davidmosk



Related Sessions at CSS 2006

- Service Data Objects
  - Steve Brodsky
- Open Source SCA – The Apache Tuscany Project
  - Jean-Sebastien Delfino
- Service Component Architecture
  - Jean-Sebastien Delfino
- SOA: You Heard the Hype – Now See How It Is Really Done!
  - Denise Hatzidakis
- The Evolution of Service-Oriented Development
  - Bill Jones
- Next Generation SOA Development
  - Bill Jones

Colorado Software Summit