SOA Approaches

- **Bottom-up**
  - Services are implemented “as needed” basis.
  - Implementations encapsulate application logic to serve the immediate requirements of the particular solution
  - Typical "Web Services" approach for many organizations
  - In many cases SOA as whole is seen as a “bottom-up” approach

- **No really strategy at all?**
  - Missing overall goals and business-alignment will probably lead to the creation of non-standardized service with poor reusability
  - Typically the scope is on "Project SOA"
SOA Approaches

Top-down
- An "analysis-first" approach
- Closely tied to organizations business logic and model
- Complemented with Service-oriented business alignment and process redesign.

Result: A high quality service-oriented architecture with good business-alignment – scope is in “Enterprise SOA”
- Requires time and money – “a long and hard road”

---

SOA Approaches

Agile or “meet-in-the-middle”
- Find an acceptable balance between incorporating service-oriented approach into business and process analysis environments without overly long "up-front" analysis phase.
- Business- and service-level analysis are done concurrently

Aims to fulfill both short- and long-term goals
- Increases resources needed both for service design and maintenance
SOA Approaches

- **Document engineering**
  - Synthesis of complementary ideas from information and system analysis, business process analysis and business informatics.
  - Documents describe the interfaces between business processes and tasks; Well-defined interfaces maintain clean and stable relationships between different actors (services).
  - Results precise specifications or models for the information needed to realize business processes \( \rightarrow \) service interfaces
  - New interesting approach; still unproven

SOA Lifecycle

- Development of Service-oriented, composite applications is on the surface quite similar to more traditional development approaches, but on deeper levels there are some differences which come from the service-oriented nature of SOA.
- Phases of SOA Lifecycle
  - Service-oriented analysis and design
  - Service development
  - Service testing
  - Service deployment
  - Service administration
Service-Oriented Analysis

- Determines the potential scope of SOA
  - Define clear scope: Project SOA or Enterprise SOA
  - What services are needed and what logic they encapsulate?
  - Identify current systems with "service potential" (assets)
  - Define and group service candidates (logical contexts)
  - Identify encapsulated logic with reuse potential (business view)
  - Identify issues with service autonomy and data ownership
  - ... 

- Critical decisions that affect the entire scope of SOA effort

Identifying and Modeling Services

- Identifying the right services is a critical step
  - Find the right services with right granularity and reuse potential, balancing both short-term and long-term goals and benefits.
  - The focus should be in business-level services
    - Any organization or process can be decomposed into a collection of business services (whether or not there exists a BPM) which represents a logical unit of work.
    - Requires thorough knowledge of underlying business logic
    - Enable composition and orchestration and promote reuse.
    - Key to enterprise level adoption of service-orientation and SOA.
  - Note: There is no universally accepted guidelines or methodologies for service identification and modeling.
Identifying and Modeling Services

- Some considerations for service modeling
  - Task-centric and Entity-centric services
    - Task-centric: accommodate a specific business process
    - Entity-centric: encapsulate a specific business entity
  - Potential for intra-process, cross-process, cross-application reuse
  - Be aware of process-specific dependencies
  - Use SOA layers as a tool for mapping services (Tue.)
  - Look forward and speculate on future reuse scenarios

- Remember general SOA Principles (Mon.)

Service-Oriented Design

- Transform logical service candidates into physical services
  - Define service interface designs
  - Separate application frontend (visual level) from logic
  - Identify potential service compositions
  - Define (basic) orchestrations for process-centric services
Service Development

- Realizing the designed services into software artifacts
  - Service development (build new services)
  - Service enablement (legacy wrapping)
  - Actual process, activities and tasks depend heavily on the selected standards, technologies, tools and existing applications.
- Some considerations
  - Remember general SOA Principles (Mon.)
  - Services are meant to be reused in contexts not known at development time, so plan accordingly.
  - Vertical slicing and "thin-thread" (end-to-end) model
  - Special care should be given to stability, security and scalability.
  - Try to stay as Stateless as possible.

Service Enablement

- Service enablement is the process that creates services form existing (legacy) applications.
Service Development

- Generate
- Test
- Service
- Implement

- Generate
- Test
- Client
- Implement
- Service
- Test
- Deploy
- Test
- Deploy
- Application
- Frontend
- Services

Service Testing

- Testing is a complex process for SOA applications
  - The scope: from single services to executing an end-to-end system test in complex application environment.
  - In an enterprise SOA environment many systems used by single composite application are external and in many cases "live"
  - Dilemma: Sandbox or live environment ?
    - Build sandbox duplicating # of existing systems and services or build stubs / simulations for those ?
    - Test in "live" environment with real cases and real data ?
  - Nature of services: will be used in context not known at dev. time
    - How to test that ? With what requirements ?
  - A traditional QA cycle, with unit, integration, systems and acceptance tests, doesn't suit well for SOA projects
Service Deployment

- Service deployment is specific to the technology platform
- Some generic topics
  - How services will be distributed?
  - Is infrastructure capacity capable of handling services?
  - How new services / versions will affect other services / applications?
  - Version control and service configuration management?
  - How encapsulated legacy systems will be monitored?
  - Security issues: authorization, authentication, ...
  - Future scalability issues?

Service Management

- Mostly standard application / capacity management which arise in complex distributed environments.
- Some generic topics
  - Service monitoring; availability, usage, capacity, security, ...
  - Service accounting and billing
  - Service brokering and directory administration
  - Message logging, storing, ...
  - Performance issues; bottleneck detection, reporting, ...
  - Service Level Agreements (SLA)
Service-Oriented Project Management

Different widely used SW project management methodologies have their strengths and weaknesses

- Suitability depends heavily on context;
  - R&D style product development vs. custom enterprise application

Some points to consider

- The nature of Enterprise SOA is highly dynamic, so chosen methodology should support iterative, agile development
- Strategic importance of SOA means usually large number of people involved in any development effort
- Be prepared to incorporate service-oriented aspects to any chosen methodology

Some points to consider (cont.)

- Business projects vs. IT projects
  - Any SOA project must be closely aligned with current business development – SOA is not IT issue!
- Program vs. Project management
  - For SOA to be successful, it must be coordinated on higher level than just one single (system or application oriented) project
- Business Services vs. SOA infrastructure
  - Two architectural levels; business services and service infra
Service-Oriented Management

Questions?