SOA ARCHITECT SUMMIT

Turn your ideas into practical solutions.

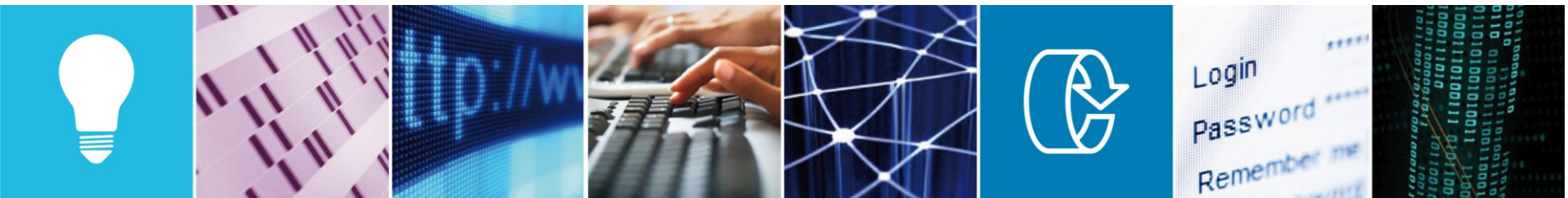
IBM

Recycling SOA Services & Artefacts

9th September 2009

Reviews reuse strategies and opportunities; establishes a rational basis for reuse decisions; recommends an approach and critical factors.

Richard Whyte ***CEng,FIET, CITP,FBCS***



Reuse, Recycle: building from existing parts

- **Strategies and Opportunities**
  - Types of reuse
  - Barriers to reuse
  - Anti-patterns
- **A rational basis for recycling**
  - Organisation
  - Building from existing parts
  - Critical success factors
- **Recommendations**

An error cannot be believed sincerely enough to make it a truth.

(Robert Green Ingersoll 1881, foremost orator and political speechmaker of late 19th century America)

Reuse strategies and opportunities

- **Reuse is a tactic to achieve an objective**
  - Technical: *Consistency, Quality, Simplicity*
  - Project: *Risk, Quality, Skills, Duration*
  - Business: *Improve time to value, Reduce cost*

- **It can lead to**
  - Poor quality “patched together” solutions
  - Components & code that don't contribute to the solution
  - High dependency environments
  - Expensive, delayed, complex “future proof” solutions

Types of reuse

- **Reuse is referencing an asset or taking a copy and making changes**
  - "By reference" (Runtime)
    - Changes affect everyone
    - Challenges for impact analysis, testing, upgrades, etc
    - Creates dependencies and version control challenges
  - "By value"
    - Creates a plethora of components with subtle differences
    - Challenges for impact analysis, testing, upgrades, etc

- **A common target architecture reduces change to assets**
  - Alignment of assumptions reduces mismatch
  - Common performance and availability standards

Barriers to reuse

Operational	Leads to	Causing
Inefficient asset catalogue **Charge-back and rewards** Culture Process Information Organisation & Investment	Finding / understanding Rewarding individuals / projects Value invention over reuse Inefficient harvesting, **indexing**, **documentation**, testing What is available, how is it used. What SLA	Difficult to add components to register: Duplication Complexity and adoption cost Ad-hoc approach to reuse Added administrative costs Penalty for success! Your load on my server
Technical	**Leads to**	**Causing**
Design **Lack of encapsulation** **Different assumptions** Inconsistent Standards Mismatch of Non-functional	Inconsistent architecture containing exceptions High cohesion/dependencies Incomplete implementation significant mediation Security, Latency, Availability, Accuracy, Retirement date	Limited “by reference” reuse opportunity Incomplete interfaces / Key data not available Additional “hidden” costs of reuse **High maintenance**

Service Oriented Architecture delivers

- **Isolation**
  - Reduce impact of change.
  - Avoid “long range” cohesion across the enterprise
  - Provided by published interfaces and routing through a mediator
- **Encapsulation**
  - Reduce impact of build decisions, different technologies and standards
  - Principle of independent implementation
- **Specialisation**
  - Specialist components (Services) with “simple” interfaces are more likely to be reused
  - System “stuff” provided by mediation: audit, security, routing, data management
- **Billing and metrics collection** (Not always included)
  - Simple to add to SOA compared to other styles
  - Provides a way to fund common components and improve shared infrastructure

SOA combined with strong governance provides the basis for removing barriers and fostering a reuse culture

Reuse can be an excuse for excess

- **Building components for all eventualities**
  - Creating many layers “just in case”
  - Adding fields that “might be useful”
  - Insisting on a technology because “it is more useful in the future”
  - ***Fine grained interfaces !***

- **Spending time and effort building for “undefined” reuse**
  - Undermines the current project and reduces the benefits
  - The architecture and governance MUST enforce characteristics for reuse
  - Humans have a poor record at predicting the future
  - Additional features are rarely tested so cannot be relied on

- **So**
  - Stick to the principles; Isolation, Specialisation, Encapsulation to allow change
  - Build only what is needed for today
  - Remember the objectives: Cost, Time,

Conclusion: SOA creates opportunity to reuse

- Concepts, principles, **experience**, and skills
- Documents, designs, standards, plans, templates
- Test harnesses, test data, and runtime environments
- Business continuity, transaction control, availability

- (Runtime) Services
- (Build time) code libraries, templates, tests, concepts

> **But there are pitfalls ☺**

A rational reuse strategy is based on

- **Clear Objectives**
  - What is to be achieved through reuse
  - How it will be measured in terms of business benefit
- **Consistent Architecture**
  - What characteristics are necessary from the architecture
  - How will harvesting operate
  - Are there clear and appropriate rewards
  - A funding model for shared components
- **Strong Governance**
  - Reuse does not trump structure, requirements, isolation, or specialisation
  - How are decisions made
- **Supportive Culture**
  - Design against known artefacts
  - Clear ownership and open discussion
  - Peer reviewed intentions

Recommendations: Strategy

- **Use SOA to provide a consistent design canvas**
  - A consistent bar for assumptions
  - Disaster recovery, scalability, interface structure, unique keys
- **Centrally control and manage reuse**
  - Products - RUNTIME services and documents provided with support
  - Assemblies - BUILDTIME documents and libraries provided by reference
  - Components - BUILDTIME code and “as-is” artefacts brovided by value
- **Decide how to meaningfully measure success**
- **Change the organisation**
  - Reward reuse and creation of reusable assets
  - Create an owner responsible for support, maintenance and modification of reusable assets
    - Promotes reuse and has an overview
- **Clear and strong governance**

Recommendations: Organisation

- **Central control**
  - Asset repository?
  - Website and support group?
- **Measuring Success**
  - Number of consumers? Website audience, Throughput
  - “Usefulness”: Developer satisfaction
  - Business measures
    - Time to value
    - Cost profile and benchmarking

Recommendation: Governance

- **Standards**
  - A small set of “Mandatory” characteristics
  - Not best practice or style guides
  - Only naming where names are critical to the design
    - Example: All communication through the ESB
- **Best practice**
  - We (the establishment) prefer this approach but its not mandatory
    - Example: All error logs should be circular
- **Style guides**
  - Naming and coding layout preferred styles etc.
    - Example: Indentation

Design Tactics: Isolation, specialisation, encapsulation

- **Architecture and principles foster reuse**
  - Re-factoring is always possible
  - Reuse does not drag a lot of baggage
  - Interfaces can be extended to accommodate new requirements

  – Service orientation
    - "Service" is the unit of reuse, version control, regression test, and documentation
    - "declarative" Interfaces

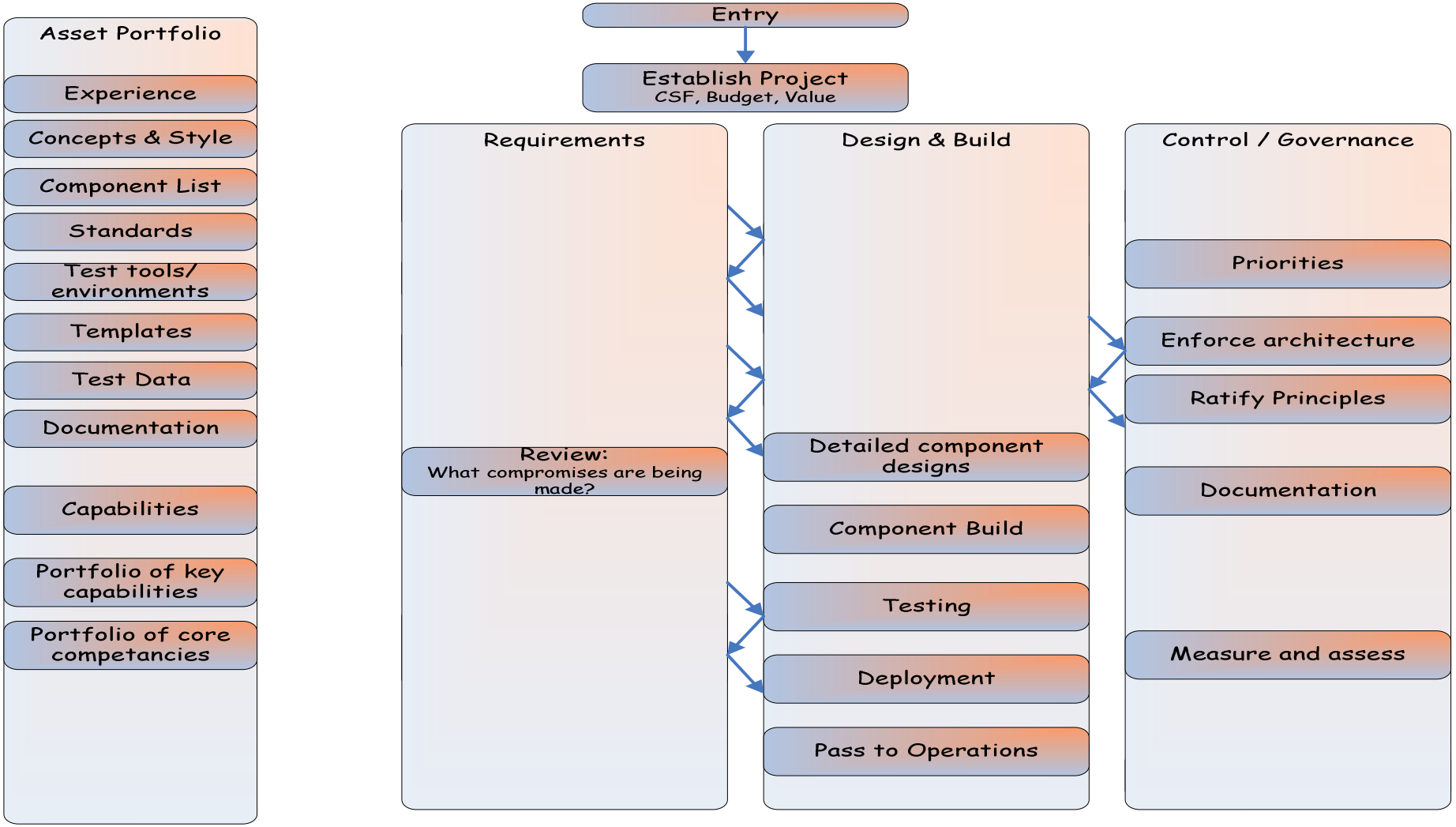
- **Design for the problem not the future**
  – Follow the rules but don't add function for possible future use
  – Responsibility of the architecture to foster reuse; not the programmer
  – Test all functions available. The fewer there are the more cost effective this is.

- **Design interfaces to hide complexity and be extensible**
  – Feature switches
  – Fine grained interfaces leak implementation and complexity

Critical Success Factors

- **Reuse is a tactic to achieve an objective**
  - Be clear about objectives
  - Don't spend time building for "undefined" reuse
- **Be prepared to pay for reuse**
  - Reusable assets need an owner (Maintain ownership?)
  - Keep a core team together; experience is an important reuse.
  - Reuse includes concepts, plans, documentation, etc.
  - Scaling and managing the cost of reuse, maintenance, upgrades
- **Design first then refine for reuse.**
  - Compromising SOA principles reduces reuse opportunities
  - Reuse using reference and value appropriately

A methodical checklist (reusable)



Finally

Reuse without a plan,

on a zero budget,

with no rules,

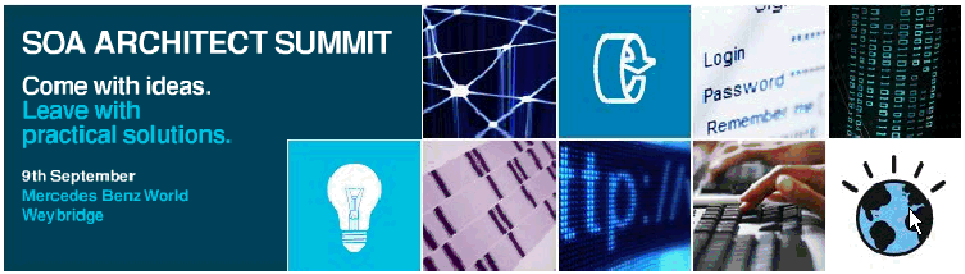
delivers to expectation

Reuse and recycling have very similar meanings.

This presentation introduces the idea of recycling to convey that parts can be wholly recycled, reused, or refined to make them useful.

SOA Architect Summit Next Steps

- **Summit Presentations at:**
  ibm.com/software/uk/itsolutions/soa/soa-summit



- **Building SOA Solutions**

– An IBM Proof of Technology

6th – 8th October in IBM Hursley

ibm.com/itsolutions/uk/soa/building-soa-solutions-pot/

- **SOA Forum**
  – please join & participate
  soaforumuk.com

- **European WebSphere Technical Conference, Hamburg, Germany**
  European WebSphere Technical Conference

Backup Notes: The cost of reuse

- Must adhere to standards and interoperability needs
- Must maintain backward compatibility
- Regression testing

- Management of repositories
- Maintenance and scaling of shared resources
- Organisation

Reuse artefacts by project phase

- Map reuse by project phase