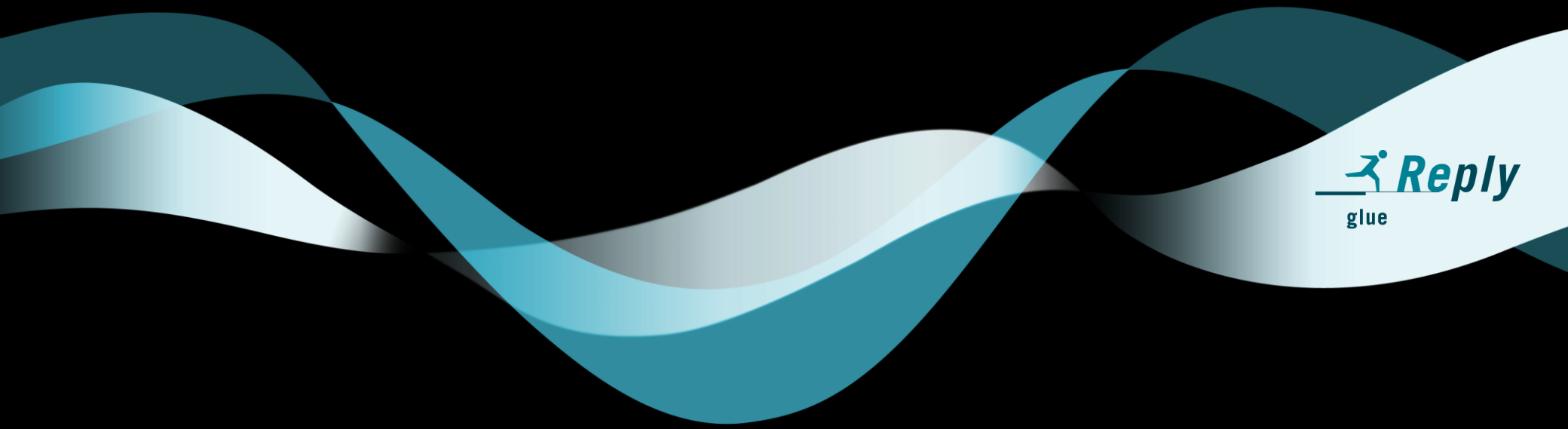
“SOA – everyone focuses on re-use
but nobody seems to worry about re-work!”

SOA Strategy



Reply
glue

Scope statement

- One of the key promises of SOA is re-use, and over time organisations are increasingly becoming successful in achieving that goal;

However…

- There are external factors that also have an equal bearing on the success of an SOA implementation.
- Factors such as changing standards and vendor technology lifecycles can impede the success of re-use alone in providing cost benefits.

Focus

- This session discusses change
  - Changing face of standards
  - Maturing of technologies
- How to minimise impact of change on design and implementation of SOA
  - Approaches
  - Techniques
  - Methods
- Focus is primarily around integration and implementation technologies

SOA technology problem

- You cannot rely on the technologies still being around in their current form in 5 years time
- *Solutions will reach an end-of-life*
- *If an implementation is tightly bound in to the technology it is going to require re-work*
- *The business case around re-use may no longer stack up*

Some examples…

- eGate

- Activesoft
- Activeworks/Broker

- InConcert

- Crossworlds/ICS
- Partner Agreement Manager
- WMQ Workflow

- Tuxedo
- eLink

- Business Connector

- InterConnect

Even standards are changing

- Supporting standards are also changing
  - Be it ws-* standards maturing
  - Best practice/proprietary standards
  - E.g. WSDL, BPEL, REST, ws-security, SCA…
- Messaging standards are also changing
  - Change in paradigm between versions of messaging standards
- Data & information standards evolving
  - Information exchange – OAGIS, ebXML, SWIFT, EDIFACT, etc.
  - Relationship with encryption standards
  - ISO standards alignment

Methods & approaches have changed too

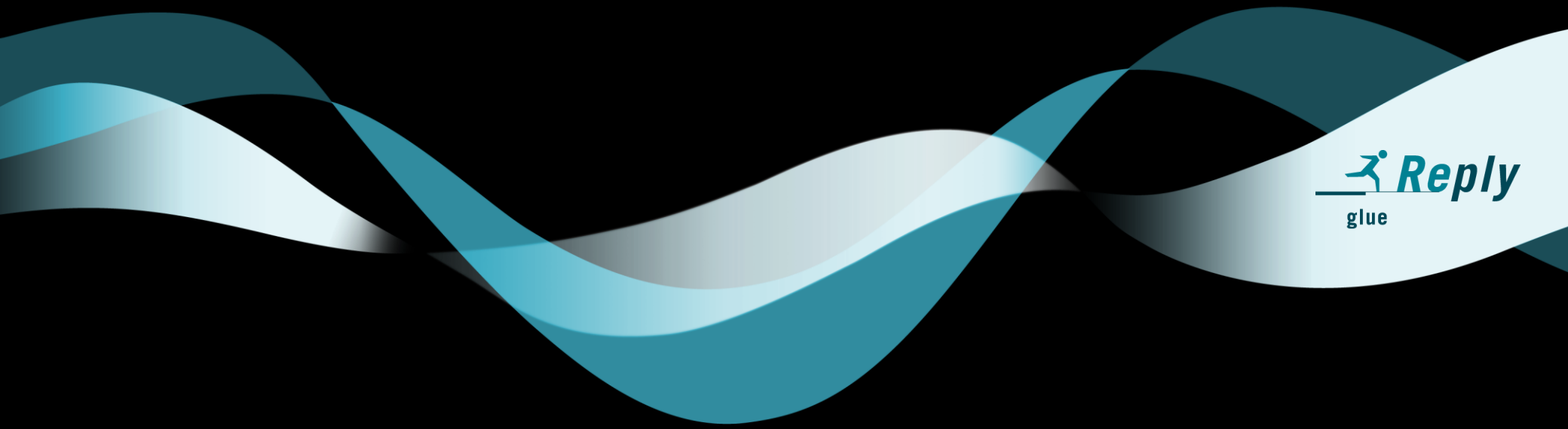
- SCRUM
- RAD
- ITIL
- COBIT
- TOGAF

Impact on business benefit

- Re-work in line with technology lifecycle
- Potential vendor technology lock-in or proprietary upgrade path
- Ongoing re-training to support technologies and standards
- Diminished ROI over time
- Not master of own business benefit destiny

“SOA – everyone focuses on re-use
but nobody seems to worry about re-work!”

Solutions



Reply
glue

Don’t just take the technology out of the box

- Prescribed vendor implementation MAY lead you down a road of vendor dependency…

- SOA technologies should be considered as assembly kits providing…

Components

Tools

- *What <u>matters</u> is how you use them…Define your own assembly instructions.*

Plan & architect first, implement later

- Plan the strategy & define the architecture of the solution domain first
- Define it at a logical level of abstraction
  - Preferably define a complete logical semantic layer
- Do not bind the architecture in to the way the physical tool would lead you to do things
- Define an architecture delivery roadmap so that the architecture can be implemented incrementally by projects and/or programmes

Avoid incorporating proprietary products & standards into your architecture solution

- Common mistakes in SOA implementations include:
  - Physical technology as logical solution
  - Coding by tool e.g. ESQL, Monk, etc.
  - Binding communication & transformation together
  - …or indeed binding transformation & routing
  - Embedding data access functions

Key goals

- Define services agnostic of their implementation
  - Provide logical view of services
    - Model the services to be implemented logically and map them against logical components implementing that service
  - Avoid defining a service specific to a particular technology

- Utilise open standards
  - Messaging & routing services
  - Data standards & semantics
  - Registry & repositories
  - Transformation & translation

  - They are often mature enough now to stay the same for an extended period in an organisation
    - Internally, it matters less if the latest version of a standard is being utilised

Consider interoperability between standards

- Define an interoperable set of standards for internal use
  - Communication
  - Messaging
  - Data
  - Access
  - Services
  - Processes

Do not confuse internal with B2B/B2C requirements

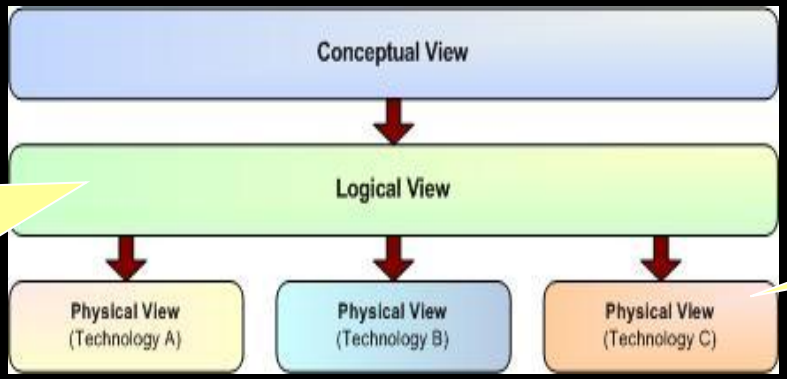
- Insulate internal integration from external change
  - Deal with external standards as a boundary issue
  - Do not attempt to embed what is done for partners for internal use
- What is required to do business with partners and consumers is not the same as what is required to operate internally
  - Organisations can equally create tight dependencies
  - External integration can become an unwanted legacy too
- Extended enterprise requires:
  - different processes
  - different business rules
  - different constraints and policies
  - standards specific to the organisational boundary

Break down the problem into its constituent parts

- Achieving business agility will be constrained by brittle & tightly coupled technology solutions
  - Not only should services be loosely coupled,
  - BUT architecture components should be loosely coupled also
- Don’t mix transformation with message sending… with translation… with …
- Equally ensure that the solutions on the service infrastructure are logically modelled
  - and not bound in to the physical implementation

Multi-vendor, multi-technology support

Be pragmatic, architect for a heterogeneous environment



Conceptual & Logical views provide a technology neutral definition of the To-Be SOA Architecture

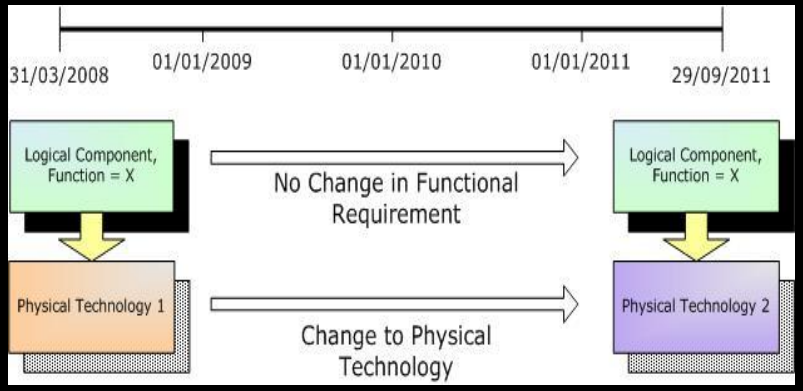
Physical view provides a detailed relationship between the Logical (technology neutral) view of the To-Be SOA Architecture and your chosen technology platform(s)

Roadmap compatibility and not just interoperability

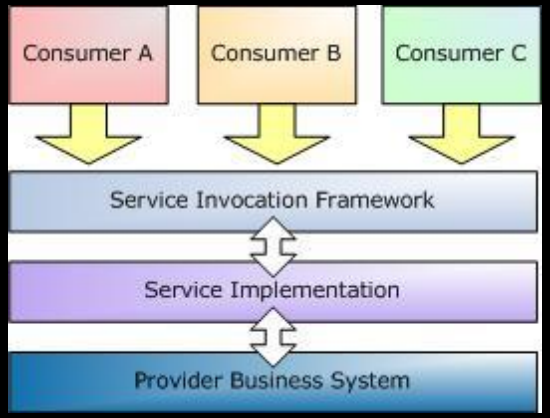
Do not be constrained by a particular vendor/ technology

Architect for end-of-life

You know it will happen so architect for it from the outset



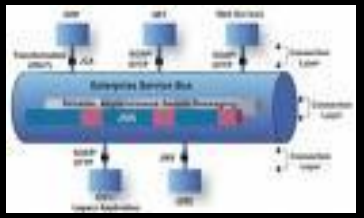
Architect a decoupling layer for end-of-life from the outset



…Then different paradigms can interoperate

Future

+



Now

+

Legacy

Thoughts to take away

- Architect the SOA infrastructure
  - At a logical level of abstraction
  - Break down the problem into clearly demarcated tasks
  - Utilise selected interoperable standards
  - Don’t bind in preferred physical technology specific implementation detail
- The way in which services are modelled should also be logical
  - Therefore the design itself is also portable
  - Provides abstraction of service from its implementation
  - Enables a migration path from legacy platforms

Contacts

www.replyltd.co.uk