

Service Oriented Architecture and Governance – 10 “Golden Rules”

August 2008

Ian Koenig

What is a Service Oriented Architecture?

- A Service oriented architecture (SOA) is essentially a collection of services that communicate with each other to provide a specific function. The communication can involve either simple data passing or it could involve two or more services coordinating an activity

But What is a Service?

- A Service is a coarse grained, self-contained, reusable set of software modules that perform a well defined set of functions through well defined interfaces. Services are independent of the applications using them, but inclusive of the computing platforms and all other resources necessary in order to operate, that are not part of other services.

Why invest in Service Oriented Architectures?

- Most Enterprises can distinguish between “Reusable infrastructure” and end-user applications (Solutions).
- By focusing on “costly” infrastructure (which may include systems, processes and even people) and encapsulating that infrastructure in such a way that it is reusable across all (or most) end-user applications, both cost and complexity can be reduced.
- In any one solution you should endeavor to fulfill at least 80% of the functionality from shared services in the SOA and no more than 20% from solution specific business logic
- “Services” are not the only way to get re-use. The most important distinguishing characteristic of a service is that it is encapsulated through well defined APIs **AND** there is a single operational instance that you scale up/out as you increase use. You never create a new instance. That would be a new service!

But what about the Dragons?

- Its easy to find references in the literature like “75% of all SOA initiatives will fail” – like from Gartner
- So, why should you not be concerned about investing in one?
- You should be cognizant of the main reasons that most fail – lack of governance and lack of organizational alignment.
- To work, the groups that own the services need to “service” their customers – the end-user application groups. All the incentives and governance needs to be in place to ensure that this happens and that the application groups don’t decide it would be less work and less angst to “just” do it themselves.



Many Services are Infrastructure

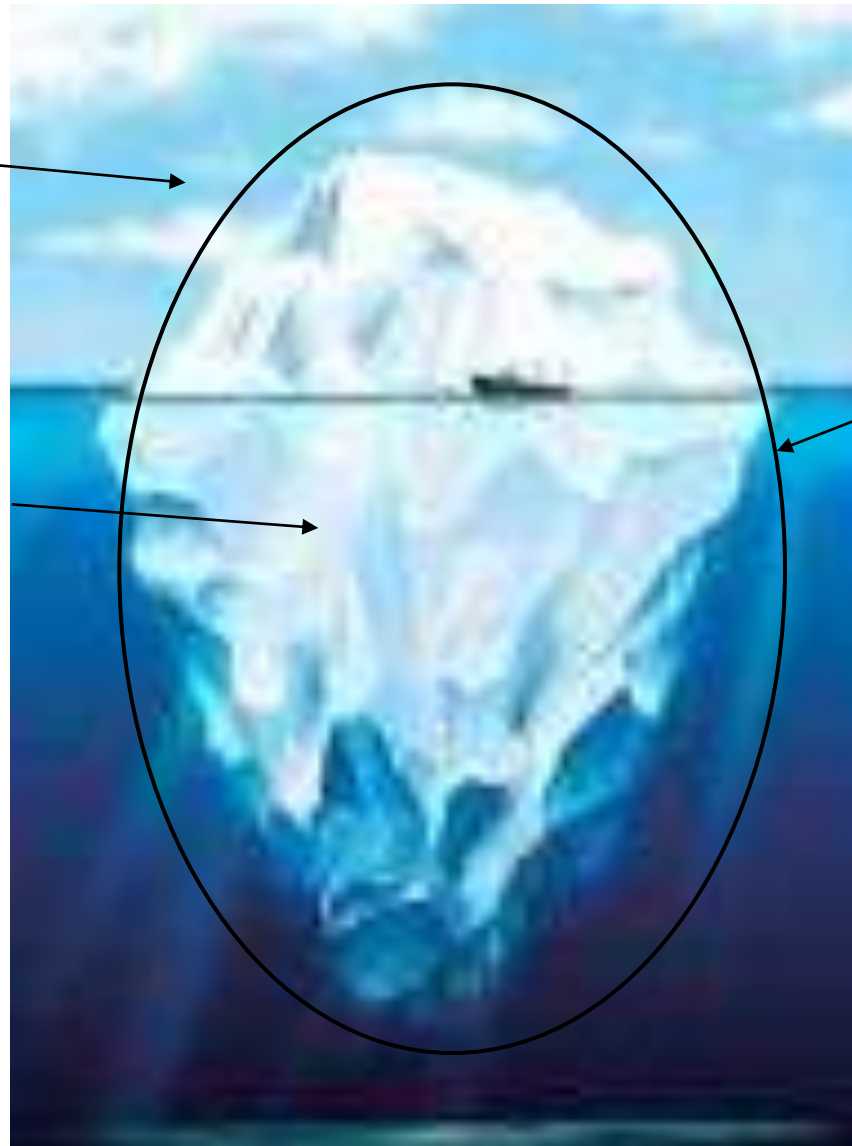


What does your infrastructure look like? Are you proud to expose it?

Service Web \neq Service

The Web
Service

The
Implementation



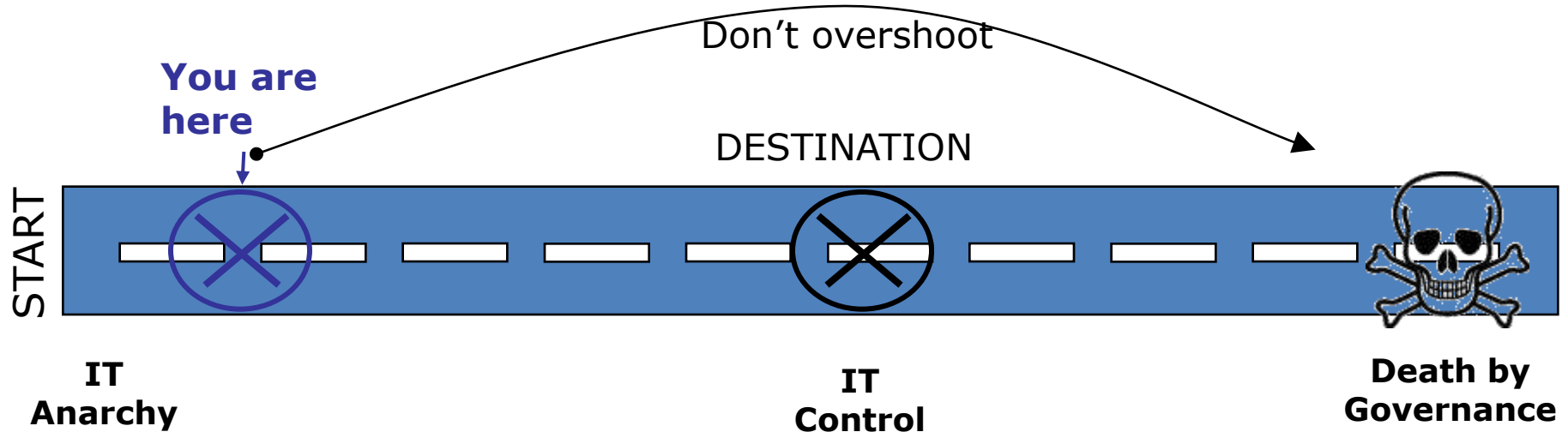
The Service

SOA – 10 “Golden” Rules

- 1. SOA requires governance:** The only way to control the complexity of an IT infrastructure built out of services is with a governance process based on "a well defined set of interface guidelines and policies"
- 2. Govern to the policies that matter:** It is important to focus on policies that are critical to making the business work. Its easy to come up with 5,000 really good ideas, but pick the 50 “rules that matter” and govern to those. Having too many policies is just as ineffective as having none (maybe worse).
- 3. People don't communicate:** Words aren't enough to communicate architecture precisely. You need diagrams and words. No matter how large your documents are, you will miss something and weightier is not necessarily better. I recommend starting with a subset of [UML](#) tools for class diagrams and sequence diagrams to clarify problems and review meetings to describe the architecture without weighty documents. But every enterprise needs to tune this to its own culture.
- 4. Make governance easy and do it early:** Since nobody likes being told after they have completed a project that they have done it wrong and need to do it over, integrate the policy management into the [Software Development Lifecycle \(SDLC\)](#) and automate as much as possible.
- 5. Reusability does not come cheap:** Despite all the hype around SOA and reuse, building reusable services is expensive. "In fact, our rough calculations are that it is approximately 2.5x as expensive to make a module reusable as not. Make sure you have 3 potential users of a service before you embark on the effort.
- 6. Interfaces are more important than implementation.** A proper interface encapsulates its implementation, so get the interfaces ([APIs](#)) right first. Start with versioning policy and loose coupling ; then extend to items like parameter use and Data model design”.
- 7. Integration is more common than "green field:** While there may be so-called "green field" architectures out there , I've never seen one and don't expect to. Many companies grow through acquisitions and accrete heterogeneous systems. SOA (through encapsulation and ‘separation of concerns’) is a good way to deal with this reality.
- 8. Identify ownership for every service:** Every service in an SOA should have an owner. There are owners at two levels. Business owners are responsible for the business aspects of the service, including the cost of running it and its value proposition. Infrastructure owners are responsible for maintaining the [service level agreements \(SLAs\)](#), and making sure customers of the service are satisfied .
- 9. Be pragmatic:** There are no perfect SOAs. There will inevitably be "deviations" from SOA best practices to meet the immediate demands of the business (your customers). Track deviations through the governance process and manage their eventual elimination “in the fullness of time”.
- 10. It's all about governance:** SOA requires on-going governance to measure how far you are from your goals and whether you are converging or diverging. Through on-going governance architects learn where the SOA needs to be enhanced and what new development is needed to solve problems in the ever changing IT landscape.

Rule #1: SOA Requires Governance

Governance¹ is a **(1) Process** which defines a set of **(2)Policies** and produces a set of **(3) Metrics** so that an **(4) Organization** can manage the eventual convergence of the technology with the architecture.



SOA by its nature enables a federated development process with individual groups producing individual services relatively independently. Only via a well defined set of interface guidelines, policies and a Governance process can you keep complexity under control.

1: Governance definition from [Burton Group](#) Research Analyst [Anne Thomas Maines](#)

Rule #2: Govern to the Policies That Matter

- Having too many policies is just as ineffective as having none (maybe worse).
- Choose policies that matter. The criteria we used was that if you could define the pain that either the business or the customer would feel by breaking the rule, it definitely mattered.
- It was a long and hard process differentiating the 5,000 really good ideas from the 50 policies that mattered. And the fact that we chose not to govern many of those good ideas does not make them any less worthy of following. Don't try to boil the ocean.
- The Policies we used were either derived from core principles or from standard patterns (like the content distribution pattern , on slide 10)

Rule#3: People Don't Communicate

- When smart people disagree on a solution, they almost always disagree on the problem they are trying to solve, not the solution they are arguing about. Unfortunately they usually skip the part where they each clearly communicate the problem domain.
- Probably the best vehicle for communicating architecture is through diagrams.
- The standard we chose was a subset of UML v2.0 (primarily class diagrams and sequence diagrams).
- You don't have to pick this standard. Just pick one and be consistent.
- Lots of people don't want to go through the effort of drawing what they are doing, but diagrams are much more precise than words and its amazing how much people learn what they "don't know" when they have to write it down in the form of a diagram.

Rule 4: Make Governance Easy and Do it Early.

- 60% of people will do the easy thing regardless of whether it's the right thing to do
- 40% of people will do the right thing regardless of whether it's the easy thing to do
- If the right thing to do is also the easy thing to do, then you get almost everybody.
- Nobody like to be told everything they did wrong at the end when its too late to fix. *"Them's fightin' words"*
- We have implemented automated governance wherever we can. We use Weblayers Center for SOA Web service governance, Black Duck ProtexIP for Open Source monitoring and Fortify for Information security "code scanning"
- We integrate automated policy management into the SDLC at software check-in and into the automated software build process. This is right and easy to do.

Rule#5: Reusability Does Not Come Cheap

- Software “re-use” has been the goal since the words *object –oriented* first graced the pages of text books in the 1960’s
- But re-use is expensive. In fact our rough calculations are that it is approximately 2.5x as expensive to make a module re-usable as not.
- This additional cost is not from the way you write the software as much as the way you deploy the software and the support and management infrastructure you place around it..
- Because if you build something to be re-usable, then you will have customers. Customers need support. Support costs.
- As a user of a “service”, your decision will be whether the cost of learning and the risk of dependency is outweighed by the value of the service
- **Therefore reusable services should be: Coarse Grained, costly to rebuild and likely to have 3 or more users.**

Rule #6 : Interfaces are More Important Than Implementation

- You can hide a lot of sins behind a well designed interface. Get the interfaces (APIs) right first.
- Interfaces must enable loose coupling
- Interfaces must adhere to the rules of proper versioning (or they would not enable loose coupling).
- We precisely defined the rules for versioning and loose coupling in our SOA.
- Defining a proper interface encapsulates an implementation such that it can be enhanced / improved / modified / optimized with minimal risk of impact to calling applications.

Rule #7 : Integration is More Common Than “a Green Field”

- There may be “green field” architectures out there, but I don’t expect to ever see one (much less work on one).
- A Service Oriented Architecture (SOA) approach (with governance) adapts very well to the more common acquire, integrate, devour and synergize style of architecture that we have become used to in Financial Services.
- The exact same Policies we use to evaluate our own architecture are used to assess part of the technology integration cost of an acquisition target .
- In fact, an SOA with policy-based governance should adapt equally well to both a green-field architecture as well as an integration architecture.

Rule #8: Identify an Owner For Each Service

- If you cannot identify an owner for each and every service in your Service Oriented architecture, then you are not quite there yet.
- Ownership can be defined in a number of dimensions:
 - Business Owner – Someone who is ultimately responsible for the service as if it was a business in its own right, the cost of running the service and the value proposition
 - Infrastructure Owner – Somebody who is ultimately responsible for the service from an availability and SLA point of view and who is responsible for the satisfaction of the service's users
 - Service Owner – Business Owner + Infrastructure Owner

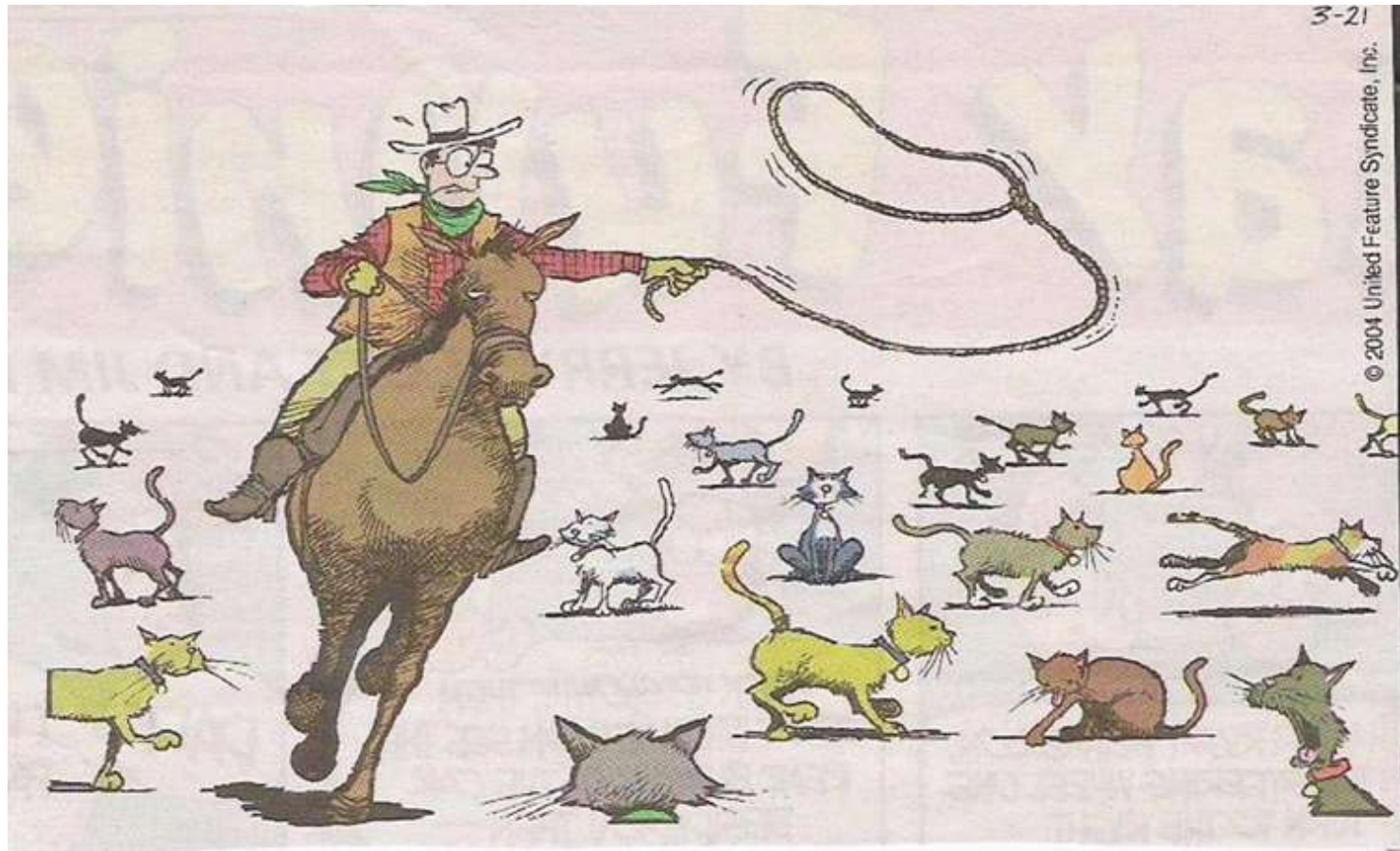
Rule #9 : Be Pragmatic

- The perfect Architecture, no matter how perfectly defined will never be achieved.
- There will always be reasons to deviate from “the straight and narrow” to meet business objectives.
- A well defined architecture and governance process tracks deviations and manages eventual convergence to the architecture over time.
- Each deviation decision is essentially a risk/reward analysis both in the short term and in the medium to long term.

Rule #10: Its All About Governance

- Once you've defined your architecture, the governance process is what allows you to measure how far you are from goals and whether you are converging or diverging.
- The governance process is also a great way to learn where the architecture needs to be enhanced, where new solutions are required for common problems and where new standards need to be defined.
- Enterprise Architecture and Governance provides: Efficiency, Agility and improves quality in complex technology environments.

Is This What Governance Feels Like To You?



"The more constraints one imposes, the more one frees oneself of the chains that shackle the spirit... the arbitrariness of the constraint only serves to obtain precision of execution."

-Igor Stravinsky

This presentation is an update / adaptation from the original:

“Establishing a SOA Center of Excellence:

Ten Valuable Lessons Learned”

which was presented at an SOA Forum session in March 2008, hosted by Greg Bjork, CEO of Weblayers, Inc.

At that time, I held the position of Chief Architect, Thomson Financial, which is now the Markets division of Thomson Reuters