

# The SOA Magazine

## Feature Article



### Project-Oriented SOA

by Leo Shuster

Published: August 13, 2008 (SOA Magazine Issue XXI: August 2008)

[Download this article as a PDF document.](#)

*Abstract: Projects are the lifeblood of an IT department. Almost everything in IT is measured through a project lens. SOA, due to its global-centric nature, is often viewed as incompatible with project-based software delivery lifecycles. Thus, most companies find themselves with the dilemma of how to effectively advance an SOA initiatives and continue to deliver projects at the same time.*

*The solution is to combine service lifecycle management, architecture, SOA governance, funding, and SOA metrics into a single comprehensive program. The ultimate goal is to ensure that through addressing project needs services are being effectively designed and implemented and that leverage takes place and is verifiable and that the overall SOA program objectives are being achieved. This article introduces an effective technique for moving your SOA program forward through an incremental, project-based approach.*

### Introduction

Everyone, from the CEO to the developer, understands the benefits of SOA and why it should be used. However, many companies still struggle with questions on how to correctly start, shape, and advance an SOA program. Even with careful and expert guidance, SOA initiatives face mounting challenges. The most critical barrier to SOA success is the very basic unit of IT operations – a project. Projects are the oldest and most widely accepted way to deliver work in an IT organization. Projects are time bound and oriented towards delivering specific outcomes for limited audiences. SOA initiatives span multiple groups and organizations and are geared towards addressing the broad needs of leveraging existing assets or creating new reusable assets. Thus, project-based work is largely considered incompatible with SOA.

Most companies that have embarked on the SOA journey find themselves in the unenviable position of trying to reconcile tactical project work with the strategic SOA initiatives. Project work requires delivery of custom applications or third party packages, while SOA's goal is to establish a base of reusable services. Projects only care about their requirements. Shared services must take the requirements across multiple projects into account to be truly reusable. Project's funding comes from the Line of Business (LoB) that is supposed to benefit from it. Since SOA efforts can span multiple LoB's, a single organization may not be the primary funding source. Merging these diametrically opposing views into a comprehensive approach has proved difficult for most IT shops.

The methodology outlined in this article bridges the gap between project work and SOA. It introduces a set of techniques that not only allow the projects to achieve their goals but also promote the creation and reuse of shared services. Additionally, it addresses the funding, reward, and enforcement issues that are necessary to achieve both project and SOA objectives.

### The Service Ownership Problem

One of the biggest political obstacles facing SOA in any organization is service ownership. Since project teams drive

the delivery of custom code or integrations, they consider that it is their domain and responsibility to build services that address project needs. Outside teams are viewed with distrust, even if they work alongside the project team (For example, code created by others is often dismissed or disregarded). Project teams consider themselves the experts on the subject area covered by the project requirements while dismissing the knowledge that exists outside of the team.

Ownership can be a touchy subject. Many project teams and IT managers subscribe to the silo mentality where they consider the whole stack – from the UI down to the data sources – as their property. Thus, they consider any services, shared or not, that address project or application needs as part of the whole stack. Any discussion or initiative that can be construed as an infringement on their territory can trigger irrational behavior or illogical conversations. Unfortunately, the reality of IT is often such that empire building and territorialism are viewed by the middle management as the best way towards success.

In order to become successful with SOA, IT organizations must break these silos. Shared services that can be consumed by a number of projects must be owned and managed separately. Services must be considered independent software products that have their own lifecycles different from those of projects or applications they serve. Their code needs to be stored separately from other code. They must have their own test cases and test suites. Shared services should reside on a dedicated, independently scalable infrastructure to ensure appropriate levels of responsiveness, scalability, and performance. Ideally, even the data accessed by the shared services must be enterprise caliber and reside on an enterprise scalable infrastructure. Figure 1 depicts how an ideal SOA infrastructure should look following these guidelines.

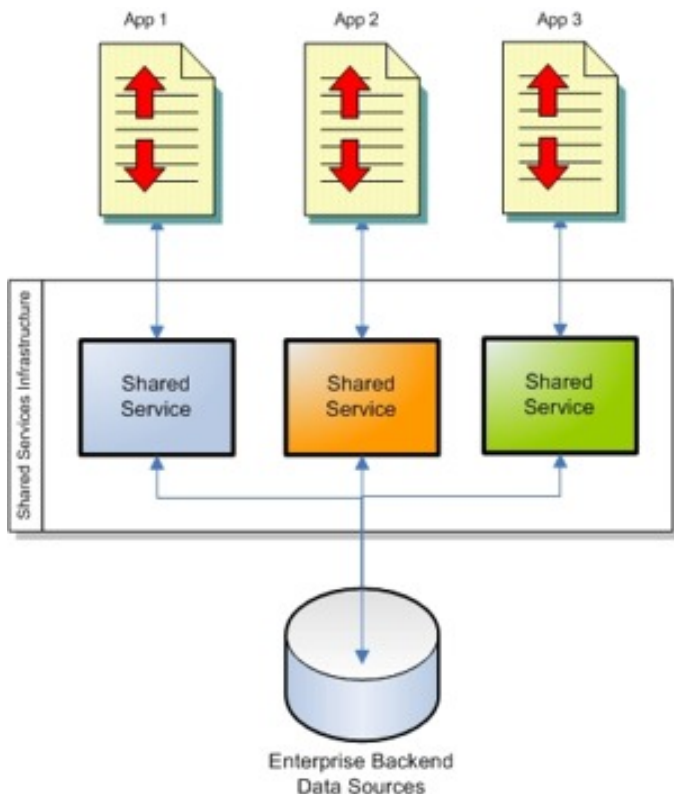


Figure 1: An “ideal” SOA infrastructure environment.

One of the critical reasons for breaking down silos and managing services independently is incongruence between SOA and project goals. Projects only care about their own timelines and objectives. If a project or application team creates a service and assumes ownership over it, the same team will have to be responsible for making changes needed by its future consumers. It will have to address new requirements that are coming from a different team, comply with that team’s timeframes, and ensure that changes do not impact existing consumers. These responsibilities represent a complete departure from traditional project goals and therefore are simply not be adhered to in many situations. As a result, the decentralized ownership of services ends up leading to increased duplication and an overall failure of the SOA program.

The way to address this issue is clear – create a centralized team responsible for shared service design, development, testing, and support. It will take responsibility for reconciling all of the service-related requirements, designing services

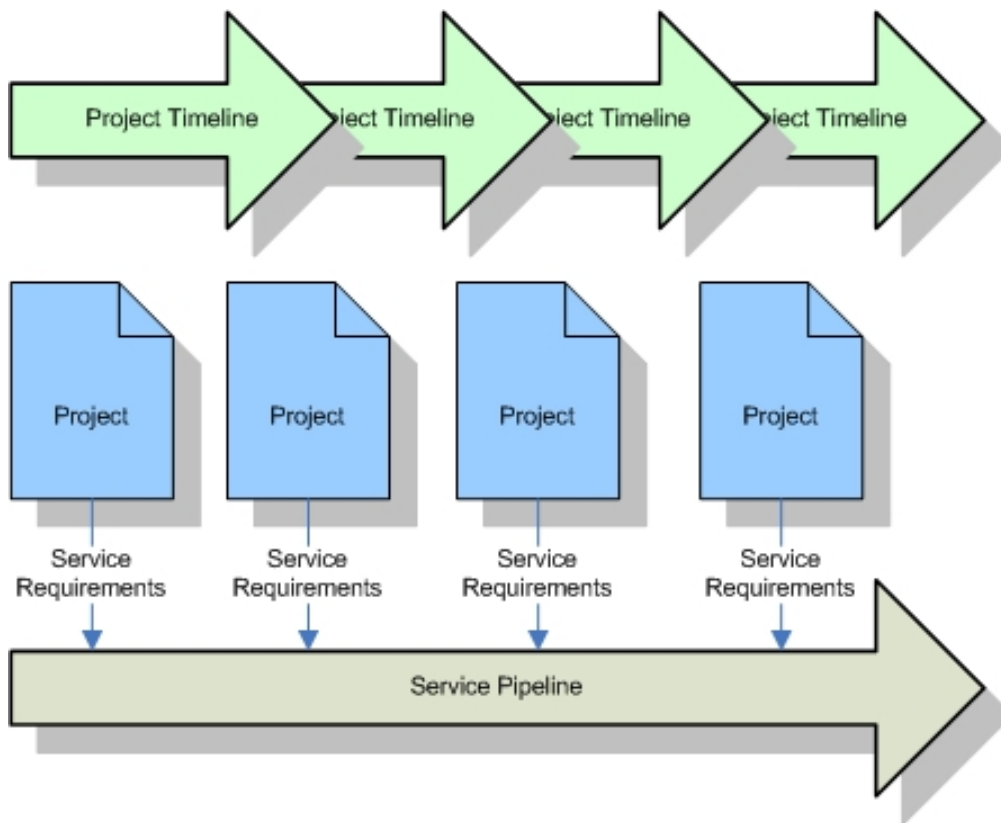
to address them, establishing and enforcing SOA standards, ensuring proper scalability of services, and managing the services as independent software products. Gartner often calls this type of a team the Integration Competency Center (ICC). This approach eliminates the previously outlined problems and minimizes the project-centric focus because services are delivered by a separate group whose primary goal is the advancement of SOA. Additionally, a centralized team is better suited for driving the adoption of shared services in the most efficient and effective way.

## Service Lifecycle Management

Most organizations deliver business initiatives via IT projects. Therefore, projects will most often drive the demand for services. There is, of course, a better way to identify what reusable services are needed, by whom, and when. A comprehensive business process mapping will create a clear roadmap for service identification and demand. Unfortunately, many companies still choose not to move in this direction and continue to allow projects to remain the driving force behind service identification and implementation.

When a service is designed and developed to address specific project needs, it is not fully reusable. New consumers invariably need changes introduced to the service to comply with their requirements. This typically involves field changes or additions, new operations, addition or removal of major entities, and even potential business logic changes. The biggest SOA secret is that services are almost never reused as-is – changes and integration costs are practically unavoidable.

In order to make services truly reusable and ensure maximum leverage, the service lifecycle must be centrally managed. The central team responsible for the delivery of services must also be charged with service identification, lifecycle management, and pipelining activities. All of the disparate service requirements supplied by different projects must be accumulated together to create a comprehensive view of the service pipeline and roadmap. Figure 2 depicts the relationship between project needs and a service roadmap.



*Figure 2: A service roadmap as influenced by projects.*

The key to the pipelining activity is accurate and timely information. This will make it possible to incorporate different project requirements into the service that is supposed to be created for a specific project. If the project timelines are close enough and the requirements are well defined, service designers should attempt to include as many of them as possible into the current release. This introduces efficiencies, scalability, and agility into the delivery mechanism through accumulating processes and information gathering across multiple projects. Centralized and consistent

service lifecycle management can make this a reality.

### Minimizing Impact of Changes and Maximizing Reuse

Since changes to services are inevitable, the architecture and design patterns must be established with change in mind. Services must be designed in such a way that most of the changes introduced as part of its evolution will have minimal, if any impact on its current consumers. Another goal of service design should be to maximize service reuse as this represents the cornerstone of SOA. Both of these goals can be achieved by using the Service Façade and Concurrent Contracts design patterns [REF-1] together with a canonical modeling approach.

Canonical modeling is a well known and established approach for abstracting service consumers from the backend data sources and introducing a common entity representation. Many SOA proponents agree that canonical modeling is a critical component in the success of the SOA program. A canonical, or standard, model attempts to establish a single, consistent representation of all the entities that will be passed through a shared service. This representation should be independent from the backend data structures and service consumer specifics, which will minimize the impact on consumers when either one changes. Additionally, because LoBs may represent the same entities in a different way, a single canonical model will help reconcile these differences and allow different parts of the organization to speak the same language. This, in turn, maximizes the potential and real reuse of services built across the company.

The Service Façade pattern is used to minimize the impact of service changes on its consumers. Every service, whether it is built using a canonical model or not, should expose a façade interface via Concurrent Contracts specific to each consumer. Consumers would not access the service directly but rather through its exposed façade contract. Each façade should be designed in such a way that it presents data in a form easily understood and ingested by individual or group of service consumers. Figure 3 shows how the façade pattern should be used to design and build shared services.

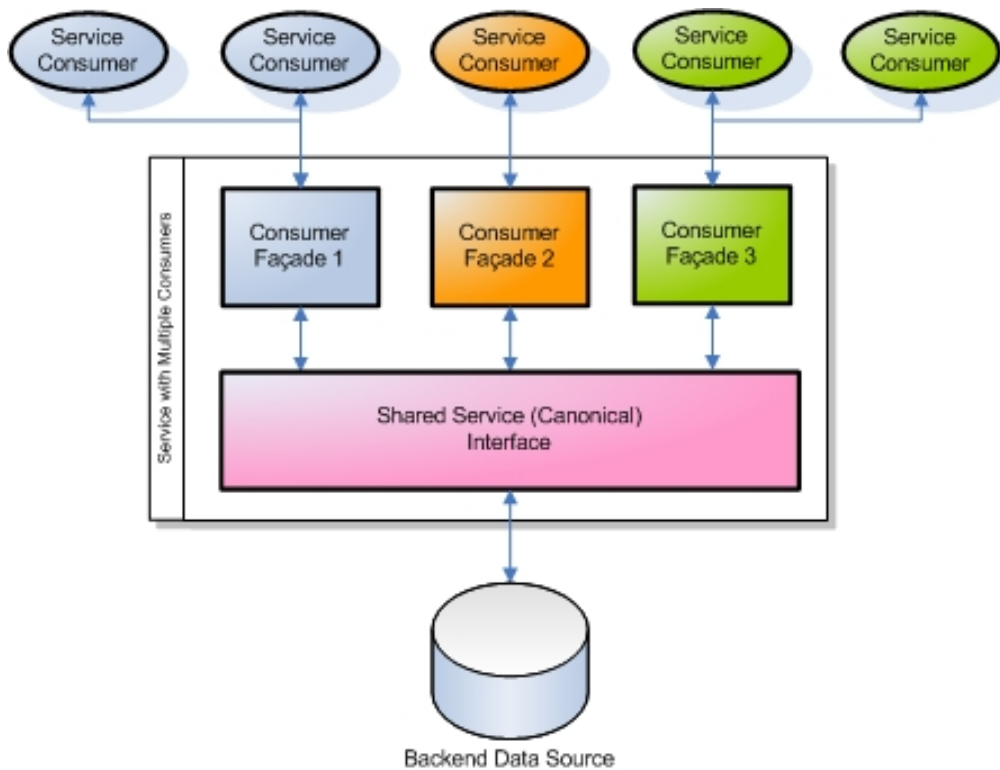


Figure 3: The Service Façade pattern in action.

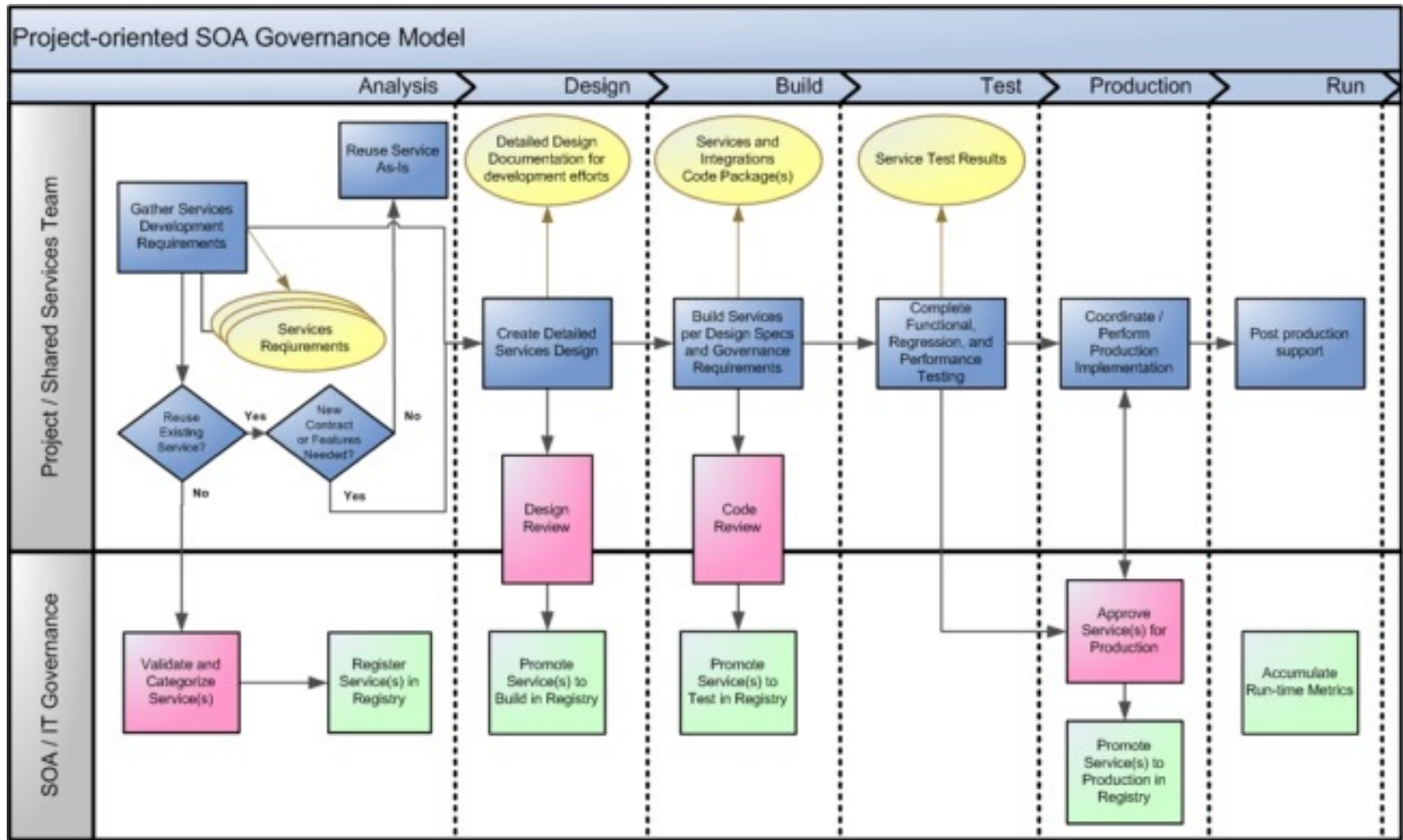
Because each façade contract is specific to one or many service consumers and does not expose internal (canonical) service contract, changes to the service implementation or even to the canonical model will have minimal, if any impact on the consumer. The mappings between the façade contract and canonical structure may need to be updated but this activity will be transparent to the consumers. From their standpoint, no changes will take place. The Service Façade pattern also ensures maximum reuse of the service because the same service instance is being used under the covers even though multiple façade interfaces may be exposed.

## SOA Governance

All of the best processes and architectures cannot be effective if they are not being followed. This is where governance comes into play. Establishing and enforcing effective governance mechanisms and processes is paramount to the success of any SOA program. The key is to ensure minimal overhead, maximum compatibility with the existing IT governance processes, and high level of synergy with all the Software Development Methodologies (SDM) being followed in the organization. The success of the SOA program is dependent on how efficient it is, how closely it can be integrated with the existing processes, and how strongly its recommendations can be enforced.

SOA governance and the project-based IT culture are largely incompatible. SOA governance inserts checkpoints into the normal flow of software development, while projects are primarily concerned with hitting their timelines at all costs. If the governance mechanisms detect a problem and ask a project to make changes, this can lead to unpleasant conversations at even the executive level. Confrontations like these may often be unavoidable; however, the best way to ensure that both governance and project goals are met is to attain visibility into the project pipeline as early as possible and influence each project's direction to be consistent with the established SOA guidelines. SOA governance doesn't have to be focused primarily on enforcement and catching non-compliers but instead should concentrate on exploiting synergies with the already established SDM processes and influencing the solutions.

Figure 4 depicts a sample SOA governance process that can be employed to satisfy both project and SOA program goals. Note that it is designed to be easily integrated with the typical waterfall SDM. The governance checkpoints are very light and can be completed in a short period of time, so that projects do not lose valuable time navigating complex governance processes. At the same time, however, all the governance steps are timely and ensure proper compliance throughout the software delivery process. They are designed with influence, not enforcement in mind.



*Figure 4: A sample SOA governance process.*

Since all the projects that utilize existing or create new services have to follow these governance mechanisms, many goals of good SOA governance can be achieved. Getting exposure to the relevant projects as early as the Analysis phase provides the ability for SOA governance to influence their direction. This also enables the Shared Services team to gain insight into the complete services pipeline, accumulate all the related requirements together, and plan service releases appropriately. Each governance checkpoint represents an opportunity to validate whether previous recommendations have been implemented and, if not, reject the project from moving forward. Most importantly, the SOA governance process should be given an opportunity to completely close the loop on all the changes or new services being deployed, which should be represented in the formal approval or rejection of moving the code into production.

Another important element of the SOA governance process that is depicted on Figure 4 but not yet discussed is the role of the Registry/Repository (RegRep). Each checkpoint prompts an action related to service registration or promotion. This is necessary not only to document a true state of the service but also to formalize and automate the whole SOA governance process. Many RegRep tools contain governance automation capabilities. Exploiting them is extremely valuable since it streamlines the whole process and eliminates inefficient manual steps. Establishing a policy that all services must be registered to be consumable closes any loophole projects can try to exploit. Since registration and promotion steps are tightly coupled with the SOA governance checkpoints, which need to be performed by an independent party, projects will not have an option to sidestep any of them. If they do, services will simply appear to be unavailable for consumption or in a state incompatible with the project needs. Finally, registries can be used to collect run-time service utilization metrics, the importance of which is discussed shortly.

## **Service Funding**

Funding for the SOA program should come from a central source. It should cover everything from the shared infrastructure, technology, tools, and methodologies. Where the money comes to build individual services, however, presents a bigger challenge. Since projects are the primary drivers behind demand for services, special consideration should be given to project needs and budgets. As discussed earlier, individual service's pipeline and roadmap should be independent from those of a project. Thus, service design and implementation can incorporate additional requirements that fall outside of the project scope. Another typical project-related problem stems from the shared nature of services. It is unfair to burden a project with the full cost of a service that will be utilized by a number of other consumers.

There are three possible ways to address the service funding concerns.

1. Make the first project to build a service provide the complete funding
2. Establish a central funding source that will cover all service design and construction expenses
3. Provide supplementary funding to projects building services

If option 1 is selected, several strategies for recouping the initial investment can be used.

- a. Do not recoup the investment
- b. Place a surcharge on each instance of service leverage
- c. Charge a small fee for each service call

As mentioned above, it is unfair for the project to carry the complete costs of the service build-out, especially if it includes additional requirements. Thus, unless the project implements one of the options to recoup its initial investment, funding option #1 is not going to be viable. Not recovering the funds is not a realistic option either as it does not incent the projects to build truly reusable services. The other cost recovery strategies may work but require detailed metrics to be captured on the service leverage and/or transactional volume.

Establishing a central funding source for all projects to use when building reusable services is probably the ideal approach. Few companies, however, would be willing to write what in essence would be a blank check for the projects to use in their service delivery efforts. The opportunity for abuse and misappropriations would be too tempting. Unless

strong governance and control mechanisms are in place, this funding method will most likely end up costing the company more money and provide unrealistically small return on investment.

Providing supplementary funding to projects building services is probably the most realistic approach. A central fund needs to be established to cover the efforts falling outside of the project scope. Since shared services would typically incorporate other projects' and enterprise requirements, the actual cost ends up being higher than what projects budgeted for their needs. Thus, the easiest way to distribute supplementary funding is to allow the projects to pay for functionality already included in their budgets and cover all the additional costs through the central fund.

Whatever the funding approach is used, it needs to be carefully administered. A party not involved in day-to-day project work is best suited to play the administrative role. The Shared Services team is the most likely candidate to control the budget and use it appropriately to further the SOA program adoption, increase service leverage, and avoid political influences.

## SOA Metrics

Once the SOA program is up and running, its effectiveness, level of adoption, and results need to be measured. This can be achieved through the collection and communication of the relevant metrics. The most popular SOA measurements are the number of services created, amount of service reuse, and cost avoidance/savings.

Since the central Shared Services team has a complete view of all the current and future service creation or reuse opportunities, it is in the best position to collect and report on the metrics. In order to accumulate accurate metrics and produce relevant reports, the following steps need to be performed.

1. Capture all the services that are being created
2. When completed, determine the cost to build each service
3. Capture all reuse opportunities

All the steps above should be completed for each project that either creates or leverages services. Any modifications to the existing service should be counted towards the total cost of the build.

Once all the data has been collected, cost avoidance can be calculated. The basic formula for individual service cost avoidance as related to a specific project is provided below. *Service Cost Avoidance = Service Build Cost – Project's Service Integration Cost*

*Where*

*Service Build Cost = Initial Service Build Cost + Cost of all Subsequent Changes*

To calculate the entire project's cost avoidance amount, simply add the cost avoidance for all the services being leveraged. To forecast the total potential cost avoidance at any point of time, multiply the number of times each service is envisioned to be leveraged by its build cost and add it all together. Since the integration costs for each ongoing or future project can only be estimated, a standard reuse factor can be applied to the service build cost. 80% is the typical number used in these situations. Note that projects creating services should not count towards cost avoidance.

In cost avoidance calculations and projections, understanding each service operation rather than the whole service reuse will lead to more accurate results. To achieve this, current and future reuse opportunities should be tracked at the service operation level. This level of granularity might be hard to achieve, however, especially when keeping track of the build costs. Thus, approximation techniques can be used that determine the operation build cost based on some manipulations of the total service build cost. Dividing the total service build cost by the amount of operations produced might be the simplest approach. If tracking is performed at the service operation level, the cost avoidance formulas above need to change to indicate operation rather than service specific metrics.

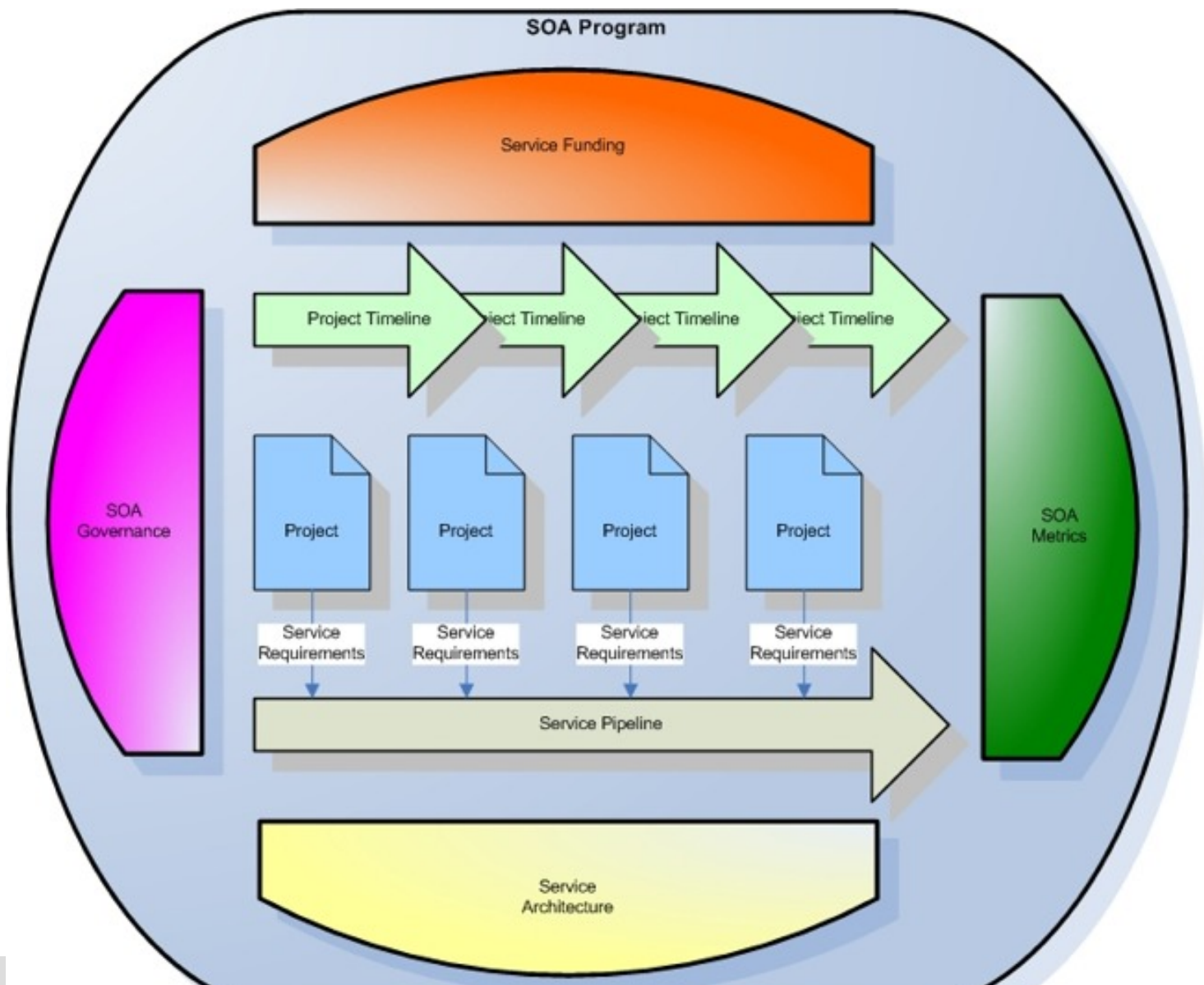
Once metrics are collected, they need to be distributed to all the SOA program stakeholders. Depending on the organization, it can be IT managers and executives, business executives, and partners. Metrics should not be reported simply for the sake of sharing the progress made by the SOA program but rather to support SOA program's goals and influence the behavior leading to achieving them. Specific targets need to be set by the IT executives and metrics should be used to determine whether they have been successfully reached or not. Metrics collection and reporting should be performed by a central team in order to ensure that the whole process cannot be compromised to misrepresent the reality or serve individual team's or group's interests.


## Conclusion

Regardless of what the technology vendors would like you to believe, SOA is a complex concept. A number of elements need to come together to truly achieve service orientation. A lot of work needs to be done to establish a successful SOA program. Yet, all of this has to be done in conjunction with delivering projects. The business does not stop. It does not and cannot wait for the SOA program to be established, fully built out, and all the services delivered. Therefore, most SOA programs face the challenge of dealing with projects while at the same time trying to deliver on their high level promises.

To address the SOA and project goal incompatibilities, service lifecycle management, architecture, SOA governance, funding, and SOA metrics need to be brought together in a comprehensive program. Creating a central team to manage this process will result in more consistent deliverables, more efficient operations, less opportunity for political influence, and faster attainment of SOA benefits.

Projects with SOA potential should be considered part of the overall services pipeline. Cumulative requirements should drive service design and development. The service architecture needs to be flexible enough to accommodate changes, minimize the impact of service changes on the existing consumers, and maximize service reuse potential. SOA governance should influence the projects to make the right decisions and catch non-compliers if necessary. A comprehensive view of the project pipeline should make this process streamlined and efficient. Specially designated funding solutions should eliminate the disincentive for projects to build reusable services. Finally, the SOA metrics should demonstrate the achieved results and influence the right behavior. Figure 5 demonstrates the relationship between all the project-oriented SOA elements.





*Figure 5: A view of project-oriented SOA.*

Those organizations that embrace the project-oriented approach to SOA will have better success in SOA adoption and delivering results. At the end of the day, the business doesn't care how many services have been built or leveraged. What really makes the business executives tick are the sales, new product introductions, new customers, real savings, achieved efficiencies, and everything else that deals with growing revenue and impacting the bottom line. Enabling business agility is the primary goal of SOA. Establishing an approach that delivers both the SOA program benefits and business goals of faster time to market and cost savings will undoubtedly make IT and the whole organization successful.

## References

[REF-1] SOA Design Patterns (Prentice Hall), [www.soapatterns.com](http://www.soapatterns.com)

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL



[Home](#) [Past Issues](#) [Contributors](#) [What is SOA?](#) [SOA Glossary](#) [SOA School](#) [SOA Books](#) [About](#) [Legal](#)

Copyright © 2006-2008  
SOA Systems Inc.