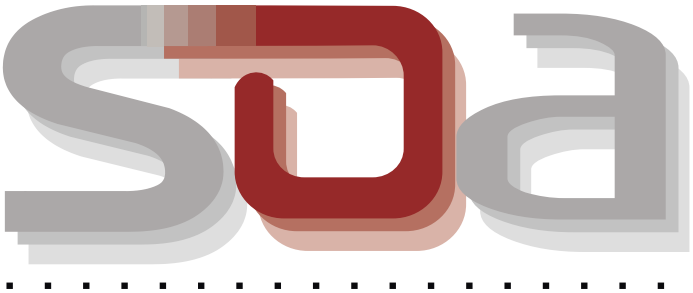
Torry Harris



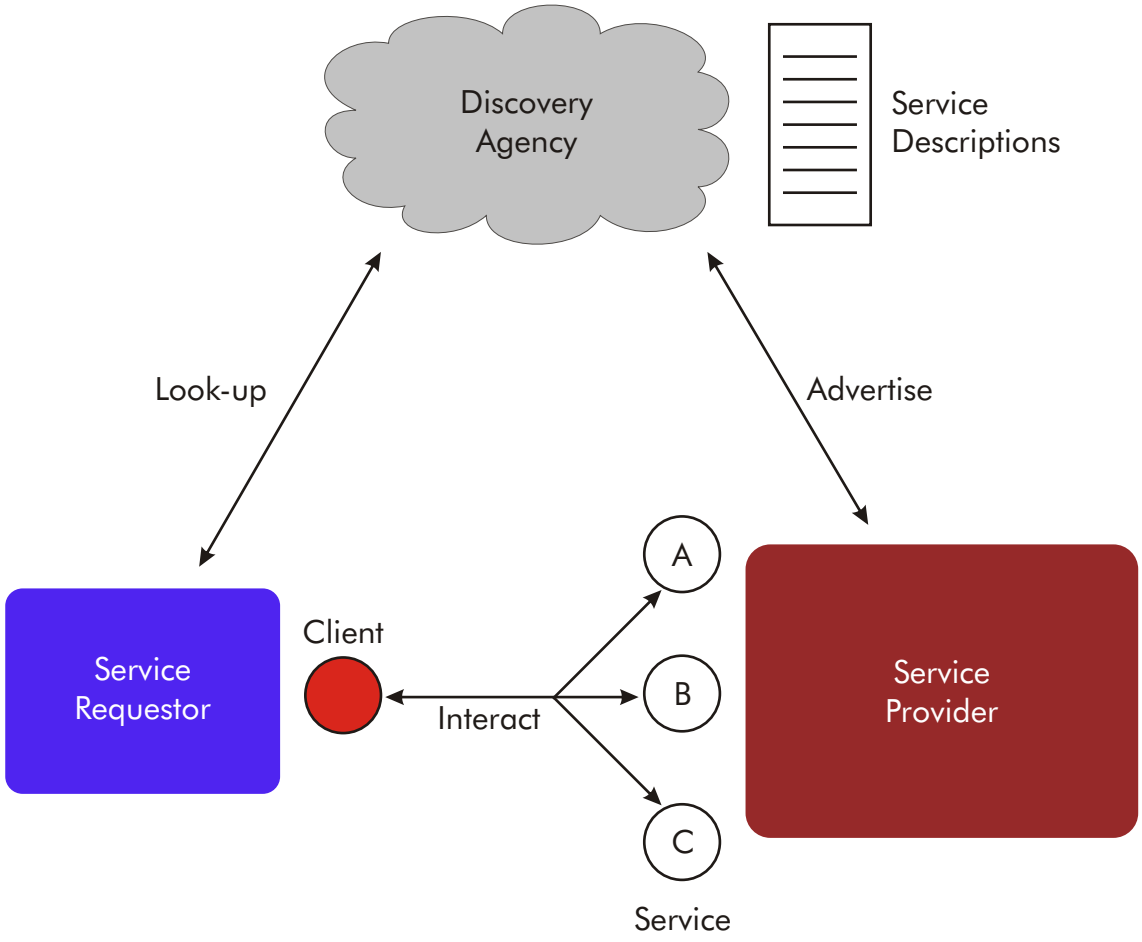
SOA Best Practices

Torry Harris

SOA

What is SOA?

SOA or "Service Oriented Architecture" has rapidly gained status as a buzzword among the IT community, particularly over the last three years. The demand for greater simplicity of reuse and maintenance of functionality has fostered the evolutionary journey from simple Object Oriented Programming, via component based software to SOA. Another reason why SOA has gained even more momentum is that network capacities have increased phenomenally, making distributed designs feasible. The dawn of platform independent languages such as XML and enterprise strategies such as Web services have also aided SOA.



Service Oriented Architecture

Web Services and XML are certainly not mandatory to achieve an SOA, but they are definitely quite useful for the purpose. However, with different vendors rolling out new strategies on the Web Services front almost on a monthly basis, it can become difficult for organizations to find guidelines to follow while the situation stabilizes. This list of basic "Do's and Don'ts" attempts to define a strategy that could help mitigate risk and maximize business value

Torry Harris

SOA

The Dos:

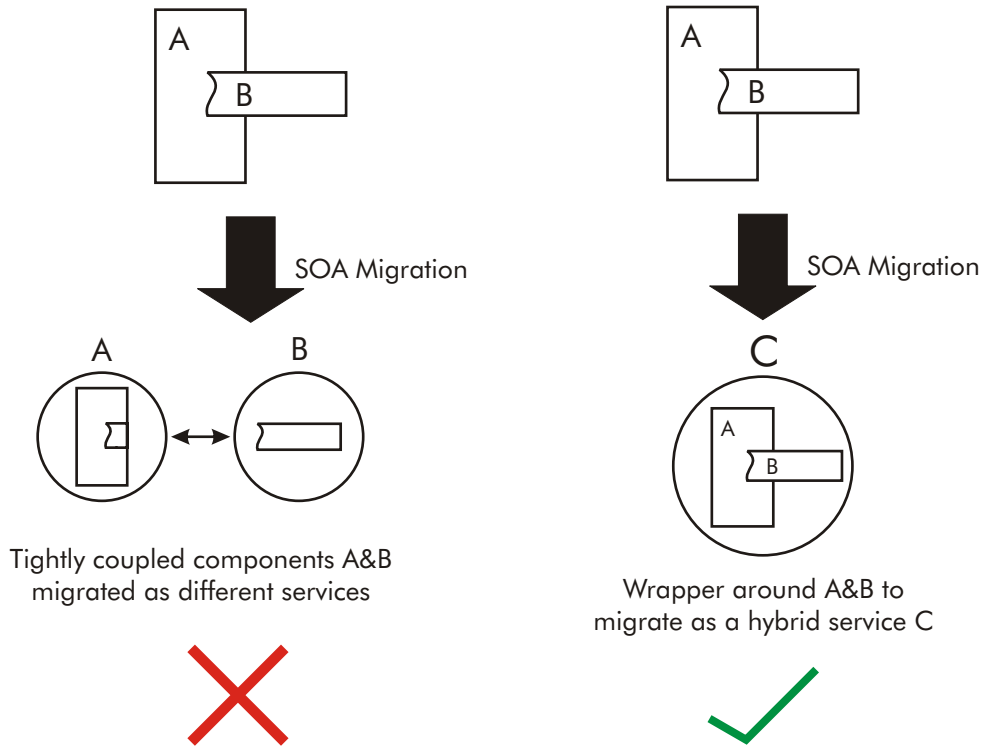
DO follow established standards

Often while developing a new design, it is tempting to tweak the established standard just a little bit to make it fit perfectly. However, with SOA, the aim is to maintain compatibility with other standards based implementations. Tampering with a standard ends up creating a proprietary implementation, which may later cause integration problems.

For example, altering a SOAP message format to augment speed requirements may result in the client being unable to understand the request altogether. Such issues may also remain dormant initially, which would mean that rectifying them at a later stage would be proportionally costlier.

DO try and use wrappers for a set of tightly coupled components

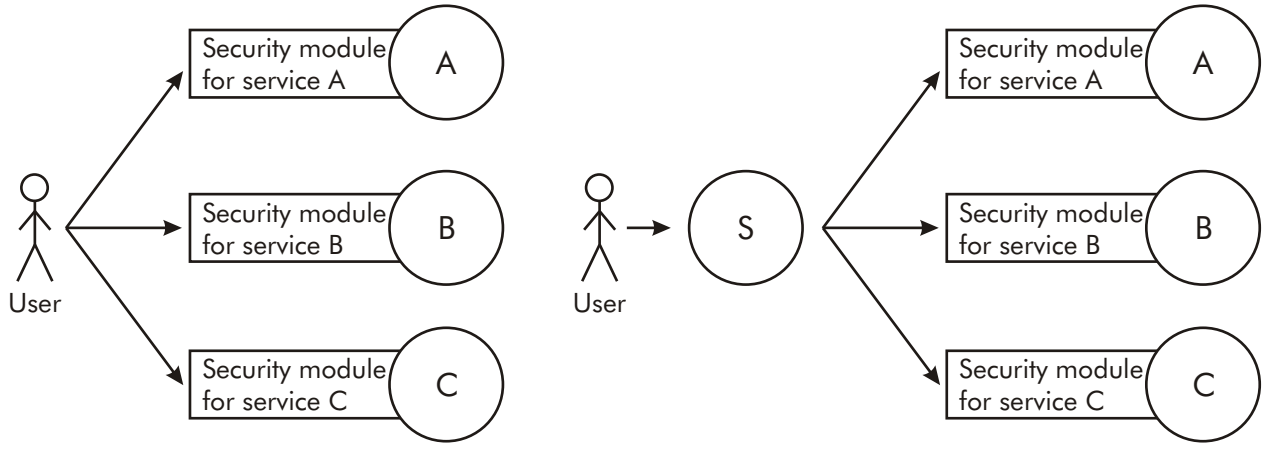
Legacy designs frequently consist of sets of components, which are tightly coupled to each other in terms of interfacing and communication. When migrating to an SOA it is important to keep in mind that every individual component need not be exposed as a service. If exposing each component is not feasible, it is possible to use a wrapper around a set of components and expose the package as a service. However, while wrapping components, access policies and business context must be kept in mind. Randomly wrapping components and exposing them as a single service can lead to security and performance problems. Ideally the components, which are wrapped, should constitute a unit that can interact with other components independently.



DO pay heed to security

Since SOA as a concept depends upon shared, re-usable services, it is mandatory to keep security in mind while incorporating it. This can be a fair sized job, since if each service is to have an authentication mechanism, it becomes tedious both in terms of development as well as usage, to manage all the credentials involved. This often leads to security being relegated to a secondary status, in favor of functionality and ease of use.

A possible workaround is to have a single service act as the authenticator for all the services. This way, the authenticator service provides a sort of proxy between the client and the services providing business functionality. Also since the authentication service is an independent component, it can have stronger protection features built into it.



DO ensure interoperability

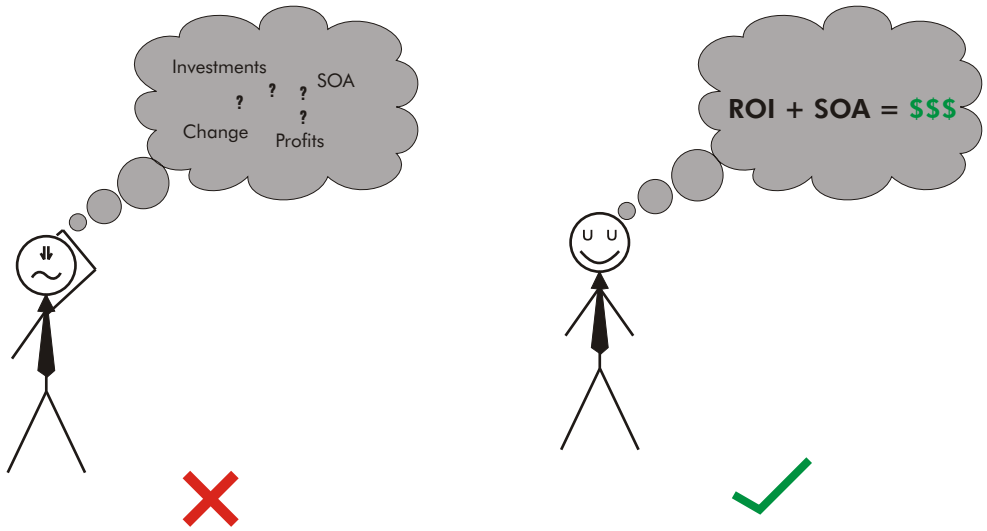
SOA is all about choice. Since every player in the field may have a different strategy, in order to ensure that the maximum number of consumers can access your service, it is important to design it in such a way that it can be interfaced from any platform and any programming language. Using a platform independent standard ensures that your users are free to choose their own implementation mechanisms. Apart from simply being friendly, this also means that your users will have to go to virtually no trouble at all to incorporate your service into their existing architecture, which increases the business value of your service.

For example, a service based on XML and SOAP over HTTP for its communication mechanism can be accessed from a various number of platforms as opposed to one that uses COM.

Torry Harris SOA

DO align ROIs with migration strategies

When justifying a move to SOA, an ROI kept in mind can be a reassuring factor, particularly when non-service oriented investments are already in place and the organization is fairly content with them. Usually, research performed for migration strategies is focused on technology and implementation, rather than on high-level advantages to the organization. A smart strategy for integrating an SOA can provide greater cost advantages than an ROI originally predicted.



Even though the initial SOA migration may be the most expensive part of an enterprise-wide initiative, the scope of your ROI will probably extend beyond the migration phase. Frequent revisions to an ROI, keeping in mind migration strategies, will improve the accuracy of its predictions as they relate to subsequent phases in a long-term program.

With SOA, the gains tend to be more tangible, because the interoperability provided by the service integration layer results in immediately recognizable savings.

The Don't's:

DON'T perpetrate tightly coupled interfaces

The essence of SOA is in its loosely coupled, highly interoperable nature. When tightly coupled interfaces are used, the number of users who can effectively access the service drops. Also, tightly coupled interfaces require clients to have inside knowledge about the working of the service, which could constitute a security risk, particularly if the clients to the service are not completely within the scope of the service providing organization. If a set of components using tightly coupled interfaces must be used, then it is most expedient to put a wrapper around them and use the group as a single package instead.

For example, if a pair of modules share a communication mechanism that is unique between them and not understandable by other modules, then instead of exposing each of these as a separate service.

Torry Harris

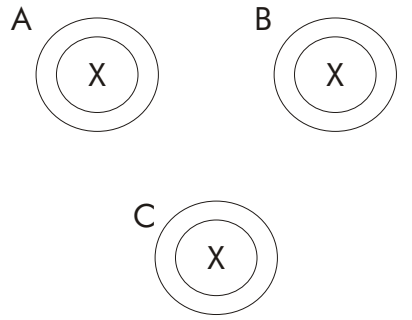
SOA

DON'T completely rely on ultra new technology, which may not be fully tested

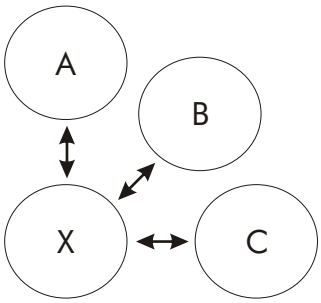
In addition to the fairly robust core standards forming the basis of Web Services and SOA, there is also a family of rapidly developing new applications from a variety of vendors, ranging from well-established corporations to fly by night outfits. While it is important to keep abreast of new technology and build on it where feasible, one needs to be slightly wary of new technology until it has passed rigorous testing under the scenario that it will ultimately be used in. Therefore, newer technology should typically be incorporated in the comparatively low risk areas of the architecture

DON'T unnecessarily clone existing services

The aim of SOA is to re-use existing services that are in place. Even though it may be tempting to merely clone and modify an existing service to create a new one, this is not a good strategy for SOA. In fact, the problems created by cloning are some of the fundamental problems that SOA seeks to solve. Primarily, when code is reused as opposed to functionality, bugs in the core code get replicated across the services. Secondly, documentation for cloned services is frequently scanty, resulting in maintenance nightmares. Thirdly, cloning leads to redundancy, which debilitates the SOAs' benefits and affects performance. Finally, any change in the basic service would have to be copied across all the services. All of these together make cloning a bad choice. It would be smarter and safer to isolate the functionality that is repeated and expose it as a single service.



Services A, B and C replicate functionality X



Services A, B and C obtain functionality X from service X

DON'T put all your eggs in one basket

Despite the fact that SOA is the coming wave in enterprise interaction, it is advisable to adopt it in a phased approach, starting with a low risk application or pilot program to determine the exact needs of your organization and how SOA can aid your business. Once the requisite knowledge base is created, then the architecture can be expanded to other spheres of greater importance.

DON'T lock yourself in

When planning for an SOA, it is best to avoid proprietary mechanisms, such as message protocols and software applications, since frequently these require clients to conform to their standards. This reduces interoperability and causes clients to re-assess whether they can avail the service provided within their own budgets or whether they need to move to another provider.