Introduction

Effective, systematic software reuse continues to be an elusive goal for many organizations. The business motivators for reuse are readily available today. IT cost reduction, and the desire for flexibility and responsiveness in IT architectures and underlying systems probably head the list.

Often you can even identify the inhibitors to achieving more successful reuse (or at least it seems that way). Yet the answers for overcoming these inhibitors do not come readily. At the same time, a technology evolution in the form of Service-Oriented Architecture (SOA) provides a foundation to achieve software reuse. A key part of the SOA value proposition is the benefits realized from software reuse.

SOA as an enabler for reuse

A survey of various SOA best practices articles in developerWorks and in other developer resources indicates that there has been much written about the importance of the interface in SOA. The service interface is the essence of the integration design. Combined with the use of standards, interfaces are the essential ingredient for creating a loose coupling where service clients and service providers can communicate regardless of programming language and platform. Services are to be independent, in that clients need not understand the inner workings of a service component; essentially the service operates as a "black box." "White box" reuse, or cut and paste, where source code is modified in order to use in another context, while useful, is not typically as beneficial as "black box reuse." "Systematic reuse programs encourage reusing software without change because of the superior benefit they receive from black-box reuse throughout the life cycle" (qtd. from *Measuring Software Reuse: Principles, Practices, and Economic Models* -- see Resources). Services encapsulate function, whether it is business function or common utility function. SOA attributes are discussed in more detail in the developerWorks article "Migrating to a service-oriented architecture" (see Resources).

All of these characteristics of SOA provide a technology environment where software reuse can thrive. However, even with the reuse enablers that SOA concepts and technologies provide, traditional reuse inhibitors can still be an issue.
Reuse inhibitors

In most development organizations, software reuse occurs on a regular basis in at least an ad hoc manner. Code is shared across projects in an informal manner. SOA provides the mechanism for more formal reuse, where the reuse occurs when services are accessed by service clients to perform a given function. The reuser (service client) does not really even know what code they are reusing, and don't really care. They just know that the service is providing the function that they require. So what are the issues? Here we examine some of the challenges associated with the creation and usage of reusable services.

- **Education and culture**: SOA technologies such as Web services may be new to some project teams and thus require learning new skills before they can begin to leverage available services. The extent of the ability for project teams to efficiently find available services, understand what they do, and determine the benefit to their project may well determine how likely they are to realize services reuse value. UDDI registries are important but not enough. WSDL documents are designed to be machine readable, and while they are also readable by humans, teams looking to reuse services typically require additional information. The services learning curve takes away from reuse value and is thus a factor for project teams in deciding whether to reuse or to instead reinvent.

- **Availability of reusable services**: Obviously, reusable services of value must exist in order to realize services reuse. Determining what these services would be in a given organization and actually getting them created and made available can be a challenge. Some level of domain analysis must be done to determine the commonality that exists across the domain, and thus determine what types of services would have reuse value. Project teams in an organization create software for their solutions that may well contain good services candidates for harvesting and hardening into reusable services in a cost effective manner for use on a broader scale. Understanding where these opportunities exist can be a challenge. Providing organizational constructs that enable execution on identified harvesting opportunities can be harder still.

Reuse engineering

Reuse engineering in its essence consists of reuse experts (reuse engineers) that are employed in positions that span across project boundaries. They aid project teams in consumption and production of assets. In their Journal of Systems and Software article, "Incentive compatibility and systematic software reuse" (see Resources), Robert Fichman and Chris Kemerer provide this description:

"Reuse engineers from the reuse center are actually "loaned out" to applications development teams to assist them with the practice of reuse. On the reuse consumption side, reuse engineers supply knowledge of what is potentially reusable, and assist in any necessary adaptations. On the production side, the reuse engineers identify where the project has the best opportunities to produce new components, and assists with the task of making those components reusable. At the end of the project, the reuse engineers take back knowledge of the domain, new assets to be made reusable, and knowledge of which existing assets are good "as is" and which need to be fixed or enhanced."

Reuse engineering for SOA
Reuse engineers must be technically adept, have project-level experience, and ultimately are the enablers for reuse. A key attribute of the reuse engineer role is that their position spans across project teams. Software reuse is the reuse engineer's "day job," and they will typically have targets and measures associated with reuse results in their sphere of influence. Investment in a role such as this that has a broader view than a single project is very important to the success of a reuse program and to a SOA initiative. Very few organizations practice systematic reuse successfully because of the focus on a project at a time instead of a broader focus ([Software Reuse: Architecture, Process and Organization for Business Success](https://www.ibm.com) -- see Resources). In my experience with applying reuse engineering to an organization, I found it to be very effective in addressing traditional reuse inhibitors, such as those described earlier. With each of these inhibitors, I'll explore the impact of reuse engineering.

### Education and culture

Reuse engineers with SOA skills who work with project teams can lower the barriers to adoption of SOA by those teams. Project managers are trained to mitigate risk on their projects. Searching for and reusing someone else's software can be risky, but reuse engineers mitigate these risks in that they are experts at locating available services. They can work with the project teams to assess and understand needs, and then expedite the search, understanding, and analysis of services and guide teams in how and when to use them. They share experiences from other projects that have used the services. All of this lowers the learning curve and the reuse risks. These engineers can educate teams on the best ways to migrate to SOA services usage with the least impact to their existing software. An example would be to introduce the usage of the Adapter design pattern to minimize disruption to application code when making changes to use a service instead of providing the capability within the application.

Reuse engineering also provides benefits in terms of positive impact on organizational culture. Software reuse is not new, and has typically been attempted many times over the years with varying degrees of success by development organizations. As a result, reuse is something that most people have an opinion about, whether it be positive or negative or somewhere in between. Reuse engineers can understand project needs that can never be expressed in a search box or form. The human interaction provides a communication level which cannot be addressed with search tools and asset repositories. This creates a great rapport between the project team and the reuse engineer which further assists the creation and adoption of a reuse culture, as project teams become more aware of what can be accomplished with cooperative reuse efforts. Reuse engineers become a channel and a conduit for project teams to bring forward opportunities for creation of reusable software assets in the form of services.

### Availability of reusable services

Services creation can be achieved in many ways. Understanding what services are of value to an organization is the first step. SOMA (Service-Oriented Modeling and Architecture -- see Resources) is a very prescriptive method that includes several complementary approaches to services identification, including domain decomposition. Reuse engineering is complementary to SOMA, in that it provides a means to understand the subject domain.
Domain analysis is a means to determine the commonality that exists across the domain, and thus determine what types of services would have reuse value. Domain analysis can be achieved in a number of ways. With a reuse engineering model it occurs one project at a time. As reuse engineers work with project teams they can capture the capabilities that project teams need to deliver. Analysis of this information in a given domain reveals where the duplication of effort exists and where services could be introduced to create reuse value. This approach can be especially effective in understanding opportunities for broad or horizontal reuse that may be left undiscoverable due to gaps between enterprise architectural directions and actual solution architectures delivered by project teams, as described in the developerWorks article, "Elements of service-oriented analysis and design" (see Resources). These common services opportunities are appealing in that they can quickly demonstrate the value of SOA and deliver a large return on investment, because the number of reuses can be quite large.

As an example, consider common capability needs such as user access revalidation for applications within an enterprise. With the increased attention on regulatory compliance, business controls such as these are that much more important. Because of the breadth of the requirement to provide the capability, solutions may indeed be reinvented over and over across organizations, with the end result that none of them are really reusable, there is a lack of centralized control for the business, and there are a lot of wasted IT dollars. This type of a capability could well be delivered as a broadly reusable set of services that applications across a heterogeneous enterprise could use. Stakeholders at various levels, from the CFO to the application developers, then realize the value of SOA and reuse in a practical way.

As reuse engineers discover these opportunities, they also capture knowledge that can reduce the migration costs to SOA. Specifically, the reuse engineers catalog services candidates. This is software that project teams have created for their solutions that may well be good candidates to harvest and harden into reusable services that meet the needs identified through domain analysis. Understanding where these opportunities exist is traditionally a challenge. Reuse engineers can categorically determine best of breed implementations and bring forth the best opportunities for generalization into services. Reuse engineering teams can also take on the harvester role and deliver the services. Alternatively they can bridge the work done on a prior project with the requirements of a new project and enable hardening into a service within the scope of the new project. This "next project generalization" model (Measuring Software Reuse: Principles, Practices, and Economic Models -- see Resources) can increase availability and usage of reusable services and do it without requiring new investment streams. Reuse engineering is the key to discovering and acting on these opportunities that are often lost.

**Summary**

Reuse is a necessary attribute of an effective SOA and contributes to the business case for SOA. Reuse engineering is an effective means of applying software reuse best practices to SOA and can deliver efficiencies in the production and consumption of services across an enterprise.