WHAT IS SERVICE-ORIENTED ARCHITECTURE?

Service-oriented architecture, or SOA, is a flexible set of design principles for constructing complex software-intensive systems from a set of universally interconnected and interdependent building blocks, called services. These services make up functions which developers make accessible over a network in order to allow users to combine and reuse them in the production of applications. These services and their corresponding consumers communicate with each other by passing data in a well-defined, shared format, or by coordinating an activity between two or more services.

SOA was conceived to achieve the goals of increased interoperability, increased federation, and increased business and technology domain alignment in systems development and integration computing. SOA as an architecture relies on service-orientation as its fundamental design principle. If a service presents a simple interface that abstracts away its underlying complexity, users can access independent services without knowledge of the service’s platform implementation. SOA realizes its business and IT benefits through utilizing an analysis and design methodology when creating services. This methodology ensures that services remain consistent with the architectural vision and roadmap.

A service may be a new application; but services are frequently wrappers around existing applications that make them network-enabled. Once an application is wrapped in a service, it is possible to create interfaces to that application. In SOA large scale, distributed applications can be built that leverage combinations of these services that may not have been anticipated by the developers of any of the included services.

Guiding principles that define the ground rules for development, maintenance, and usage of SOA include: reuse, granularity, modularity, compositability, componentization and interoperability; standards-compliance, and; services identification and categorization, provisioning and delivery, and monitoring and tracking.

Specific architectural principles for design and service definition focus on themes that influence the intrinsic behavior of a system and style of design:

- **Service encapsulation** – Many services are consolidated for use under the SOA. Often such services were not planned to be under SOA.
- **Service loose coupling** – Services maintain a relationship that minimizes dependencies and only requires that they maintain an awareness of each other.
- **Service contract** – Services adhere to a communications agreement, as defined collectively by one or more service-description documents.
- **Service abstraction** – Beyond descriptions in the service contract, services hide logic from the outside world.
- **Service reusability** – Logic is divided into services with the intention of promoting reuse.
- **Service compositability** – Collections of services can be coordinated and assembled to form composite services.
- **Service autonomy** – Services have control over the logic they encapsulate.
- **Service optimization** – All else equal, high-quality services are generally preferable to low-quality ones.
- **Service discoverability** – Services are designed to be outwardly descriptive so that they can be found and accessed via available discovery mechanisms.
- **Service relevance** – Functionality is presented at a granularity recognized by the user as a meaningful service.

Web services can implement a service-oriented architecture and make functional building-blocks accessible over standard Internet protocols independent of platforms and programming languages. These services can represent either new applications or just wrappers around existing legacy systems to make them network-enabled. (From Wikipedia.com).

What is the relationship between SOA and EdUnify?

EdUnify as a centralized resource enables SOA. EdUnify directs users and organizations to information about Web services and to technical documentation needed to employ the services published. Users can annotate and follow best practices for coding and describing Web services thereby harmonizing like services, customize options and create unique solutions rather than be forced onto a one-size-fits-all application.

EdUnify supports loose coupling of applications to minimize dependencies of fixed resources. Encapsulated services are published in a standard form, making them easier to work with and find. The protocol, agreements and levels of abstraction are not dictated by EdUnify, but promoted to help technical staff employ the services and overcome barriers and scarcity of knowledge often imposed by the technical depth implied by SOA.

With large scale exposure of Web services, developers are provided with many opportunities to leverage services and create larger SOAs. As these architectures grow and merge, EdUnify will have enabled the creation of a massively interconnected architecture that will enhance the day-to-day operations of all members of the education community.