



Introduction

- Software reuse is the process of creating software systems from predefined software components.
- A reusable component may be code, but the bigger benefits of reuse come from a broader and higher-level view of what can be reused.
- Software specifications, designs, tests cases, data, prototypes, plans, documentation, frameworks, and templates are all candidates for reuse.



Need of Software Reuse

- Software products are costly.
- Software project managers are worried about the expensive software development and are desperately find for ways to cut development cost are,
 - A possible way to reduce development costs is to use parts again from previously developed software.
 - In addition to decrease development cost and time, use again also leads to the higher quality of the developed products.



The major advantages for software reuse are :

- Increase software productivity.
- Shorten software development time.
- Improve software system interoperability.
- Develop software with fewer people.
- Move personnel more easily from project to project.
- Reduce software development and maintenance costs.
- Produce more standardized software.
- Produce better quality software and provide a powerful competitive advantage.




Need of Software Reuse

- Software products are costly.
- Software project managers are worried about the expensive software development and are desperately find for ways to cut development cost are,
 - A possible way to reduce development costs is to use parts again from previously developed software.
 - In addition to decrease development cost and time, use again also leads to the higher quality of the developed products.



The major advantages for software reuse are :

- 
- Increase software productivity.
 - Shorten software development time.
 - Improve software system interoperability.
 - Develop software with fewer people.
 - Move personnel more easily from project to project.
 - Reduce software development and maintenance costs.
 - Produce more standardized software.
 - Produce better quality software and provide a powerful competitive advantage.

What is a COTS component:

What is a Commercial Off the shelf (COTS) component:

- ✓ Independent and replaceable part of a system that fulfills a clear function
 - ✓ Works in the context of a well defined architecture
 - ✓ A component communicates with other components by its interfaces
 - ✓ Developed by different developers, using different languages and different platforms
-

Advantages of COTS:

- ✓ Development cost is reduced
 - ✓ Development time is reduced
 - ✓ Complex systems can be built by reusing pre-existing components
 - ✓ Testing effort is reduced
-

Distributed System

A Distributed System in which the Data, Process, and Interface component of information Systems are distributed to multiple locations in a computer network.

In this system, each processor has its local memory. The processors communicate with one another through various communication lines, such as high buses or telephone lines. The implementation of a distributed system is complicated and difficult.

Advantages:

- Resource sharing
- Computation speedup
- Reliability
- Communication



Layers of Distributed System

- **Presentation Layer** is the actual user interface. The inputs are received by this layer and the outputs are presented by this layer.
- **Presentation Logic layer** includes processing required to establish user interface.

Example: Editing input data, formatting output data.

Layers of Distributed System

- **Application Logic Layer** includes all the logic and processing required to support the actual business application.
Example: Calculations.
- **Data Manipulation Layer** includes all the command and logic required to store and retrieve data to and from the database.
- **Data Layer** is actual stored data in the database.

- A **service-oriented architecture (SOA)** is an architectural pattern in computer software design in which application components provide services to other components via a communications protocol, typically over a network.
- The principles of service-orientation are independent of any vendor, product or technology.
- A service is a self-contained unit of functionality, such as retrieving an online bank statement hence a service is a discretely invokable operation.
- Services can be combined to provide the functionality of a large software application.
- SOA makes it easier for software components on computers connected over a network to cooperate.

- **A service-oriented architecture (SOA)** is an architectural pattern in computer software design in which application components provide services to other components via a communications protocol, typically over a network.
- The principles of service-orientation are independent of any vendor, product or technology.
- A service is a self-contained unit of functionality, such as retrieving an online bank statement hence a service is a discretely invokable operation.
- Services can be combined to provide the functionality of a large software application.
- SOA makes it easier for software components on computers connected over a network to cooperate.

Every computer can run any number of services, and each service is built in a way that ensures that the service can exchange information with any other service in the network without human interaction and without the need to make changes to the underlying program itself.



SOA implementations rely on a mesh of software services.

Services comprise unassociated, loosely coupled units of functionality that have no calls to each other embedded in them.

Each service implements one action, such as filling out an online application for an account, or viewing an online bank-statement, or placing an online booking or airline ticket order.

Principles

The following guiding principles define the ground rules for development, maintenance, and usage of the SOA:

- ▶ Reuse, granularity, modularity, compensability, componentization and interoperability.
- ▶ Standards-compliance (both common and industry-specific).
- ▶ Services identification and categorization, provisioning and delivery, and monitoring and tracking.

Attributes of a SOA

In addition, one might take the following factors into account when defining a SOA implementation:

- ▶ Efficient use of system resources
- ▶ Service maturity and performance
- ▶ EAI (Enterprise Application Integration) is defined as the use of software and computer systems architectural principles to integrate a set of enterprise computer applications.

SOA and Web service protocols

- ▶ **Service encapsulation** – Many web 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.

SOA and Web service protocols...

- ▶ Implementers commonly build SOAs using web services standards (for example, SOAP) that have gained broad industry acceptance.
- ▶ These standards (also referred to as Web Service specifications) also provide greater interoperability and some protection from lock-in to proprietary vendor software.
- ▶ One can, however, implement SOA using any service-based technology, such as Jini, CORBA or REST.

SOA and Web Services

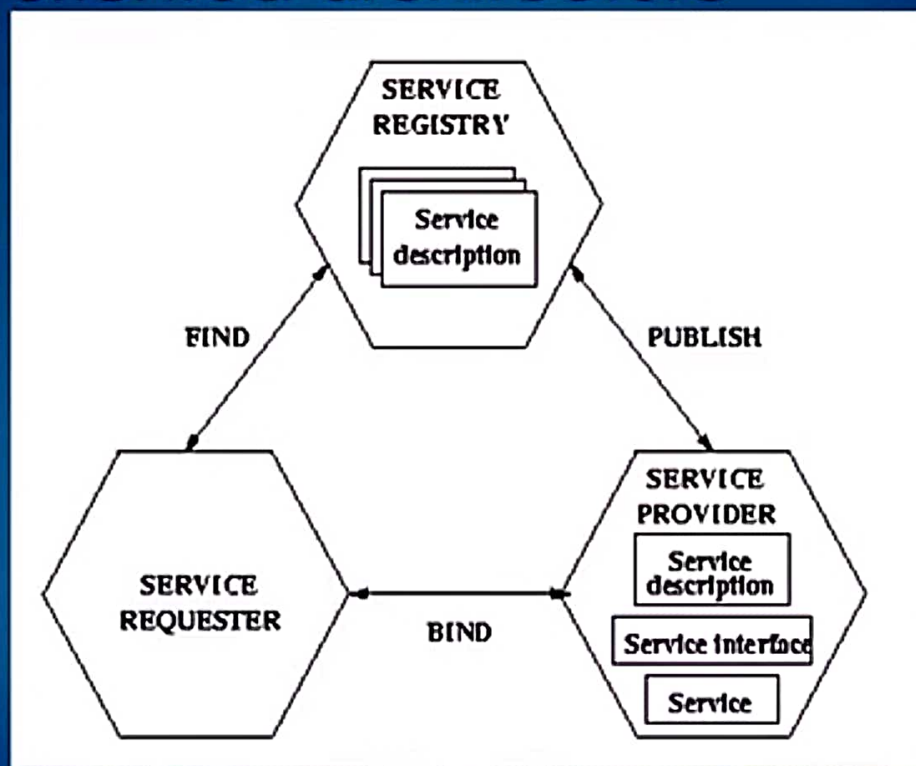
- ▶ Service-oriented architecture is
 - ▶ Derived from the client-server architectural style.
 - ▶ Clients (service consumers or requesters) and servers (service providers) connected by a service "bus".
 - ▶ Services defined using formal interfaces (contracts).
 - ▶ Service bus supports point-to-point and messaging styles of communication.
 - ▶ Support for system qualities, e.g., security and transaction management.
- ▶ Web services
 - ▶ Services provided in a SOA deployed over the web.

Key standards

- ▶ SOAP (Simple Object Access Protocol)
 - ▶ A message exchange standard that supports service communication.
- ▶ WSDL (Web Service Definition Language)
 - ▶ This standard allows a service interface and its bindings to be defined.
- ▶ UDDI (Universal Description Discovery and Integration)
 - ▶ Defines the components of a service specification that may be used to discover the existence of a service.

Service-oriented architecture

10



SOAP (Simple Object Access Protocol)

- ▶ **SOAP**, is a protocol specification for exchanging structured information in the implementation of web services in computer networks.
- ▶ It uses XML Information Set for its message format, and relies on other application layer protocols, most notably Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.

WSDL (Web Service Definition Language)

- ▶ WSDL is an XML-based interface definition language that is used for describing the functionality offered by a web service.
- ▶ The acronym is also used for any specific WSDL description of a web service (also referred to as a WSDL file), which provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns.

UDDI (Universal Description Discovery and Integration)

- UDDI is a platform-independent, Extensible Markup Language protocol includes a (XML)-based registry by which businesses worldwide can list themselves on the Internet, and a mechanism to register and locate web service applications.
- UDDI is an open industry initiative, sponsored by the Organization for the Advancement of Structured Information Standards (OASIS), for enabling businesses to publish service listings and discover each other, and to define how the services or software applications interact over the Internet.