SERVICE ORIENTED

Decomposing problem logic required to solve any large problem into a collection of smaller and related pieces

Coarse grain : Larger functionality
Fine grain : Smaller functions

SOA

It is a model in which the automation logic is decomposed into smaller and distinct units of logic.

To ensure reusability
- Standardized Interface

HISTORY

Mainframe
- Modify is costlier due to high coupling
- Monolithic

Client server
- Mainframe issues resolved
- Presentation logic on client-side
- Server :- Business + Data

Distributed Approach
- Decouple everything (Presentation, Business/App, Database)
- Central part (i.e. Business) is heavy

- DCOM and CORBA
  - uses RMI (Remote Method Invocation)

- SOA

  - Decompose in such a way that it is easily accessible and whenever required (components.

    Level of dependency
    - Not too much

Drawbacks without SOA
- coupled system (sessions need to be maintained)

Why SOA?
- Agility
- Faster delivery

Fundamental units of SOA / SOA principles
- Loosely coupled
  Minimize dependency i.e. call only when required.

- Service contract
  Share a standard interface
  Achieved using WSDL document which is auto-generated (Web Service Description Language)

change in code leads to republishing of

Without this, other services won't know that this service exists.

- Autonomy
  service has control over its logic which further leads to loose coupling

- Abstraction / Encapsulation
  Hide the logic in a standard way so that it doesn't depend on any platform

- Reusability
  It has to be loosely coupled and autonomous

- Composability
  Composing different services to provide desired output
  service contract must be shared
  Abstraction required
  services must be loosely coupled and autonomous

- Statelessness
  If state needs to be maintained, it decreases reusability as it increases coupling.

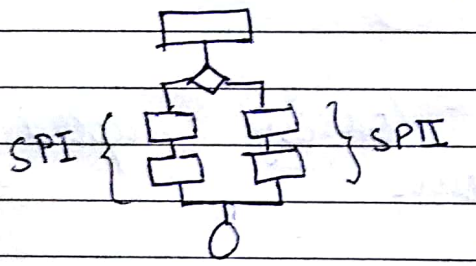
- Discoverability
  share something with others so that communication is possible

Repository where you stored desc

UDDI : helps us achieving discoverability.

SOAP (Simple Object Access Protocol)

- How services encapsulate?
  - Fine – small process step / logic
  - Subprocess
  - Coarse / Entire process



whole flowchart = entire process

SP : Sub-process

▭ : Small process step

- How services relate?
  - Using WSDL document i.e by sharing contract
- How services communicate?
  - Using SOAP
- PRIMITIVE SOA / BASELINE SOA

← Includes following components :
- Services : serve a certain functionality
- Descriptions

- Messages

WS-I (Web service Interoperability)

- 2nd gen / CONTEMPORARY SOA
  - Extensions (WS-*)

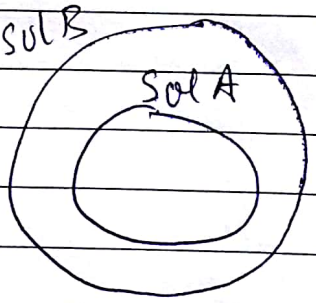
Characteristics
- Increases quality of service
  (i) Reliability of messages
      WS-RM (WS - Reliable Messaging)
      - Intelligent messages : They have their own state and maintain themselves
  (ii) Security
      ws - Security
      ( ) Service level
      Message level
  (iii) Less Overhead

- CSOA is fundamentally autonomous
  - service level autonomy can be achieved only using messaging framework
- CSOA is based on open standards
- CSOA supports vendor diversity
  - Legacy app^n can be joined using wrapper service

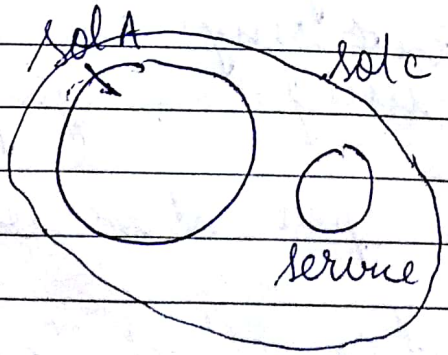
Camlin | Page
Date / /

Members

- CSOA promotes discovery mechanism
  - so that we can have loose coupling across enterprise
- CSOA supports intrinsic (internal) interoperability
- CSOA promotes federation
  joining of components together
- CSOA promotes architectural composability

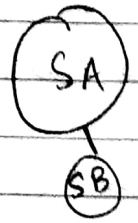


Solution level composability



Service level composability

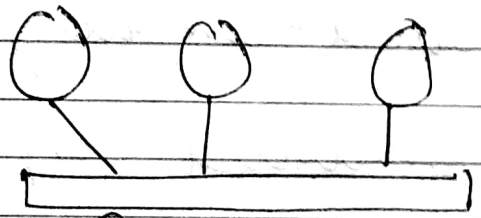
- CSOA encourages reusability
- CSOA emphasizes extensibility



don't break existing service, add the required ones

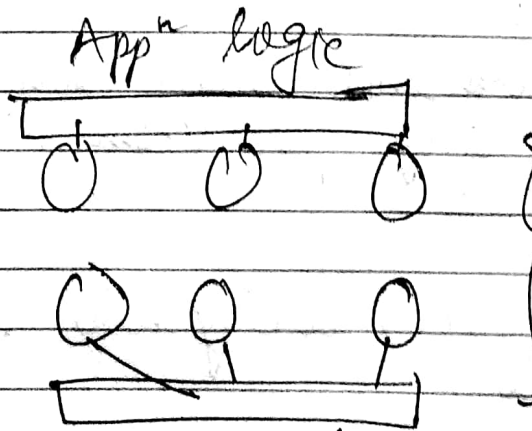
If new service is updated, discription needs to be republished

- CSOA supports business modeling



} Standardized Web services (2)

Business logic — (1) is abstracted using



} Allows loose coupling

- CSOA implements layers of abstraction
- CSOA promotes organizational agility

Camlin Page
Date / /

Pg: 54 Complete SOA definition

- Misperceptions about SOA / False SOA syndromes
  - An application that uses web-services is service oriented
    - Following of standards and service orientation is required
  - SOA is marketing term used to referanding of web-services
  - SOA is marketing term used to referand distributed-computing
  - SOA simplifies distributed computing
  - An application with web services that uses WS-* extension is service oriented
  - If you understand web services you won't have any problem in building SOA
  - Once you go SOA, everything becomes inter-operable

Real
~~True~~ SOA : Following standards and service orientation by removing misperceptions

~~Ideal~~

Ideal SOA : Total decomposition of logic

- Common tangible benefits of SOA

- provides improved integration
- inherent reuse
- streamlined architectures and solutions
  - well-organized / managed
- leveraging the legacy investment
  - take max benefit of anything
- establishing standardized XML representation
- focused investment on communication infrastructure
- SOA gives you best of breed alternative
- Organizational agility
  - response to change
  - changes are easy due to loose coupling and proper design

- Common pitfalls of adopting SOA / Take Cares
- building SOA like traditional distributed architecture
  - Properties of distributed archi :
    - Remote Proc. Call Messages ie Fine grain

- improper partitioning (mixture of fine and coarse)
- non standardizations

- not standardizing SOAs
- not creating transition plan
- not starting with XML based architecture
- not understanding performance requirement
- not understanding web service security
  - transport level and message level security both are required

ARCHITECTURE

Application ⊆ Enterprise
eg: It is blueprint of house
eg: city layout

Client-server ~~v/s~~ SOA

SINGLE TIER

Client	Server
Presentation	Presentation + Business + Database
- thin client	- intelligent
- dumb client	- entire app^n logic
- minimal processing / no app^n logic	- bulk of processing

- Communication possible
  - Synchronous
  - Asynchronous
- single point failure system

TWO TIER

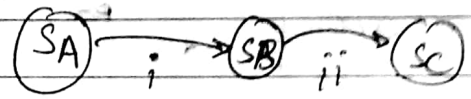
Presentation + Business	Database
- fat client	- database server
- bulk of processing	- data related logic
- bulk of app^n logic	- data related processing

- Communication possible
  - Synchronous

- Database can be upgraded
- High maintenance cost

Client server V/S SOA

	CLIENT	SOA
(i) Appⁿ logic	- It puts bulk of appⁿ logic in client software	- Any software which understands SOAP can become requester

eg: 

SCENARIO i: SB acts as requester
ii: SB acts as provider

	- Client server stores business rules in stored procedures	- Loses the flexibility of storing business logic
	- It has tight coupling	- It has loose coupling
(ii) Appⁿ processing	- Most appⁿ logic is processed by the client (but still bottleneck occurs on server side) eg: if server fails (even though performing less tasks, the system fails)	- provides distributed processing

Camlin	Page
Date	/ /

- Each client requires dedicated database connection and synchronous communication
- Client requires more resources and stateful communication

- The communication can be synchronous or asynchronous as well as / stateless recommended

(iii) Tech.
- 3GL languages
  - C, C++, VB
  - Microsoft Access, Older Oracle
- 4GL and above
  - C#, Java,
  - JSON, Cloud DB

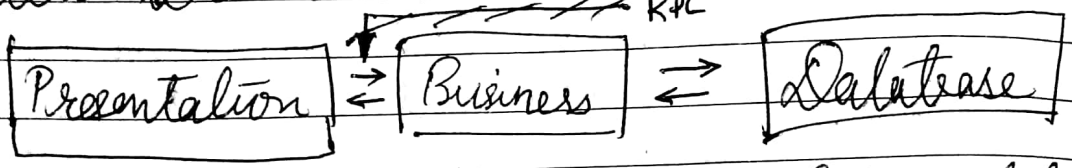
(iv) Security
- Server side / Database level security
- Client side
  - App level
  - OS level

- more complicated security because we don't have seperate client and server
∴ msg level security
- msg level content management

(v) Admin-istration
- High maintenance cost
- Low

Camlin Page
Date / /

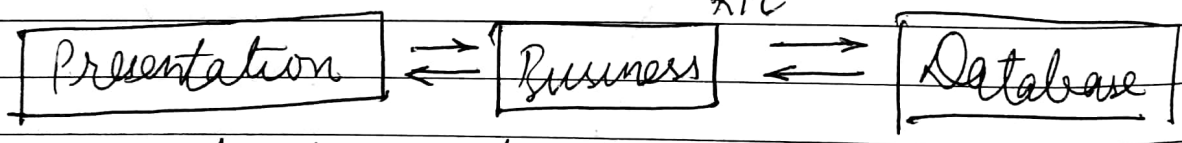
Multi-tier client server



Presentation	Business	Database
- Client - presentation related proces - Semi-intelligent client	- App logic - Component leased logic - Bulk processing - Resource pooling	- Data related logic " " processing

Drawback : Middle layer had too many responsibilities

Distributed - Internet Architecture



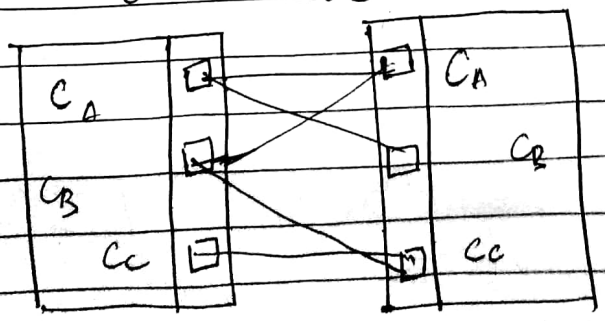
Presentation	Business	Database
- dumb client - browser - no logic - presentation related processing	- App logic - Component based logic - App + Web server - Resource pooling - Bulk processing	- Data related logic and processing

∴ RPC, tight coupling

Camlin Page
Date / /

DIA v/s SOA

	DIA	SOA
(i) Application logic	- logic resides at one/more central servers	- also stored on servers but in form of web services
	- designed with varying degree of ~~multi~~granularity	- designed using proper partitioning using SOA principles
	- same server components comm^{n} via proprietary APIs	- services provide standard interface to any kind of application
	- actual references of components are embedded within the code.	- loose coupling is assured using discovery mechanism
	- RPC is used for intra-server communication	- instead of exchanging parameter data, services use document styled messages



□ - proxy stubs
C_i - Component i

CHARACTERISTICS

(ii) Application processing

- It promotes use of proprietary protocols
  eg: DCOM, CORBA
  owned by someone ∴ we have to rely on their code
- standardized message based comm protocol
- communicate using SOAP parsing
- validation, serialization required

CHARACTERISTICS

- active connection required → do not required due to discovery mechanism
- stateful / stateless communication → stateless
- synchronous data exchange → asynchronous is preferred as we want to achieve loose coupling

(iii) Technology

- Both use HTML, HTTP, XML and web services
- XML and WSP are optional (suggestion v/s compulsion)
- CSOA is built upon XML representation and web service platform

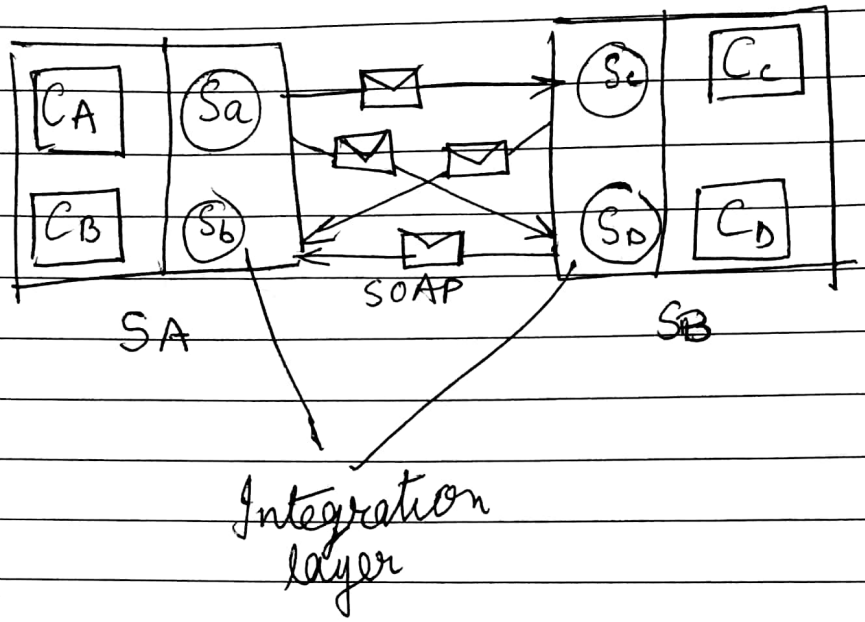
(iv) Security
- Delegation
- Impersonation
- simple encryption
- Standard security codes provided by WS-security standards

Securing component and web services are required in both

(v) Administration
- keep track of all active component references } "
- local and remote problem tracking } "
- monitor the server demands i.e. load balancing } "
- database administration } "
- scalability - expand the solution } "
- take care of maturity of WS* extensions
- resource mgmnt for repository

Date / /

SOA v/s ~~Hyd~~ Hybrid, web service architecture



Relation b/w

- Service orientation and Object orientation

- Service orientation supports composition of loosely coupled logic
- Object orientation supports composition but also encourages inheritance and other relationships which leads to tightly coupled dependencies

- Service orientation is based on designing services.
- Object orientation is focused for creation of objects

- In service orientation units of logic scope can vary
- Object orientation units of logic tend to be smaller in scope

- Service orientation prefers the units of logic to be as stateless as possible
- Object orientation binds data and logic together, hence, it tends to be stateful
- Service orientation emphasizes loose coupling between the units of logic
- Object orientation also supports reusable routines, but internally, they have predefined dependencies

CHAPTER 5 : Web Services & Primitive SOA

Web Services Framework
- Web services
- Service description (WSDL)
- Messaging framework (SOAP)
- Discovery mechanism (UDDI)

Service Classification

When does a service becomes web service

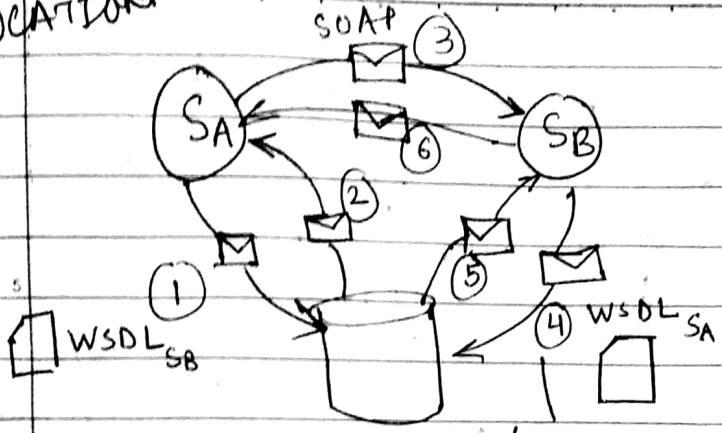
- Temporary (Service can change role based on user requirement)
- Permanent
- both
  - service orientation provision
  - provide business functionality

Temporary : Based on the role services assume at run time

Permanent : Based on the application logic a service provides
- Generic functionality - Utility service
- Controls other services - Controller service
- Provides logic - Business service

- Service provider — Publish service description, Invocation
- Service requestor — Invocation
- Intermediary — Active / Passive — Search proper provider

INVOCATION



- Responsibility of requestor to check for updates from time to time

because it doesn't know S_A's structure

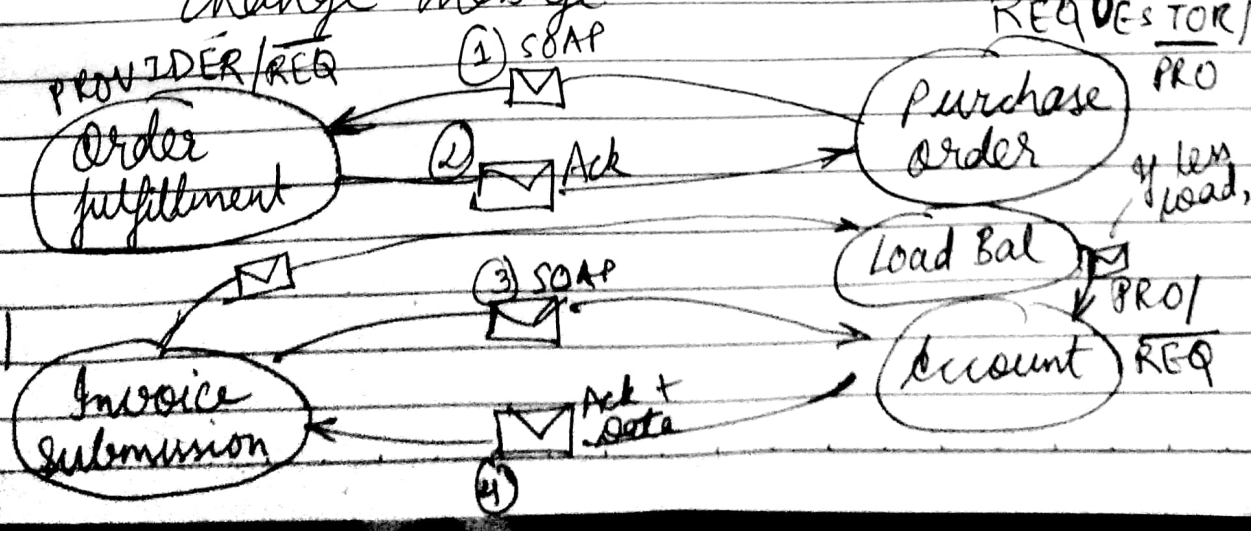
- Every time these steps need not to be performed. Save once in your project and check at regular intervals for updated service

ACTIVE: It is a service which ~~roots~~ routes the message to subsequent location and it can add/remove message contents

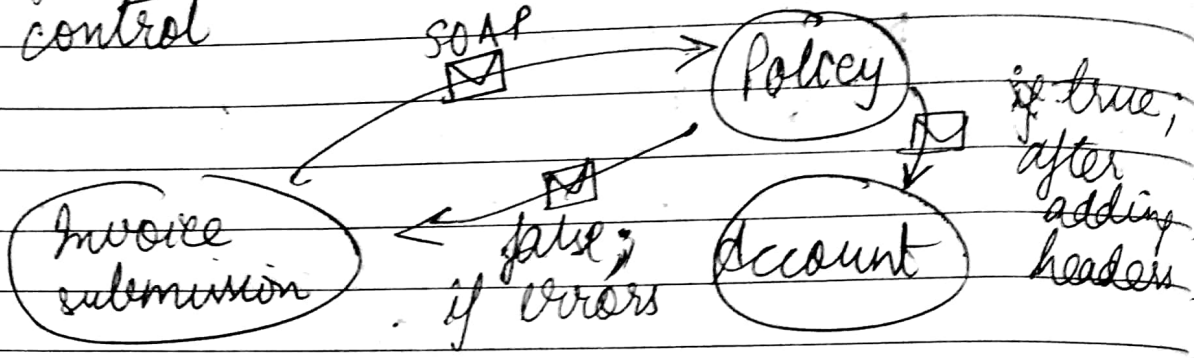
~~PASSIVE: It roots the message to subsequent location~~

PASSIVE: It routes the message to subsequent location and doesn't change message

eg:



Load Balance (PASSIVE INTERMEDIARY)
- Forwards the message if load is in control



Policy (ACTIVE INTERMEDIARY)