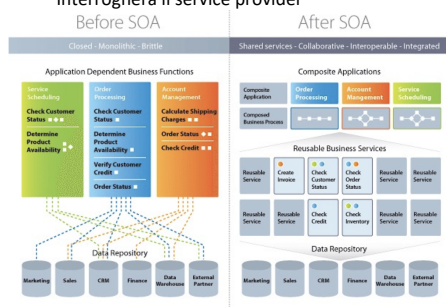


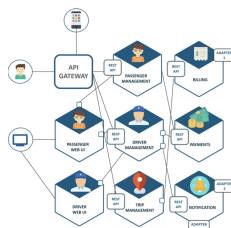
# Service computing

Monday, 3 April 2023 08:48

- IOT (internet of things)
  - Sistema controllato, monitorato oppure interato con dispositivi elettronici che possono comunicare tramite delle interfacce che si possono connettere ad internet
  - Implementano approccio di servizio
    - A distanza offrono delle disponibilità
    - Le cose diventano dei servizi che si comunicano/organizzano per ottenere un risultato
    - Possono essere riutilizzati da altre applicazione ed essere riconfigurate
  - Es: Cellulare, che è una piattaforma che genera contenuti attraverso dei servizi
- Service computing
  - Prima avevamo applicazioni client-server che comunicano con messaggi  
Aka sistemi singoli, e vengono creati con sviluppo top-down (si suddivide in parti, es Interfaccia DB ecc)
    - Risultato = Un applicazione compatta lato server con accoppiamento
    - Però l'accoppiamento si tende ad evitare siccome, ogni modifica avrà un impatto su tutto
  - Ora le applicazioni includono servizi da altre aziende
    - Anche il backend si comporta come se fosse un insieme di servizi
    - Obiettivo di avere una applicazione che è facile da mantenere
    - Nasce architettura orientata ai servizi (SOA)
      - Si focalizza sul riuso di elementi separati chiamati servizi
      - Spacchettamento applicazioni in sezioni atomiche che offrono funzionalità
      - Facendo così, se A e B usano funzionalità C, non dobbiamo scrivere AC e BC  
Ed ogni volta che dobbiamo riscrivere C dobbiamo farlo sia su A che B  
Ma mettiamo in comune C attraverso un servizio
      - I servizi facendo così possono essere sia dentro alla nostra azienda, ma anche disponibile agli altri, e quindi possiamo creare API a pagamento
    - Pro:
      - Riusabilità (I servizi usati da più applicazione)
      - Sviluppo agile ed orientato al business
      - Si possono creare API e quindi venderli
      - Facile la scalabilità
      - Ottimizzazione e -costi
    - Contro
      - Più complesso lo sviluppo e mantenimento (life-cycle management)  
Siccome tutti i servizi devono svilupparsi contemporaneamente senza discrepanze
      - Quindi tanta dipendenza tra di essi
      - E' difficile rendere un qualcosa prima SOA dopo Soa
    - Per esserlo deve:
      - Avere una descrizione con funzionalità
      - Deve essere accessibile tramite rete con protocolli conosciuti
      - Le sue funzionalità devono essere integrati con altri semplicemente
      - Orientato al servizio (quindi risolvere a dei bisogni funzionali-non funzionali)
        - Funzionali = Implementazione di determinate funzioni, cosa viene fornito
        - Non funzionali = Come viene fornito, non cosa, tipo la velocità
    - Componenti:
      - Servizio
      - Descrizione servizi
    - Ruoli:
      - Servizi provider
      - Service broker, chi gestisce i servizi e la documentazione/catalogo
      - Service requestor, che cerca dentro al catalogo, ed una volta trovato Interogherà il service provider



Es.



- Un servizio è un pezzo di software, che può essere scoperto/invocato attraverso la rete attraverso il suo URI  
E le sue interfacce sono ben definite e descritte e note, permettendo iterazioni dirette
  - Se utilizza il protocollo HTTP, è un servizio Web
  - Le interfacce sono gestite da un middleware
  - Le informazioni sono documentate, aka XML, JSON, HTML, ecc
- SLA
  - E' un contratto tra il provider del servizio e chi lo userà
  - Descrive le funzionalità del servizio e le sue qualità

- Qualità = SLO = Requisiti non funzionali  
Es. Response time mean = 0.1s, disponibilità = 99.9%, costo=0.02\$

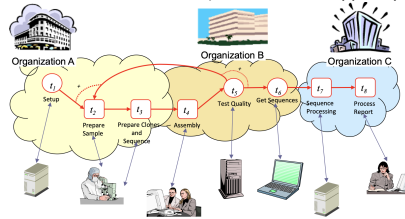
## o WS

Quando siamo nel web:

- Il nome deve essere scoperto, e l'interfaccia pubblica
- I servizi devono essere componibile

Cioè abbiamo 2 o più servizi collegati tra di loro, e che quindi sono compatibili  
E quando abbiamo tanti servizi possiamo decidere di metterli insieme in business process

E questi servizi messi insieme possono essere sia manuali che automatizzati, ma devono essere distribuiti (stessa azienda oppure più aziende)



Però se ci concentriamo solamente sul software, allora abbiamo 2 strategie:

- Orchestrazione

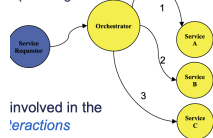
1 attore fa tutto

Ed è infatti più facile da monitorare

Sono molto comuni e facile da comprendere



» (in the right)



involved in the  
interactions

- Coreografia

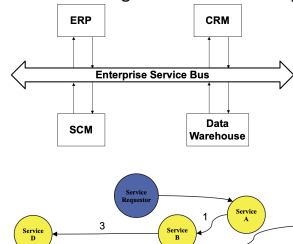
Potrebbe non essere lineare

Viene implementato con ESB, enterprise service bus

Che è praticamente una coda di messaggi a cui i servizi si possono registrare.

Con questo bus è possibile rendere indipendente un servizio dall'altro

E facilita l'integrazione con nuove applicazioni

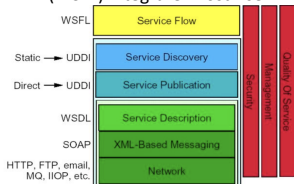


- Devono avere una buona semantica
- Infrastruttura specifica
- Potremmo avere bisogno di servizi di servizi, es:
  - Sicurezza
  - Monitoraggio
- Ha un sistema di organizzazione specifico

## • SOAP services

### o Noi abbiamo:

- A rete i protocolli che permettono comunicazione tra i servizi  
Il più utilizzato è http
- (SOUP) Messaggistica, se sappiamo come, allora dobbiamo definire dove
- (WSDL) Poi dobbiamo fornire gli endpoint, quindi descrivere i servizi
- (UDDI) Per salvare e cercare i nostri servizi
- (WSFL) Integrare i nostri servizi



### o SOAP

- serve per la definizione dei messaggi fra chi invia e chi riceve
- Si basa su XML che ne specificano la semantica
- Definisce soltanto i messaggi, quindi è indifferente come li riceviamo
- Iterazione request-response
- Usato per i servizi orientato ai documenti (formato testuale)
  - In caso contrario, potremmo fare chiamate di procedura remota  
Aka richiediamo di eseguire una funzione per noi
- Struttura:
- Envelope  
Aka Raccoglie il messaggio
  - Header (opzionale)  
Maggiore flessibilità, e possono contenere autenticazione, Routing, contesto  
A differenza dell'header http, questo varia da applicazione ad applicazione  
Quando in HTTP abbiamo dei campi fissi, qui noi li possiamo decidere
  - Body

Il messaggio in sé per sé

- Di solito il protocollo di trasporto utilizzato è l'http  
In questo caso l'header dell'html dovrà essere SOAP+xml

o WSDL

- Linguaggio basato sull'xml che serve per descrivere interfaccia di un servizio
- E' molto preciso, troppo preciso, e si è passato verso il JSON/simili siccome alle aziende

Il servizio la definizione di una API più semplice con oggetti più leggibili

- Nate le linee guida REST
- Definisce:
  - Come usare un servizio
    - Messaggi  
Definiti usando il soap di norma, però potrebbe usare anche altri
    - Operazioni  
Stabilisce qual è il pattern tra 1 client ed 1 server, cioè quali e quanti messaggi scambiati
    - Interfacce  
Raggruppamento di operazioni in maniera indipendente
  - Dove andarlo a chiamare
    - Binding  
A quale indirizzo IP ogni parte astratta è disponibile
    - Servizio
    - Endpoint  
Dove definiamo nome, il binding e poi l'indirizzo