Luca Cabibbo

Architettura dei Sistemi Software

Now on to lezione

Architettura a servizi

dispensa asw510

ottobre 2024

A system is never the sum of its parts.
It is the product
of the interactions of its parts.

Russell L. Ackoff

- Riferimenti

- Luca Cabibbo. **Architettura del Software**: Strutture e Qualità. Edizioni Efesto, 2021.
  - Capitolo 29, Architettura a servizi
- [SAP] Len Bass, Paul Clements, Rick Kazman. **Software Architecture in Practice**. Addison Wesley, fourth edition, 2022
- [POSA4] Buschmann, F., Henney, K., and Schmidt, D.C. **Pattern-Oriented Software Architecture (Volume 4)**: A Pattern Language for Distributed Computing. Wiley, 2007.
- Alonso, G., Casati, F., Kuno, H., and Machiraju, V. **Web Services**: Concepts, Architectures and Applications. Springer, 2004.
- Bass, L., Weber, I., and Zhu, L. **DevOps**: A Software Architect's Perspective. Addison-Wesley, 2015.
- Erl, T. **SOA**: Principles of Service Design. Prentice Hall, 2008.
- Papazoglou, M.P. **Web Services**: Principles and Technology. Pearson, 2008.

Il documento è un'introduzione all'architettura a servizi (Service-Oriented Architecture, SOA), presentata come uno stile architetturale per progettare sistemi software distribuiti. Ecco un riassunto dei punti principali:

Introduzione

- L'architettura a servizi si è sviluppata per supportare gli obiettivi di business delle organizzazioni attraverso sistemi flessibili e interoperabili.
- I sistemi vengono organizzati in servizi software che incapsulano funzionalità di business.
- I servizi possono essere sviluppati con tecnologie diverse e provenire da organizzazioni separate.

Caratteristiche dei servizi

- Interoperabilità: Consente l'integrazione tra sistemi eterogenei tramite protocolli standard del web (ad esempio, SOAP e REST).
- Composizione: I servizi possono essere combinati in nuovi servizi per soddisfare obiettivi di business complessi, con modalità di orchestrazione (mediata) o coreografia (interazione tra pari).
- Autonomia: Ogni servizio è indipendente e governato nel proprio ambiente.

Tecnologie a supporto

- Middleware a servizi facilita interoperabilità e composizione.
- Standard web come SOAP e REST sono centrali per descrivere e invocare i servizi.
- Strumenti come service registry o enterprise service bus aiutano nella scoperta e nella mediazione dei servizi.

Progettazione dei servizi

I servizi devono:

1. Avere un contratto formale (interfaccia standardizzata).
2. Essere incapsulati (l'implementazione è nascosta ai consumatori).
3. Essere debolmente accoppiati, riusabili e componibili.
4. Essere stateless (senza gestione dello stato per interazioni scalabili).
5. Consentire la scoperta e l'accesso dinamico tramite rete.

Applicazioni

- SOA supporta non solo sistemi aziendali, ma anche il cloud computing (es. modelli IaaS, PaaS e SaaS) dove risorse come VM o storage sono offerte come servizi.

Conclusioni

L'architettura a servizi, sostenuta da tecnologie moderne, permette la realizzazione di sistemi distribuiti adattabili e scalabili. Tuttavia, una progettazione efficace richiede linee guida precise che tengano conto delle esigenze di business e delle complessità tecnologiche.

- Obiettivi e argomenti

- Obiettivi
  - introdurre gli aspetti fondamentali relativi ai servizi e all'architettura a servizi
  - presentare alcuni principi per la progettazione dei servizi
- Argomenti
  - introduzione all'architettura a servizi
  - servizi e interoperabilità
  - composizione di servizi
  - architettura a servizi
  - principi per la progettazione dei servizi
  - discussione

* Introduzione all'architettura a servizi

- L'*architettura a servizi* è uno stile architetturale (sorto alla fine degli anni '90) per l'organizzazione di sistemi software distribuiti
  - l'obiettivo generale è sostenere la costruzione di sistemi software in grado di soddisfare gli obiettivi di business, correnti e futuri, delle organizzazioni
  - esistono diversi tipi specifici di "architetture a servizi" – qui ci concentriamo sulle idee comuni fondamentali

Insieme di
In grado di soddisfare obiettivi di business correnti e futuro più delle organizzazioni quindi non più interesse per qualità tecnologiche ma anche economiche
Nell'architettura a servizi il sistema software è realizzato come un insieme di servizi software
Gli obiettivi rappresentano funzionalità di business dell'applicaizonne quindi cerchiamo di strutturare il software COME è strutturata l'attività di businesss. I servizi devono derivare da qualche strutturazione di dominio.
I servizi possono essere realizzati con tecnologie diverse (no vincoli mono tecnologici dell'architettura a componenti) e. I servizi possono anche essere presi da organizzazioni diverse.
Per far funzionare tutto questo serve un supporto anche tecnologico per sostenere interoperabilità (sia tecnologica che dal punto di vista dell'organizzazione) e componibilità (serve un meccanismo flessibile di composizione di servizi in modo da offrire nuovi servizi, nuovi obiettivi di business…)

Introduzione all'architettura a servizi

- L'approccio generale dell'architettura a servizi
  - per realizzare il sistema software di un'organizzazione, al fine di sostenere il business dell'organizzazione
    - il sistema software viene strutturato in termini di "servizi software" – questo viene fatto facendo riferimento alla strutturazione del business dell'organizzazione
    - i processi di business dell'organizzazione vengono realizzati come un'integrazione flessibile e agile di questi servizi
    - è anche possibile fruire dei servizi di altre organizzazioni, e offrire i propri servizi ad altre organizzazioni

Architettura a servizi

- L'*architettura a servizi* organizza un sistema software come la composizione di elementi architetturali chiamati "servizi"
  - ogni *servizio* incapsula una ben precisa funzionalità di business di un'organizzazione
  - i servizi di interesse possono essere offerti anche da più sistemi software diversi – che possono anche appartenere ad organizzazioni separate
  - la componibilità e l'interoperabilità dei servizi rivestono un ruolo fondamentale

Tecnologie a servizi

- Le *tecnologie a servizi* (*middleware a servizi*) hanno lo scopo di fornire le capacità utili per la realizzazione dell'architettura a servizi
  - l'*interoperabilità* dei servizi – la possibilità di invocare i servizi indipendentemente dalla loro implementazione
    - sulla base di protocolli standard per il web
  - la *composizione* dei servizi – la capacità di definire nuovi servizi sulla base dell'invocazione di altri servizi
    - per comporre i servizi in modo flessibile
- Le tecnologie a servizi sono complementari alle tecnologie a componenti
  - non hanno l'obiettivo di sostituirle o di soppiantarle
  - molte tecnologie a componenti moderne supportano oggi anche l'interoperabilità e i servizi

Servizi e architettura a servizi

- Definizioni preliminari
  - un *servizio* è un elemento software che incapsula una funzionalità o un "servizio di business" di un'organizzazione
    - per svolgere un compito, risolvere un problema, o condurre transazioni per conto di un utente o di un'applicazione
  - l'*architettura a servizi* organizza un sistema software come la composizione di un insieme di servizi interoperabili
    - questi servizi possono essere offerti anche da sistemi software diversi, di organizzazioni separate e realizzati con tecnologie differenti
    - un'organizzazione può anche fruire delle capacità e dei servizi offerti da altri sistemi e da altre organizzazioni – allo stesso modo, può offrire i propri servizi ad altre organizzazioni

- Servizi e interoperabilità

- Una caratteristica fondamentale delle tecnologie a servizi è l'interoperabilità – tra sistemi diversi, anche di organizzazioni separate
  - tra questi sistemi sussistono in genere diverse forme di eterogeneità (tecnologiche, ma non solo)
    - come consentire le interazione tra sistemi diversi, anche malgrado queste eterogeneità?
  - le tecnologie a servizi offrono delle soluzioni concrete per l'interoperabilità, sulla base di opportuni protocolli standard del web
    - per questo, si parla anche di *servizi web* (*web service*)

Dal web ai servizi web

- Non si confondano i servizi web con il web
  - il web consente l'accesso a dati/servizi tramite pagine web
    - l'accesso al web è pensato per le persone – ma è difficoltoso per i client software
  - i servizi (servizi web) hanno invece l'obiettivo di fornire l'accesso a servizi applicativi a dei client software
    - mediante l'adozione di protocolli standard del web
    - per sostenere l'interoperabilità tra componenti e applicazioni software

- Interoperabilità

- **Interoperabilità** (*interoperability*)
  - il grado in cui due o più sistemi possono interagire in modo effettivo, scambiando tra loro informazioni significative, tramite interfacce, in un contesto particolare

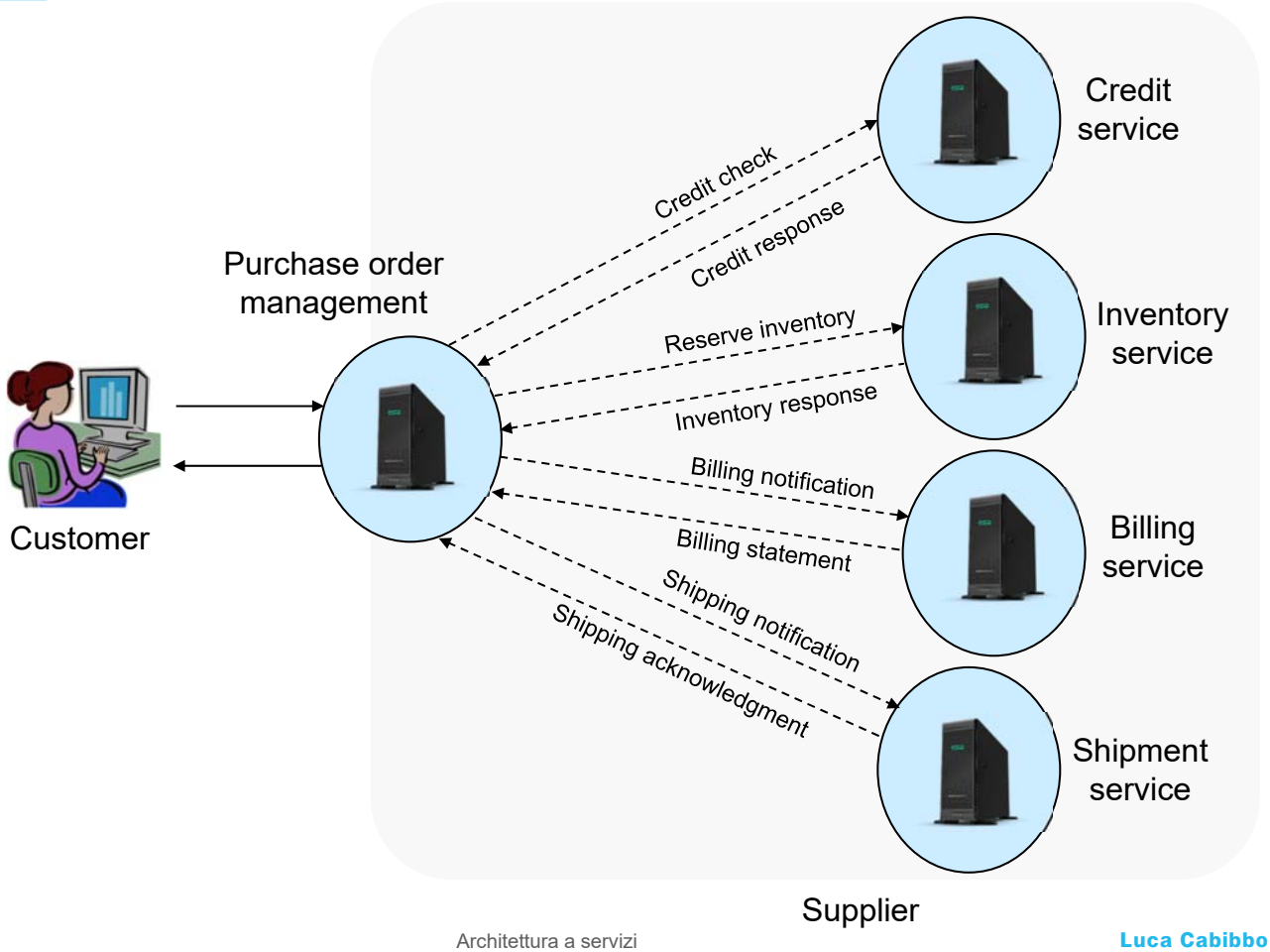
Interoperabilità

- Alcune tattiche principali per l'interoperabilità
  - *interfacce dei servizi*
    - le interazioni tra sistemi devono avvenire tramite delle interfacce, specificate in modo neutrale
  - *gestione delle richieste e delle risposte*
    - deve essere effettivamente possibile l'invocazione dei servizi, sulla base sulle loro interfacce
  - *scoperta dei servizi*
    - i consumatori dei servizi devono poter scoprire identità, locazione e interfaccia dei servizi di interesse
  - *aderire a standard*
    - la standardizzazione è uno degli abilitatori principali dell'interoperabilità e dell'integrabilità – alcuni si focalizzano solo sulla sintassi e la semantica dei dati, mentre altri includono descrizioni più ricche

- Composizione di servizi

- Un'altra caratteristica fondamentale dell'architettura a servizi è la possibilità di comporre servizi in modo flessibile
  - la *composizione* consente di definire un nuovo servizio ("servizio composto") a partire da un insieme di altri servizi ("servizi componenti")
    - la composizione si basa sull'"invocazione" dei servizi componenti, mediante le loro interfacce
  - la flessibilità è maggiore rispetto alla composizione di componenti
    - può riguardare servizi di più sistemi o organizzazioni
    - deve poter avvenire sia staticamente (durante lo sviluppo) che dinamicamente (a runtime)
    - la composizione di servizi deve dare luogo a un nuovo servizio

Un esempio – gestione di ordini d'acquisto



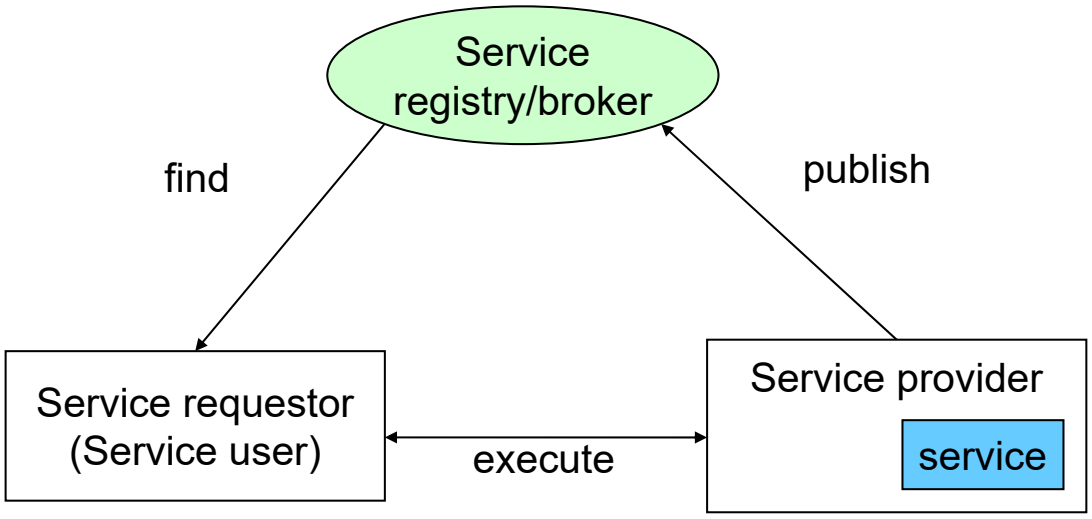
- Capacità fondamentali dei servizi

- Ecco le capacità fondamentali offerte dalle tecnologie a servizi
  - interoperabilità
    - descrizione di servizi
    - invocazione di servizi
    - scoperta di servizi
  - composizione
    - per definire nuovi servizi composti, in modo flessibile
- Queste capacità
  - sono motivate dagli obiettivi dell'architettura a servizi
  - sono realizzate mediante degli opportuni standard basati sul web

- Architettura a servizi (di base)

- Una descrizione minimale dell'architettura a servizi
  - un *servizio* è un componente o un'applicazione software in grado di erogare un certo insieme di funzionalità di business
  - l'*interfaccia di un servizio* è la descrizione delle funzionalità del servizio – sulla base di uno standard aperto del web
  - il *fornitore* di un servizio è un sistema o un'organizzazione che rende il servizio accessibile – mediante la sua interfaccia e tramite tecnologie web standard
  - un *client* (o *consumatore*) di un servizio è un sistema o un'organizzazione che fruisce del servizio – mediante la sua interfaccia e tramite tecnologie web standard
  - un *service registry* (o *service broker* o *service bus*) è un servizio per supportare la scoperta dei servizi oppure per mediare l'invocazione dei servizi

Architettura a servizi (di base)



- Note terminologiche

- Alcuni modi comuni di utilizzare il termine “servizio”
  - nello stile architetturale client-server
    - un servizio rappresenta un insieme di funzionalità, un server è un componente che eroga il servizio, e un client è un componente che fruisce del servizio
    - in pratica, un servizio è un’interfaccia, che è implementata da un server
  - nell’architettura a servizi
    - un servizio è un componente o un’applicazione che eroga un certo insieme di funzionalità, l’interfaccia del servizio è la descrizione di queste funzionalità, e un client è un componente o un’applicazione vuole fruire del servizio
    - in pratica, un servizio è l’implementazione di un’interfaccia di un servizio

* Servizi e interoperabilità

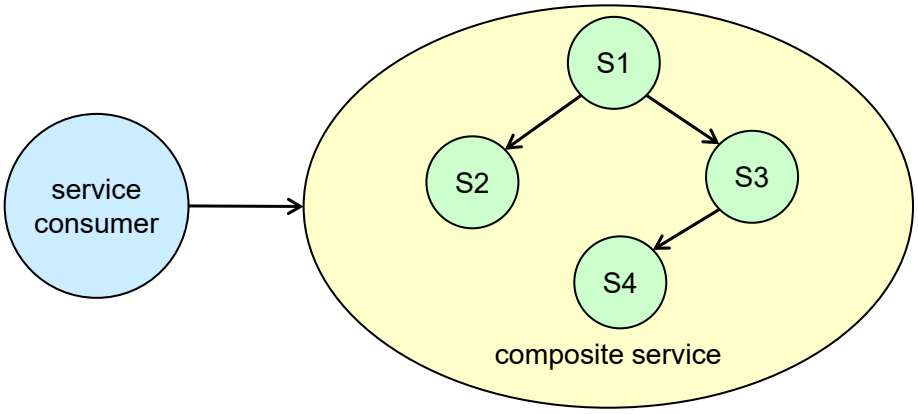
- L'interoperabilità tra componenti software in esecuzione su piattaforme diverse è una caratteristica essenziale dei servizi
  - non era supportata (almeno inizialmente) dalle tecnologie a componenti
  - è invece una capacità fondamentale delle tecnologie a servizi
    - queste tecnologie offrono delle soluzioni concrete a problemi pragmatici di interoperabilità
    - i grandi produttori di software che realizzano queste tecnologie hanno accettato la natura eterogenea delle soluzioni IT delle organizzazioni attuali, e hanno compreso che questa eterogeneità non sarebbe diminuita nel futuro

Interoperabilità e standard

- Le tecnologie a servizi sono basate su un insieme di standard tecnologici per l'interoperabilità – indipendenti dalle piattaforme e neutrali rispetto ad essi
  - alcune proposte iniziali (come CORBA) non sono mai state accettate in modo diffuso
  - viceversa, alcuni standard per servizi più recenti sono stati accettati e sostenuti dalla maggior parte dei produttori di software
    - soprattutto SOAP e REST – per lo meno, gli standard fondamentali
  - SOAP e REST sono due famiglie principali di standard per i servizi, accettati in modo diffuso
  - i servizi SOAP e REST sono descritti in un capitolo successivo

* Composizione di servizi

- La possibilità di definire servizi come *composizione* di altri servizi è un'altra caratteristica fondamentale dei servizi



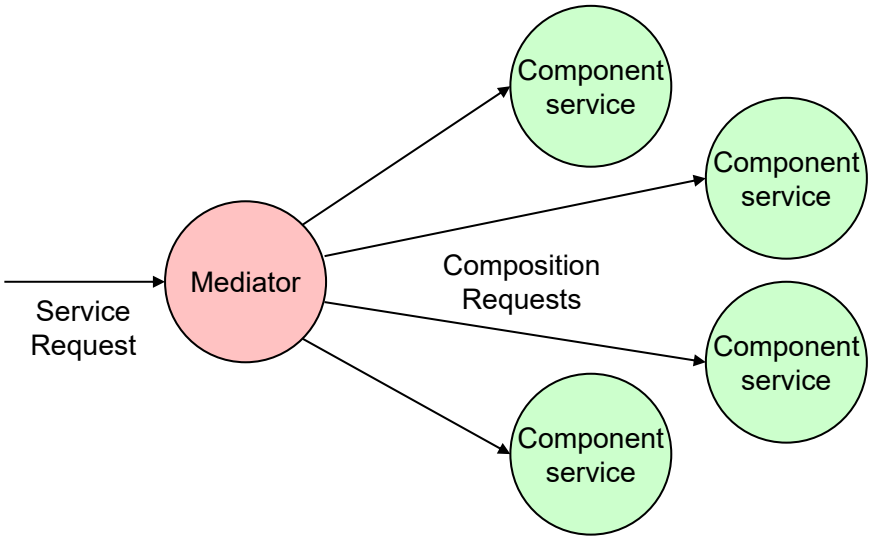
  - ad es., un servizio di prenotazione di viaggi organizzati può essere definito sulla base di altri servizi di prenotazione alberghiera, aerea, noleggio automobili, ...

Servizi composti e servizi componenti

- La composizione di servizi consente di definire un nuovo servizio composto a partire da un insieme di servizi componenti, in modo flessibile
  - la composizione è basata sull'"invocazione" dei servizi componenti
  - un servizio composto può rappresentare un intero processo di business (business process) – ottenuto come composizione di servizi che rappresentano dei compiti o delle attività di business, che devono essere svolti in un ordine appropriato

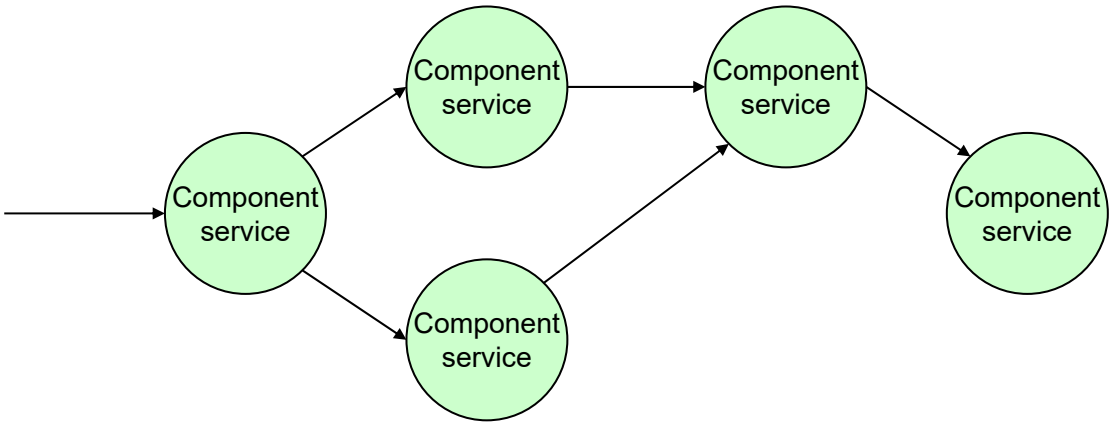
Modalità di composizione

- Due modalità principali di composizione di servizi
  - *orchestrazione* – basata sull'uso di un mediatore



Modalità di composizione

- Due modalità principali di composizione di servizi
  - *coreografia* – basata su interazioni tra pari, senza un coordinamento centralizzato



Orchestrazione e coreografia a confronto

- Supponiamo che, quando un nuovo cliente si registra in un sistema, sia necessario inviare un'email di conferma al cliente e creare un record per i punti fedeltà del cliente
  - nell'orchestrazione, il servizio di registrazione del cliente, dopo aver registrato il cliente, chiede al servizio di posta elettronica di inviare un messaggio al cliente e chiede al servizio dei punti fedeltà di creare un record per il cliente
  - nella coreografia, il servizio di registrazione del cliente, dopo aver registrato il cliente, pubblica una notifica dell'evento di registrazione del nuovo cliente – gli altri due servizi (quando riceveranno la notifica del nuovo cliente) faranno ciò che serve
  - come gestire il caso in cui ai nuovi clienti sia necessario anche inviare un regalo di benvenuto?

Discussione

- Ulteriori osservazioni sulla composizione di servizi
  - un servizio componente può anche essere utilizzato nella composizione di più servizi
  - un servizio composto può anche essere utilizzato come servizio componente nella composizione di altri servizi

* Architettura a servizi

- L'*architettura a servizi* (*service architecture*) è uno stile architetturale per la costruzione di sistemi o applicazioni software distribuite sulla base della composizione di un insieme di servizi
  - esistono diverse descrizioni dell'architettura a servizi come pattern architetturale
    - ne proponiamo una piuttosto generale [SAP]
  - inoltre esistono diversi tipi specifici di "architetture a servizi"
    - nei capitoli successivi presenteremo alcuni importanti casi specifici dell'architettura a servizi

Architettura a servizi

- Contesto
  - ci sono diversi servizi – offerti da vari fornitori di servizi e di interesse per vari consumatori di servizi
  - i consumatori di servizi vogliono poter comprendere e usare questi servizi – ma senza nessuna conoscenza dettagliata delle loro implementazioni
- Problema
  - si vuole sostenere l'interoperabilità di componenti e servizi distribuiti – scritti con linguaggi di programmazione diversi, in esecuzione su piattaforme differenti, forniti da diverse organizzazioni e distribuiti su Internet
  - si vuole sostenere la scoperta di questi servizi – per poterli poi combinare per realizzare delle interazioni significative
  - si vogliono ottenere livelli di prestazioni, sicurezza e disponibilità accettabili

Architettura a servizi

- Soluzione
  - organizza il sistema in termini di un insieme di componenti distribuiti che forniscono e/o consumano *servizi*
    - i componenti fornitori di servizi e i componenti consumatori di servizi possono usare linguaggi di programmazione e piattaforme differenti
    - i servizi sono in larga misura indipendenti – sono rilasciati indipendentemente, e spesso appartengono a sistemi e organizzazioni differenti
  - i componenti hanno interfacce che descrivono i servizi che forniscono o che richiedono a/da altri componenti
  - è anche possibile specificare attributi di qualità dei servizi – e fornire garanzie sui servizi tramite SLA
  - i componenti effettuano le loro computazioni richiedendo servizi gli uni agli altri

Architettura a servizi

- Alcune ulteriori osservazioni
  - i componenti software che forniscono e consumano servizi possono assumere forme molto differenti tra loro
  - oltre a questi componenti, un'architettura a servizi comprende in genere degli ulteriori servizi infrastrutturali – ad esempio
    - un registry dei servizi – per la scoperta dei servizi
    - un enterprise service bus – per l'invocazione dei servizi
    - un server per la composizione e l'orchestrazione di servizi
  - il collegamento tra servizi può avvenire sulla base di diversi tipi di connettori
    - ad es., SOAP, REST oppure l'uso di messaggi asincroni

Discussione

- Questa descrizione proposta da [SAP] è poco soddisfacente – perché cattura solo alcune delle problematiche che devono essere affrontate dall'architettura a servizi
  - si concentra soprattutto sugli aspetti più tecnologici – come l'interoperabilità e la composizione di servizi
  - non coglie invece numerosi degli aspetti, concreti e complessi, che vanno affrontati nei problemi di interoperabilità e di integrazione – come il contesto di business o le linee guida per la progettazione dei servizi e delle loro interfacce
  - inoltre, l'adozione dell'architettura a servizi è sostenuta dalle tecnologie a servizi – ma l'utilizzo di una tecnologia a servizi non garantisce la realizzazione di un'architettura a servizi efficace

* Principi per la progettazione dei servizi

Un aspetto importante è sempre quello metodologico. Cioè abbiamo detto che l'applicazione software è divisa in più servizi: ma QUALI CARATTERISTICHE devono avere questi servizi. Servono linee guida, o meglio principi

- Una questione fondamentale nell'architettura a servizi è la seguente
  - come progettare i servizi – che sono il costrutto fondamentale dell'architettura a servizi – al fine di sostenere gli obiettivi di business di questo stile architetturale?
  - esistono diversi principi che guidano la progettazione dei servizi [Erl 2008]
    - questi principi hanno una valenza generale e possono essere applicati in ogni architettura a servizi – anche se talvolta con interpretazioni differenti
  - i principi fondamentali riguardano
    - contratto formale (interfaccia)
    - astrazione dei servizi (incapsulamento)
    - autonomia dei servizi
    - accoppiamento debole, riusabilità, componibilità

L'idea di base è che debba essere sostenuta la componibilità flessibile un insieme di servizi provenienti dalla mia e da altre organizzazioni

Principi per la progettazione dei servizi

- Principi per la progettazione dei servizi
  - *i servizi condividono un contratto formale (interfaccia)*
    - ciascun servizio deve condividere la propria interfaccia, per descrivere in modo neutrale ogni informazione di interesse per interagire con il servizio

In modo contrattuale, quindi in termini di pre e post condizioni, in termini di quando costa quel servizio, ecc

  - *i servizi realizzano un'astrazione della logica sottostante (incapsulamento)*
    - l'interfaccia è l'unica parte del servizio esposta al suo esterno
    - l'implementazione del servizio deve essere invisibile e irrilevante per i consumatori del servizio

Principi per la progettazione dei servizi

- Principi per la progettazione dei servizi
  - *i servizi possono essere scoperti*
    - i servizi e le loro interfacce devono poter essere scoperti da chi vuole poter utilizzare dei servizi di interesse
  - *i servizi hanno un'interfaccia accessibile in rete*
    - i consumatori di un servizio devono poter effettivamente invocare il servizio in rete, in modo remoto, sulla base di protocolli standard
  - *la locazione dei servizi è trasparente*
    - i consumatori di un servizio devono poter scoprire dinamicamente la posizione del servizio, tramite un registry dei servizi

Principi per la progettazione dei servizi

- Principi per la progettazione dei servizi
  - *i servizi sono debolmente accoppiati*
    - i servizi devono essere progettati per interagire in modo debolmente accoppiato
  - *i servizi sono riusabili*
    - i servizi devono essere progettati per sostenere un potenziale riuso
  - *i servizi sono componibili*
    - i servizi devono poter essere utilizzati per comporre altri servizi
    - questo sostiene la riusabilità, la creazione di livelli di astrazione, e la possibilità di creare e modificare rapidamente applicazioni e processi di business

Principi per la progettazione dei servizi

- Principi per la progettazione dei servizi
  - *i servizi sono autonomi*
    - l'autonomia di un servizio riguarda la sua implementazione e il suo ambiente di esecuzione
    - ogni servizio deve avere un'autonomia di governo completa entro il suo ambiente di esecuzione
  - *i servizi sono stateless*

    Qui il motivo non è sostenere la scalabilità ma per mantenere un accoppiamento basso e sostenere riusabilità e componibilità
    - i servizi non devono gestire informazioni sullo stato delle conversazioni con i loro client
    - questo avrebbe infatti un impatto negativo sul loro accoppiamento, la loro riusabilità e la loro componibilità

Una caratterizzazione alternativa

- Un memo di Jeff Bezos indirizzato al team di Amazon [circa 2002]
  - All teams will henceforth expose their data and functionality through service interfaces.
  - Teams must communicate with each other through these interfaces.
  - There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.
  - It doesn't matter what technology they use. HTTP, Corba, pub/sub, custom protocols – doesn't matter. Bezos doesn't care.
  - All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.
  - Anyone who doesn't do this will be fired.
  - Thank you; have a nice day!

-Cloud computing e servizi

- La nozione di *servizio* riveste un ruolo centrale nel cloud computing – che consente l'accesso a risorse (di varia natura) sotto forma di servizi
  - tre modelli di servizio principali – IaaS, PaaS e SaaS – e talvolta si parla anche di *XaaS* (*everything-as-a-service*)
  - ma sono davvero dei servizi?
    - sì, perché le diverse risorse vengono fornite agli utenti del cloud (i consumatori) in una modalità che soddisfa i principi per i servizi che abbiamo appena discusso

Cloud computing e servizi

- Ad esempio, nel modello *IaaS* (*Infrastructure as a Service*)
  - il consumatore può utilizzare diversi servizi infrastrutturali – come VM, storage e reti
  - il consumatore accede a questi servizi in rete, mediante una loro interfaccia contrattuale (ad es., il tipo di AMI) e in modo astratto dalla loro implementazione
  - il consumatore effettua il provisioning e gestisce i servizi mediante un'API REST (un'interfaccia a servizi)
  - il consumatore può comporre questi servizi, per definire degli ambienti di esecuzione complessi
  - il fornitore dei servizi ha autonomia sull'implementazione e sulla locazione dei servizi
- Considerazioni analoghe possono essere fatte anche per i modelli PaaS e SaaS

* Discussione

- Abbiamo introdotto alcuni aspetti di base dell'architettura a servizi
  - l'architettura a servizi ha lo scopo di sostenere gli obiettivi di business, correnti e futuri, delle organizzazioni
    - esistono diversi importanti casi specifici dell'architettura a servizi – che discuteremo in capitoli successivi
  - l'architettura a servizi è sostenuta dalle tecnologie a servizi
    - che supportano l'interoperabilità tra servizi e la composizione di servizi, sulla base di diversi standard per servizi – che sono discussi nel capitolo successivo
    - queste tecnologie integrano le tecnologie a componenti – gli forniscono delle capacità complementari e aggiungono così valore alle piattaforme di middleware esistenti
  - l'adozione dell'architettura a servizi è sostenuta dalle tecnologie a servizi – ma l'utilizzo di una tecnologia a servizi, da sola, non garantisce la realizzazione di un'architettura a servizi efficace