Luca Cabibbo

Architettura dei Sistemi Software

Broker

dispensa asw440

marzo 2021

Intelligence is not the ability to store information, but to know where to find it.

Albert Einstein

- Riferimenti

- Luca Cabibbo. **Architettura del Software**: Strutture e Qualità. Edizioni Efesto, 2021.
  - Capitolo 24, Broker
- [POSA1] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M. **Pattern-Oriented Software Architecture**: A System of Pattern, Volume 1 (**POSA1**), Wiley, 1996.
- [POSA4] Frank Buschmann, Kevlin Henney, Douglas C. Schmidt. **Pattern-Oriented Software Architecture (vol. 4)**: A Pattern Language for Distributed Computing. John Wiley & Sons, 2007
- Bachmann, F., Bass, L., and Nord, R. Modifiability Tactics. Technical report CMU/SEI-2007-TR-002. 2007.

- Obiettivi e argomenti

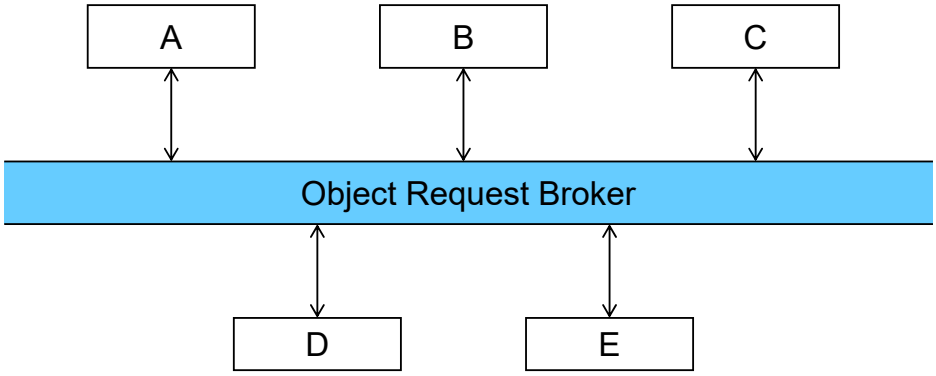
- Obiettivi
  - presentare il pattern architetturale Broker
- Argomenti
  - introduzione
  - Broker [POSA]
  - discussione

* Introduzione

- In un sistema distribuito ci possono essere una molteplicità di componenti server che erogano dei servizi
  - possono essere più istanze/repliche di componenti che erogano uno stesso servizio – oppure anche componenti relativi a servizi diversi
  - la locazione in rete di questi componenti potrebbe anche variare dinamicamente nel tempo
  - è utile un meccanismo per fornire flessibilità e trasparenza rispetto alla locazione dei servizi in rete

Broker

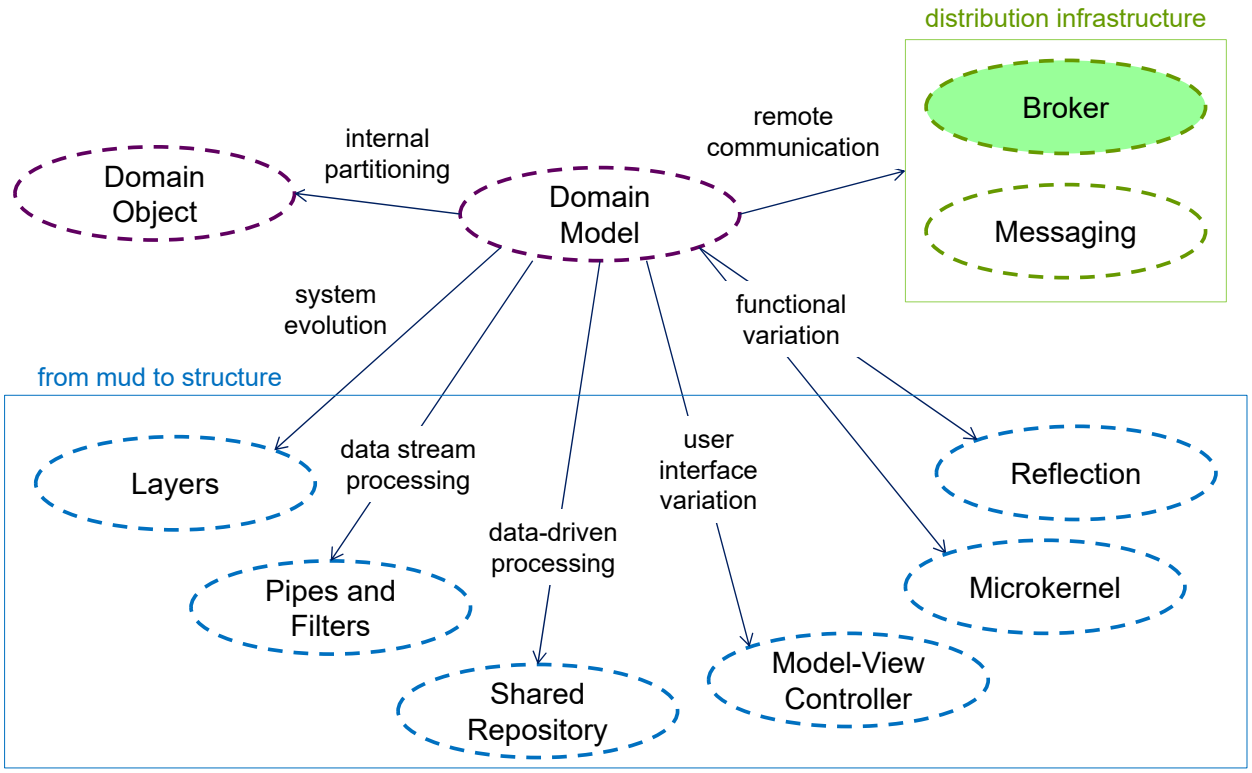
- Una prima soluzione è stata realizzata nelle tecnologie a *oggetti distribuiti* (fine anni ‘80 e primi anni ‘90)
  - la comunicazione tra oggetti distribuiti è supportata da un *object request broker* (*ORB*) – o semplicemente *broker*
  - il broker è essenzialmente un bus software che realizza un’infrastruttura di comunicazione tra gli oggetti distribuiti



* Broker [POSA]

- *Broker* è un pattern architetturale fondamentale POSA, della categoria “distribution infrastructure”
  - supporta lo stile di comunicazione dell’invocazione remota
  - fornisce un’infrastruttura di comunicazione per rendere trasparenti alcune complessità della distribuzione
  - è uno dei contributi principali delle tecnologie a oggetti distribuiti
  - ha ancora oggi un ruolo fondamentale nei sistemi distribuiti – nelle sue diverse varianti ed evoluzioni

Relazione con altri pattern [POSA]



Broker [POSA]

- Il termine *broker* indica, in generale, un *intermediario*
  - "un professionista che ricerca e acquista, per conto del cliente, nel mercato di riferimento, il prodotto che offre il miglior rapporto qualità-prezzo" [Wikipedia]
- Il pattern architetturale **Broker** [POSA]
  - un broker è un intermediario per coordinare la comunicazione tra diversi componenti remoti
  - consente di strutturare sistemi distribuiti con componenti disaccoppiati che interagiscono mediante l'invocazione di servizi remoti

Esempio

- City Information System – CIS
  - sistema di informazioni turistiche
  - portale verso altri sistemi (esterni) che effettivamente offrono servizi per turisti
    - informazioni su alberghi e ristoranti, trasporti pubblici, musei, visite guidate, ...
    - anche con la possibilità di fare prenotazioni/acquisti
  - ci possono essere più sistemi esterni che possono soddisfare uno stesso tipo di richieste
  - è possibile la registrazione dinamica di nuovi sistemi esterni
  - il CIS si propone come un punto di contatto singolo per il turista nei confronti dei sistemi esterni
    - ovvero, come un “broker” tra turista e sistemi esterni

Broker

- Contesto
  - un sistema distribuito, in cui ci sono più componenti che erogano dei servizi
  - è necessaria un’infrastruttura di comunicazione per proteggere le applicazioni dalla complessità della distribuzione

Broker

- Problema
  - si vuole organizzare un sistema distribuito, con più componenti distribuiti, in modo flessibile
    - i componenti dovrebbero poter interagire mediante l'invocazione dei servizi di loro interesse – esprimendo queste invocazioni in modo unificato e indipendente dalla posizione dei componenti
  - in particolare, si desiderano
    - componenti che possono interagire, ma disaccoppiati
    - trasparenza nell'accesso ai componenti – indipendenza dalla locazione, dai meccanismi di comunicazione interprocesso e dalla disponibilità dei componenti
    - possibilità di aggiungere, rimuovere o sostituire componenti a runtime
    - se possibile, interoperabilità tra componenti eterogenei

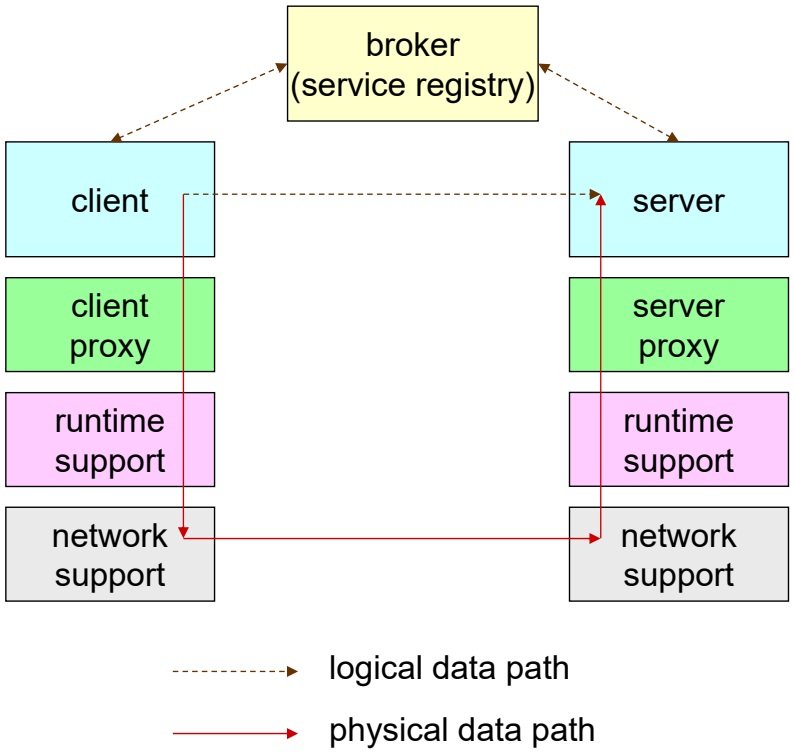
Broker

- Soluzione (struttura)
  - organizza il sistema distribuito come un insieme di componenti che interagiscono mediante invocazioni remote
  - introduci un componente intermediario *broker* per gestire la comunicazione tra questi componenti distribuiti
    - il broker definisce un modello di programmazione distribuita e incapsula l'infrastruttura di comunicazione del sistema distribuito
    - il broker realizza un disaccoppiamento tra utenti (client) e fornitori (server) dei servizi – inoltre sostiene la separazione tra le funzionalità applicative e i dettagli della comunicazione
  - utilizza degli ulteriori intermediari *proxy* – per aiutare i componenti client e server nell'interazione con il broker

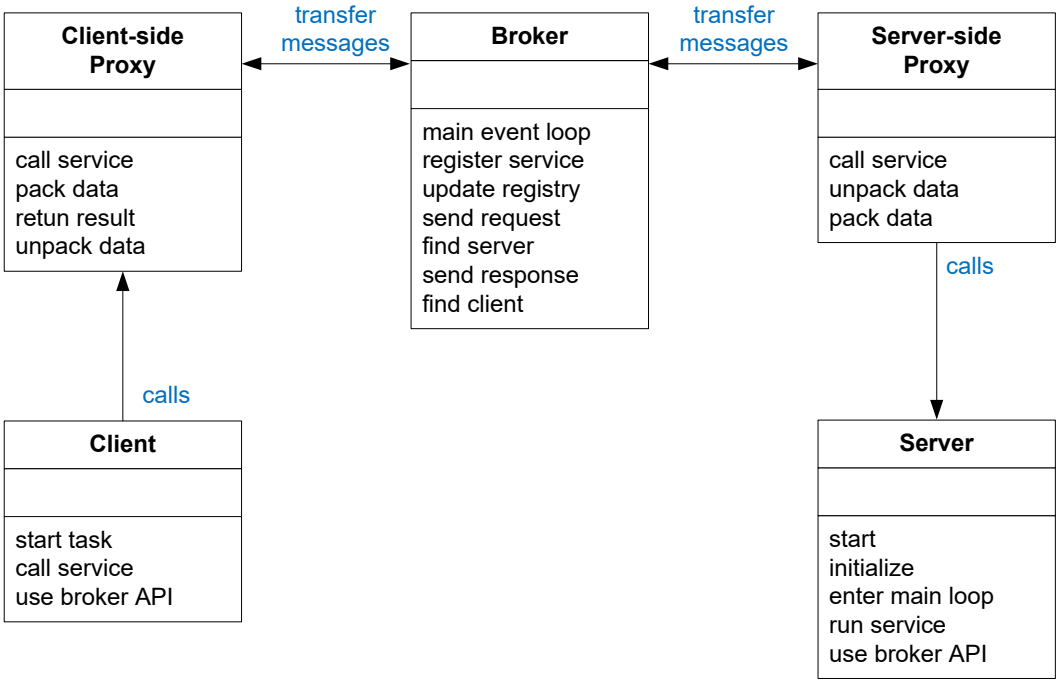
Broker

- Soluzione (scenari)
  - un server, al suo avvio, registra i propri servizi presso il broker
  - un client accede ai servizi indirettamente, tramite il broker
    - il broker seleziona un server in grado di erogare il servizio
    - se un server diviene indisponibile, il broker può scegliere dinamicamente di sostituirlo con un altro server

Broker



Struttura



Partecipanti

- *Server*
  - un oggetto o componente che offre servizi
  - i servizi sono esposti tramite una interfaccia
  - ci sono molti server – che offrono servizi diversi o anche gli stessi servizi
  - ogni server registra i propri servizi presso il Broker
- *Client*
  - un oggetto o componente che vuole fruire di servizi
  - ci sono molti client concorrenti
  - un client inoltra le proprie richieste al Broker
  - “client” e “server” vanno intesi in modo flessibile

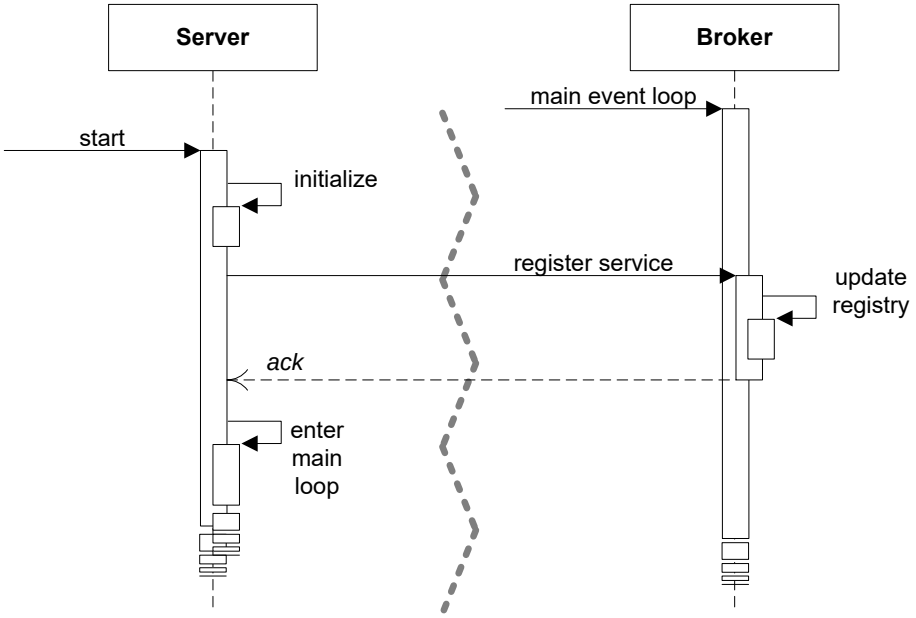
Partecipanti

- *Broker*
  - l'intermediario tra i client e i server nel sistema distribuito
  - è responsabile di trasmettere richieste e risposte tra client e server
  - è responsabile di gestire un registry dei servizi e dei server del sistema distribuito
  - offre ai server (mediante delle API) la funzionalità per registrare i loro servizi
  - offre ai client (mediante delle API) la funzionalità per richiedere l'esecuzione di servizi
  - può offrire altri servizi aggiuntivi

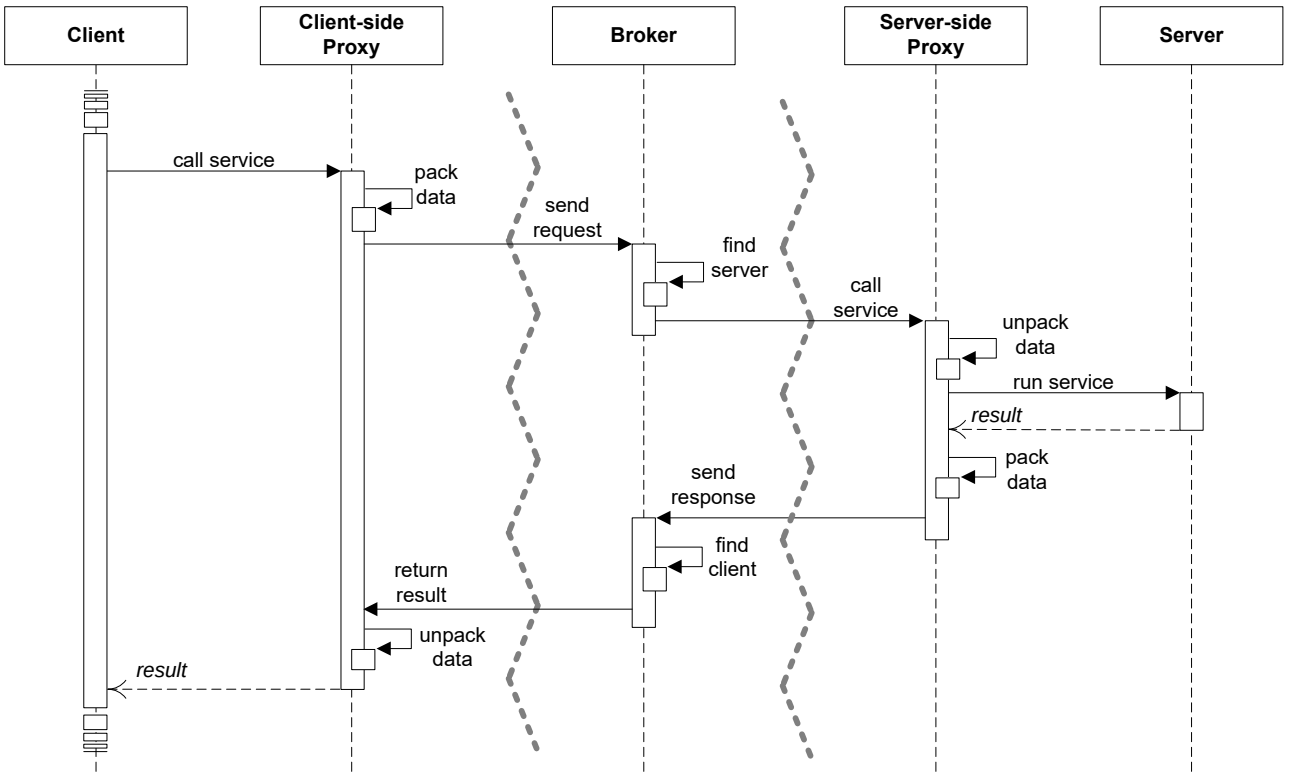
Partecipanti

- *Proxy lato client*
  - intermediario tra client e broker (e server)
  - vive localmente al processo del client
  - un remote proxy – fornisce trasparenza rispetto alla distribuzione
  - responsabile di inviare richieste al server (tramite il broker) e di ricevere risposte dal server (tramite il broker)
- *Proxy lato server*
  - intermediario tra (client e) broker e server
  - vive localmente al processo del server
  - responsabile di ricevere richieste dal client (tramite il broker), di invocare il servizio effettivo e di trasmettere le risposte al client (tramite il broker)

Scenario 1 – registrazione server



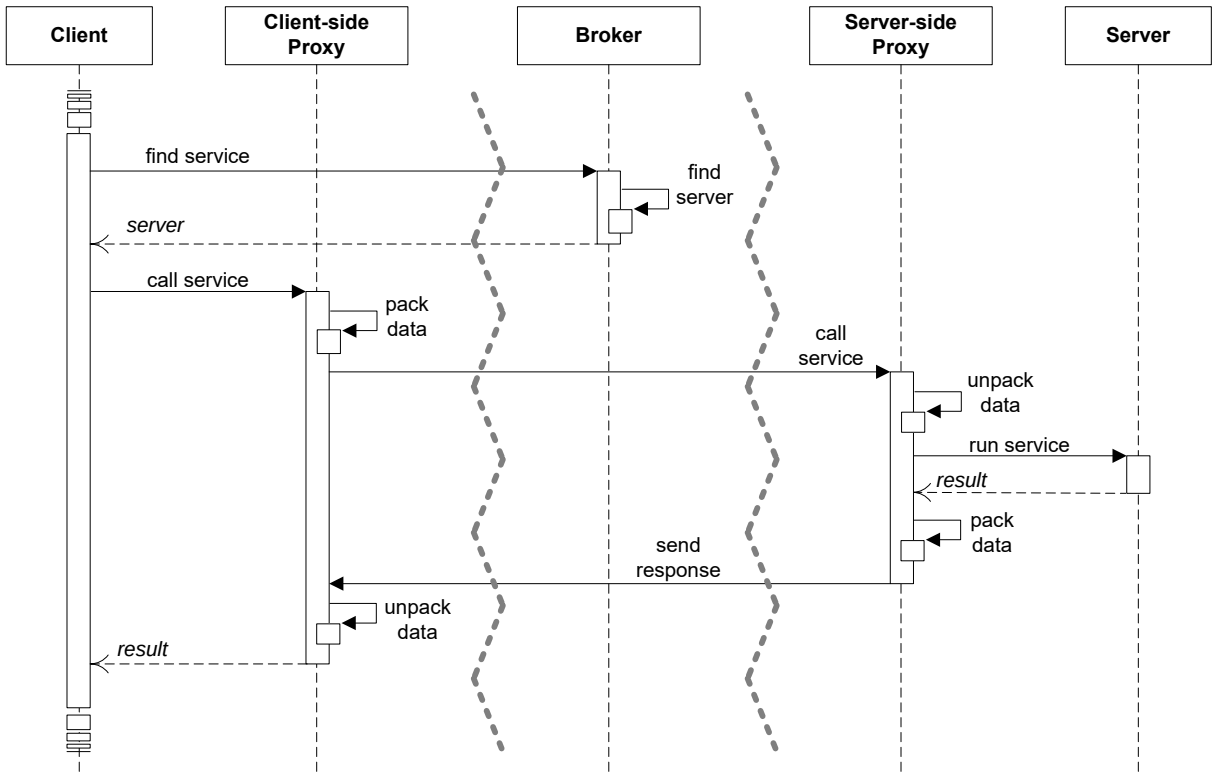
Scenario 2 – gestione richiesta client



Scenario 2 – varianti

- Due varianti principali per lo scenario per la gestione di una richiesta di un client
  - *comunicazione indiretta*
    - tutte le richieste e le risposte transitano attraverso il broker
  - *comunicazione diretta*
    - il broker è responsabile solo di mettere in comunicazione client e server
    - dopo di che, client e server comunicano in modo diretto

Scenario 2 – comunicazione diretta



Scenario 2 – varianti

- Confronto tra le varianti
  - nella comunicazione diretta, l'overhead di comunicazione è minore
  - nella comunicazione indiretta, il client è protetto in modo continuo da eventuali indisponibilità e da variazioni di locazione dei servizi
  - la comunicazione indiretta abilita un ulteriore scenario di interoperabilità, in cui il client e il server potrebbero essere eterogenei e potrebbero essere basati su protocolli o su formati diversi
    - una federazione di broker e degli ulteriori componenti "bridge"

Conseguenze

- Benefici
  - ☺ trasparenza dalla posizione
  - ☺ modificabilità ed estendibilità dei componenti
  - ☺ riusabilità di servizi esistenti
  - ☺ possibile l'interoperabilità tra tipi di client, server e broker diversi
- Inconvenienti
  - ☹ riduzione delle prestazioni, a causa dell'indirezione del broker
  - ☹ minor tolleranza ai guasti rispetto a una soluzione non distribuita
  - ☹ maggiore complessità
  - ☹ difficile da verificare

- Usi conosciuti

- Il pattern architetturale Broker è molto diffuso
  - la prima implementazione di un broker che è stata ampiamente usata è in CORBA (Common Object Request Broker Architecture, 1991)
  - il pattern broker è usato, in forme variate ed evolute, anche nelle tecnologie a componenti (come Java EE e .NET), nella comunicazione asincrona (message broker) e nell'architettura a servizi

* Discussione

- Il pattern architetturale Broker suggerisce di organizzare un sistema distribuito come un insieme di componenti che interagiscono sulla base di invocazioni remote
  - descrive l'infrastruttura di comunicazione per supportare le invocazioni remote, e per rendere trasparenti agli sviluppatori alcune complessità della distribuzione

Discussione

- Il pattern Broker (con le sue varianti ed evoluzioni) ha un ruolo fondamentale in molti sistemi distribuiti – soprattutto in quelli che (per sostenere prestazioni, scalabilità e disponibilità) hanno queste caratteristiche
  - sono composti da più componenti (o servizi)
  - questi componenti devono essere rilasciati in un ambiente distribuito in modo flessibile
  - questi componenti possono essere presenti in più repliche
  - il numero delle repliche dei componenti può variare dinamicamente
  - anche la locazione dei componenti può variare dinamicamente
  - i client di questi componenti devono essere comunque in grado di accedere alle funzionalità offerte dai componenti