Programmazione distribuita

P1 - Architetture

G.Campo

Architetture locali e distribuite

ARCHITETTURE **LOCALI** : tutte le entità logiche e componenti sono sullo stesso host

ARCHITETTURE **DISTRIBUITE** : applicazioni e componenti possono risiedere su nodi diversi, messi in comunicazione

- accesso a dati non disponibili localmente
- esecuzione di programmi concorrenti

- maggior complessità
- gestione della comunicazione

Tipologie

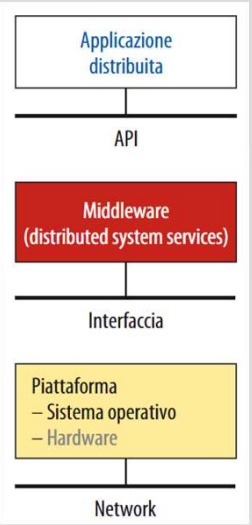
- **Client-Server**: un processo su un host richiede un servizio; un processo su altro host rimane in attesa di richieste
- **Multi-livello** (layers/tiers): ogni nodo è client del livello inferiore e server del livello superiore (es. arch. 3 tiers)
- **Completamente distribuite**: ogni nodo/oggetto si comporta da client e/o da server nei confronti di tutti gli altri (es. peer-to-peer)

Middleware

Definizione/i

Il MIDDLEWARE è una classe di tecnologie software che serve per la **gestione della complessità ed eterogeneità** presenti nei sistemi distribuiti.

Si interpone tra **Applicazioni** e **Sistema Operativo locale** creando un'architettura a livelli.



Funzionalità

- Servizi di astrazione e cooperazione
- Servizi di amministrazione del sistema
- Servizi di comunicazione
- Servizi per lo sviluppo delle applicazioni

Obiettivi

- Permette e garantisce l'interoperabilità delle applicazioni su diversi sistemi operativi
- Permette la connettività tra servizi che devono interagire e collaborare su piattaforme distribuite (es. espone Application Program Interfaces - API)

Meccanismi

- chiamata a procedure remote (Remote Procedure Call - RPC)
- scambio di messaggi (protocolli)
- **esposizione di Application Program Interfaces – API**
- (altri)

Pattern (I)

Definizione

«Un PATTERN SOFTWARE è
- una soluzione ben provata
- e ampiamente applicabile a un particolare problema di progettazione ricorrente,
- che è descritta in una forma standard, in modo che possa essere facilmente condivisa e riusata.»

«La SOLUZIONE è
- specificata descrivendo relativamente agli oggetti o componenti che la costituiscono,
- i ruoli insieme alle loro responsabilità e relazioni, nonché i modi in cui essi collaborano.»

Descrizione della soluzione
- Nome
- Contesto
- Problema
- Soluzione
- Conseguenze

Pattern (II)

Tipologie

- Architetturali
- Design
- Idiomi
- Linguaggi
- etc.…

Es. 1 - Design Patterns

GOF (1995)

- Adapter
- Proxy
- Factory
- Façade
- Observer
- Repository

….

Es. 2 - Architectural Patterns

POSA (Patterns-Oriented Software Architecture)

- Vol.1 (1996)
- Vol. 2 (2000)
- Vol. 3 (2004)
- Vol. 4 (2007)
- Vol. 5 (2007)

Pattern-Oriented Software Architecture - Wikipedia

Pattern Architetturali

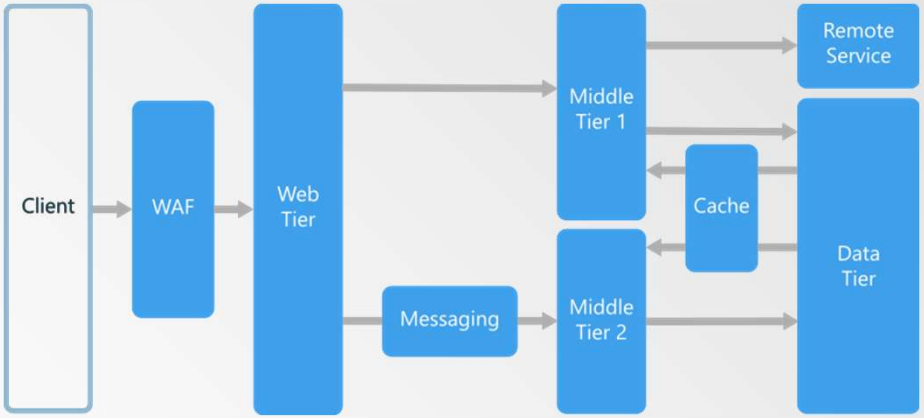
Architetture Distribuite

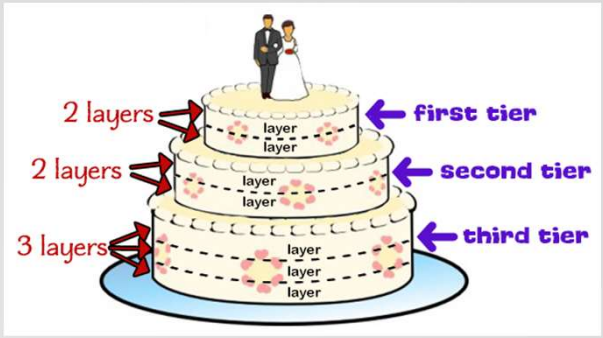
Architecture style	Dependency management	Domain type
N-tier	Horizontal tiers divided by subnet	Traditional business domain. Frequency of updates is low.
Web-Queue-Worker	Front and backend jobs, decoupled by async messaging.	Relatively simple domain with some resource intensive tasks.
Microservices	Vertically (functionally) decomposed services that call each other through APIs.	Complicated domain. Frequent updates.
Event-driven architecture.	Producer/consumer. Independent view per sub-system.	IoT and real-time systems
Big data	Divide a huge dataset into small chunks. Parallel processing on local datasets.	Batch and real-time data analysis. Predictive analysis using ML.
Big compute	Data allocation to thousands of cores.	Compute intensive domains such as simulation.

Architecture styles - Azure Application Architecture Guide | Microsoft Docs
Event-Driven Architecture (amazon.com)

Architettura Multi livello

A. Multi livello (N-tier)





Layer (strato)

[N-tier architecture style - Azure Architecture Center | Microsoft Docs]

"Logical layers are merely a way of **organizing** your code. Typical layers include Presentation, Business and Data – the same as the traditional 3-tier model. But when we're talking about layers, we're only talking about logical organization of code. In no way is it implied that these layers might run on different computers or in different processes on a single computer or even in a single process on a single computer. All we are doing is discussing a way of organizing a code into a set of layers defined by specific function."

↳ distribuisco le **responsabilità**

Tier (livello)

"Physical tiers however, are only about where the code runs. Specifically, tiers are places where layers are deployed and where layers run. In other words, tiers are the physical deployment of layers."

↳ distribuisco il **carico applicativo**

- Rockford Lhotka -

Architettura “Hexagonal”

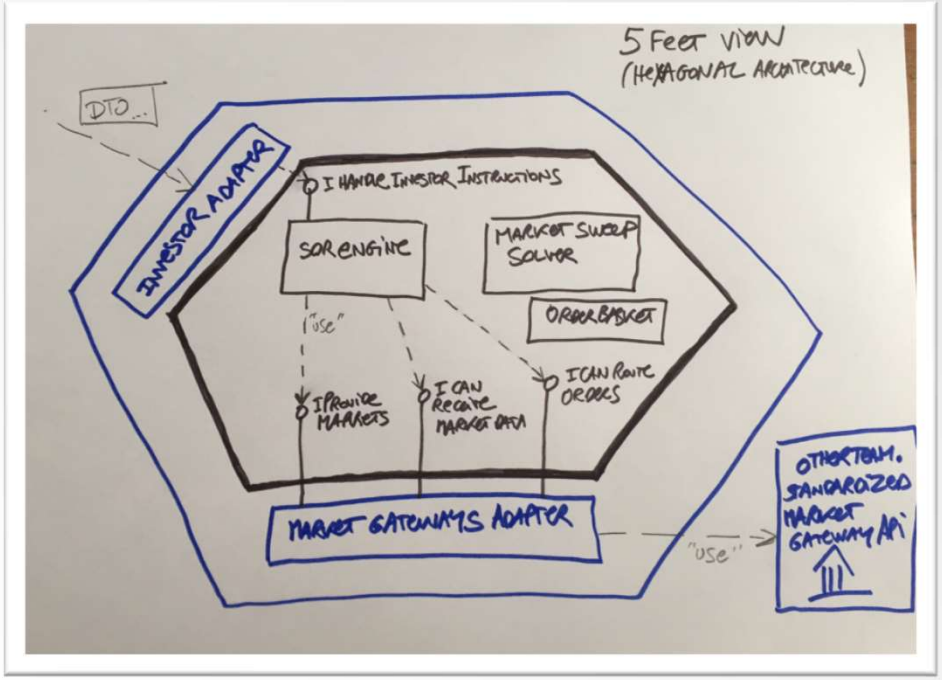
Alistair COCKBURN

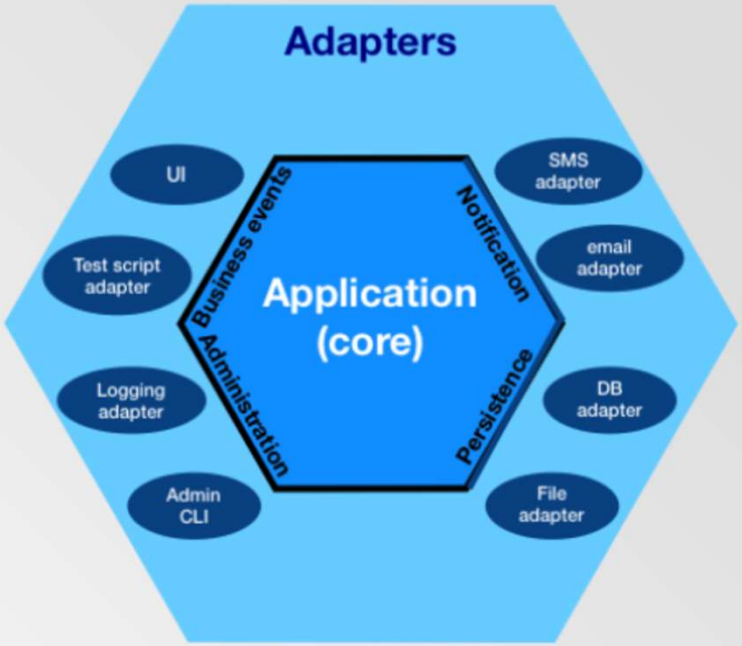
Agile (2001)

Manifesto for Agile Software Development (agilemanifesto.org)

The Heart of Agile | More powerful, More human

Hexagonal (2005)





Architettura SOA (I)

Caratteristiche

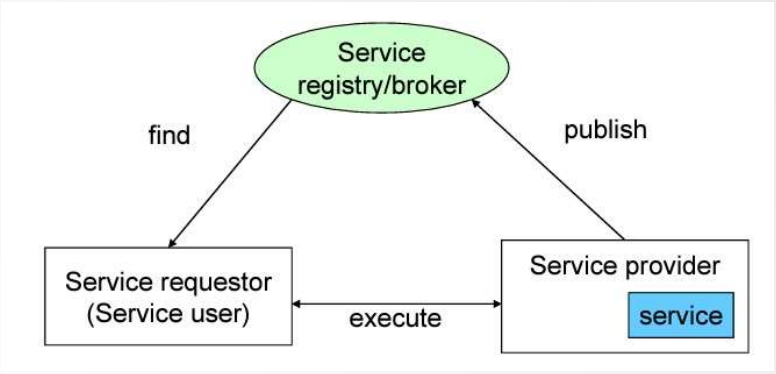
- Contratto ed astrazione (incapsulamento)
- Accoppiamento debole
- Componibilità e riusabilità
- Stateless
- Accessibile in rete

Architettura SOA (II)

Elementi

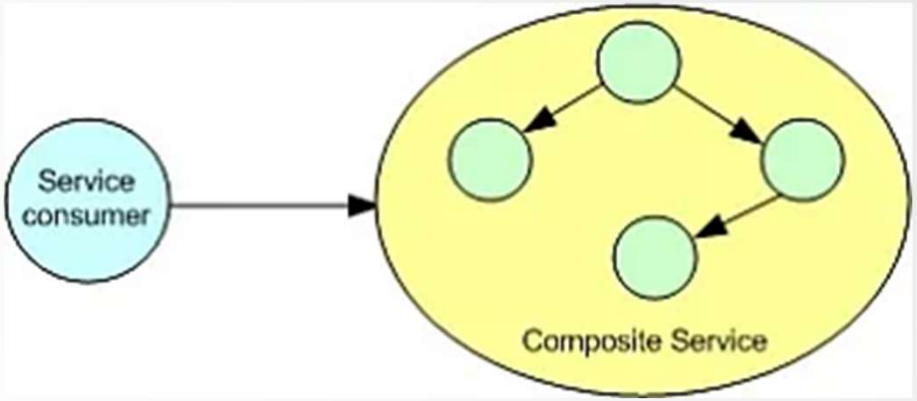
- Servizio — componente in grado di erogare un insieme di funzionalità
- Interfaccia di servizio — descrizione delle funzionalità del servizio basata su uno standard aperto del web
- Fornitore [s. provider] — sistema o organizzazione che espone il servizio in modo che sia accessibile mediante la sua interfaccia tramite tecnologie web standard
- Client (consumatore) [s. requestor, s. user] — sistema che fruisce del servizio mediante la sua interfaccia tramite tecnologie web standard
- (service) Registry [s. broker , s. bus] — servizio infrastrutturale per supportare la scoperta di servizi oppure per mediare l'invocazione di servizi

- Luca Cabibbo -

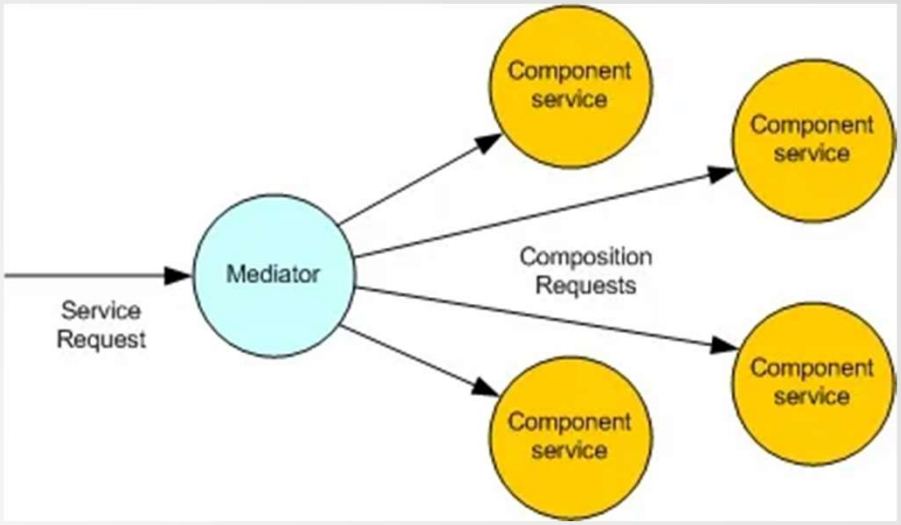


Architettura SOA (III)

Composizione di servizi



Orchestrazione di servizi



Copyright

Tutto il materiale raccolto ed utilizzato per realizzare le slides a fini didattici è di dominio pubblico ed, in buona fede, ritenuto in licenza CC. Qualora i rispettivi autori e proprietari lo abbiano rilasciato con diversa licenza, possono contattarmi e segnalarmelo; in tal caso al materiale verranno prontamente apportate le opportune e debite variazioni.

Il presente materiale è quindi di proprietà dei relativi autori e del sottoscritto, e può essere utilizzato solo per motivi di ricerca ed insegnamento pubblico senza fini di lucro.

Gabriele Campo
(gabriele.campo@itismattei.net)