NOME	TIPO	DESCRIZIONE	PRO	CONTRO	ESEMPI
Stratificato(Layered)	Strutturale	Organizza il sistema in un insieme di livelli, ognuno dei quali fornisce un insieme di servizi e si appoggia al livello subito sottostante	/	/	Android
Repository Model	Strutturale	I dati condivisi sono mantenuti in un database centrale a cui hanno accesso tutti i sotto-sistemi, che quindi non possono comunicare direttamente	1) Modo efficiente di condividere grandi quantità di dati, 2) Gestione centralizzata di backup, security, ecc	1) I sottosistemi devono trovare un compromesso sul modello dei dati per il repository, 2) evoluzione dello schema difficile e costosa, 3) difficile da rendere distribuito	Applicazioni in cui i dati sono prodotti da un sottosistema e utilizzati da un altro. Ad esempio Case Tool (VP)
Client-Server Model	Strutturale	Dati e computazione distribuiti su server che forniscono servizi e client che li utilizzano. La rete permette ai client di accedere ai Server. Dati i 3 stati: (Presentation, Business, Data) Logic, essi possono essere distribuiti tra client e server (Thin Client se ha solo il Presentation Logic, Thick/Fat se ha anche l'Application Logic)	/	/	Web Apps
Client-Server 3 Livelli	Strutturale	La logica del sistema è uno strato separato che gestisce multi-utenti. Gli strati di logica e dati sono distribuiti su più server	/	/	Web Apps
P2P Model	Strutturale	Ogni componente esegue il suo processo giocando il ruolo sia di client sia di server (la sua interfaccia specifica i servizi che offre e quelli che richiede, e i dati sono ciò che distingue i peer)	Scalano bene (new peer = new data) e sono fault-tolerant	/	eMule, Skype
Pipe & Filter	Strutturale	I filtri effettuano trasformazioni che elaborano i loro input per produrre output. Le pipe sono connettori che trasmettono i dati tra i filtri	1) Supporta il riuso delle trasformazioni, 2) Intuitivo: permette di capire il funzionamento del sistema come composizione dei filtri, 3) Aggiunta di nuove trasformazioni facile, 4) Ez da implementare (anche OO), 5) Supporta concorrenza/parallelismo	1) Non adatto a sistemi interattivi, 2) Porta allo sviluppo di sistemi batch dove ogni filtro è una trasformazione completa dell'input, 3) overhead di (un) parsing dei dati nei filtri	Shell Unix, Invoice Processing System
Call-Return Model	Di controllo (centralizzato)	Il Main/Driver gestisce l'esecuzione degli altri sottosistemi (Routine) dando anche un ordine (modello top-down dove il controllo si sposta dalla radice verso il basso). Solo chi ha il controllo è in esecuzione	/	/	Sistemi sequenziali
Manager Model	Di controllo (centralizzato)	Se i sottosistemi controllati possono funzionare in parallelo. Il Manager controlla avvio, terminazione e coordinamento degli altri processi	/	/	Sistemi concorrenti, real time (Allarme banca, con sensori interrogati iterativamente dal manager)
Broadcast Model	Di controllo (su eventi)	Un componente annuncia (broadcast) >=1 eventi (invece che invocare direttamente una procedura). Tutti sono in ascolto e chi è interessato esegue la procedura, analogamente ai BUS	1) Semplice aggiungere, togliere o sostituire componenti, 2) semplice il riuso dei componenti	1) I sottosistemi che inviano un evento non sanno quando verrà gestito, 2) Lo scambio dei dati può complicare (o passo i dati con l'evento o uso un repository che però può complicare le performance, 3) difficili da implementare e testare	Sistemi militari e di sorveglianza
/	Eterogenee gerarchiche	X in stile Y, con dentro A,B,C in stili diversi da Y	/	/	Sistema con pipe&filter con componente interna in layered
/	Eterogenee mix	X in stile A,B,C misto	/	/	Compilatore (pipe&filter + repository)
Microservizi	Strutturale	In contrapposizione ai monoliti: suite di piccoli servizi, ognuno autonomo sul suo processo realizzando una funzionalità, comunicante con meccanismi lightweight (RPC, REST, ...) e con il suo DB. Si può avere un API Gateway che espone un interfaccia verso i client (poi chiamando i vari servizi che compongono il servizio chiamato esposto dall'interfaccia)	Innoavaitivo e contrapposto ai monoliti, mitiga dunque i problemi nella crescita delle app (complessità, risoluzione bug, modifiche, tempi per deployment, lavoro in team)	1) Stabilire la dimensione dei micro servizi, 2) Sviluppo del meccanismo di comunicazione, 3) Esposizione ai disservizi di rete, 4) Gestione dello schema partizionato dei DB, 5) Difficoltà di Testing, 6) Maggior consumo di risorse e memoria	Amazon, Ebay, Netflix, Uber, ...