



Luca Cabibbo
Architettura
dei Sistemi
Software

Introduzione ai sistemi distribuiti

dispensa asw410
ottobre 2023

*The most important things in life
are the connections you make with others.*

Tom Ford



- Riferimenti

- ❑ Luca Cabibbo. **Architettura del Software: Strutture e Qualità**. Edizioni Efestò, 2021.
 - Capitolo 21, **Introduzione ai sistemi distribuiti**
- ❑ [POSA4] Buschmann, F., Henney, K., and Schmidt, D.C. **Pattern-Oriented Software Architecture (Volume 4): A Pattern Language for Distributed Computing**. Wiley, 2007.
- ❑ Coulouris, G, Dollimore, J., Kindberg, T., and Blair, G. **Distributed Systems: Concepts and Design**, fifth edition. Pearson, 2012.
- ❑ Shaw, M. **Procedure Calls are the Assembly Language of Software Interconnections: Connectors Deserve First-Class Status**. Technical report CMU/SEI-94-TR-002, 1994.
- ❑ Taylor, R.N., Medvidovic, N., and Dashofy, E.M. **Software Architecture: Foundations, Theory, and Practice**. John Wiley & Sons, 2010.
- ❑ Bernstein, P. **Middleware**. Communications of the ACM, 1996.



- Obiettivi e argomenti

□ Obiettivi

- introdurre i sistemi distribuiti
- introdurre i connettori e il middleware
- introdurre brevemente due stili fondamentali di comunicazione distribuita e i pattern architetturali POSA che li rappresentano

□ Argomenti

- introduzione ai sistemi distribuiti
- connettori
- middleware
- stili di comunicazione e pattern architetturali POSA
- discussione



* Introduzione ai sistemi distribuiti

□ Alcune possibili definizioni – un **sistema distribuito** è

- un sistema software in cui l'elaborazione è distribuita su più computer (nodi)
- un sistema di elaborazione in cui un numero di componenti coopera comunicando in rete [POSA4]
- un sistema in cui i componenti hardware o software posizionati in computer collegati in rete comunicano e coordinano le proprie azioni solo tramite lo scambio di messaggi [Coulouris]
- un sistema in cui il fallimento di un computer di cui nemmeno conosci l'esistenza può rendere inutilizzabile il tuo computer [Lamport]



Architettura dei sistemi software distribuiti

- Architettura dei sistemi software distribuiti (intuizioni)
 - un sistema distribuito è costituito da un insieme di elementi software, in esecuzione su più nodi, che comunicano tra di loro mediante una rete
 - sono possibili diverse modalità di comunicazione tra gli elementi software distribuiti
 - sono possibili diverse modalità di organizzazione per gli elementi software distribuiti
 - i sistemi distribuiti possono offrire numerosi benefici – ma possono anche sollevare diversi inconvenienti
 - la sfida è cercare di ottenere i possibili benefici, minimizzando gli inconvenienti



Benefici della distribuzione

- 😊 Connettività e collaborazione
 - possibilità di condividere risorse hardware e software
- 😊 Tolleranza ai guasti
- 😊 Prestazioni e scalabilità
- 😊 Sistemi inerentemente distribuiti
- 😊 Apertura
- 😊 Economicità
 - i sistemi distribuiti offrono, in genere, un rapporto qualità/prezzo migliore rispetto ai sistemi centralizzati



Svantaggi legati alla distribuzione

- ☹️ Complessità
- ☹️ Sicurezza
- ☹️ Non prevedibilità
- ☹️ Gestibilità
- ☹️ Complessità accidentale
 - introdotta dall'uso di strumenti di sviluppo non opportuni
- ☹️ Metodi e tecniche non adeguati
- ☹️ Continua re-invenzione e riscoperta di concetti e soluzioni



Ipotesi errate comuni sui sistemi distribuiti

- ☐ Un esempio di “complessità accidentale” – 8 **ipotesi errate** comuni sui sistemi distribuiti identificate da Peter Deutsch
 1. la rete è affidabile
 2. la latenza è zero
 3. la banda a disposizione è infinita
 4. la rete è sicura
 5. la topologia della rete non cambia
 6. c'è un amministratore (che risolve tutti i problemi della rete)
 7. il costo di trasporto è zero
 8. la rete è omogenea



- Sistemi centrali e virtualizzazione

- La tecnologia dei *sistemi centrali* (o *mainframe*) è ancora attuale e in continua evoluzione
 - è una tecnologia “ricca di funzionalità sempre più avanzate e di innovazioni tecniche via via più sofisticate”
 - “oggi i mainframe sono presenti e hanno un ruolo insostituibile in tutto il mondo nelle infrastrutture informatiche di importanti aziende, società di servizi pubbliche e private e grandi istituzioni nazionali ed internazionali”
- La *virtualizzazione* è una tecnologia importante oggi nel contesto dei sistemi software, soprattutto distribuiti
 - la virtualizzazione di sistema consente a uno o più computer fisici di ospitare più macchine (computer) virtuali
 - nei sistemi software distribuiti, i nodi possono essere costituiti da computer fisici o anche virtuali



* Connettori

- Nell'architettura di un sistema software è possibile distinguere due tipi principali di elementi software
 - *componenti* – responsabili di *funzionalità* e di *dati/informazioni*
 - *connettori* – responsabili delle *interazioni* tra componenti
- Questa distinzione ha un ruolo fondamentale, soprattutto nei sistemi distribuiti
 - riflette l'indipendenza tra gli aspetti funzionali e quelli relativi alle interazioni



Componenti e connettori – esempi

- Alcune possibili tipologie di componenti e connettori
 - componenti: moduli
 - connettore: una chiamata di procedura tra moduli

 - componenti: processi
 - connettori: una chiamata di procedura remota, una pipe, o un protocollo che regola lo scambio di messaggi tra processi

 - componenti: un processo e una base di dati
 - connettore: l'accesso alla base di dati da parte del processo



Componenti e connettori

- I *componenti* [Shaw] sono il luogo della computazione e dello stato
 - ogni componente ha una *specificata di interfaccia* che definisce le sue proprietà (funzionali e di qualità)
 - ogni componente è di un qualche tipo – ad es., server, client, peer, filtro, memoria, ...
 - l'interfaccia di un componente comprende la specifica dei “ruoli” che il componente può rivestire nell'interazione con altri componenti – ad es., l'essere il client o il server di un certo servizio
 - questi “ruoli” sono chiamati *porte*



Componenti e connettori

- I **connettori** [Shaw] sono il luogo delle relazioni tra componenti
 - i connettori sono mediatori di interazioni
 - ogni connettore ha una **specifica di protocollo** che definisce le sue proprietà
 - ad es., regole sul tipo di interfacce che il connettore è in grado di mediare e impegni sulle proprietà dell'interazione (come sicurezza, affidabilità e prestazioni)
 - ogni connettore è di un qualche tipo – ad es., chiamata di procedura remota, pipe, evento, broadcast, ...
 - il protocollo di un connettore comprende la specifica dei **ruoli** (**role**) che devono essere soddisfatti – ad es., client e server
 - la composizione dei componenti avviene mettendo in relazione porte di componenti con ruoli di connettori

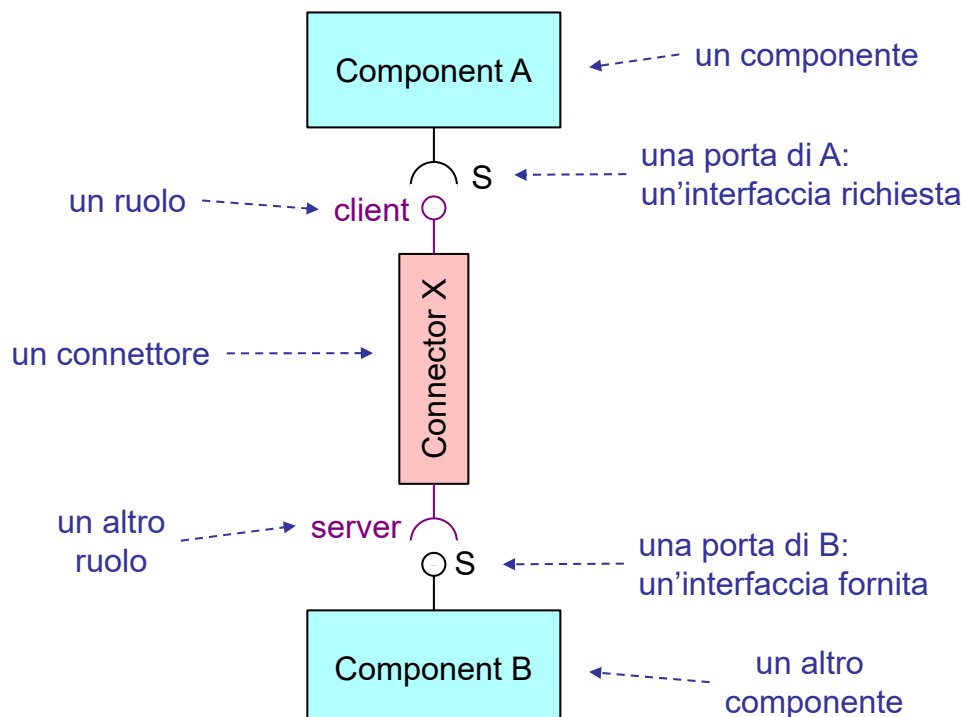
13

Introduzione ai sistemi distribuiti

Luca Cabibbo ASW



Componenti e connettori



14

Introduzione ai sistemi distribuiti

Luca Cabibbo ASW



Connettori

- Ci sono diversi motivi per trattare i connettori separatamente dai componenti [Shaw]
 - la distinzione tra componenti e connettori è un'applicazione del principio di separazione degli interessi
 - la scelta e la progettazione dei connettori (delle interazioni) è importante tanto quanto quella dei componenti
 - alcune scelte architettoniche non hanno una collocazione naturale in nessuno dei componenti di un sistema
 - la progettazione dei connettori può essere fatta separatamente da quella dei componenti
 - i connettori sono spesso indipendenti dalle applicazioni – sono potenzialmente astratti e riutilizzabili in più contesti
 - questo ha portato allo sviluppo di numerosi servizi di middleware



- Ulteriori caratteristiche dei connettori



- [Taylor] descrive delle ulteriori caratteristiche dei connettori e ne fornisce una classificazione
 - le caratteristiche fondamentali di un connettore sono le modalità di gestione del *flusso di controllo* e di gestione del *flusso di dati* tra due o più componenti



Ruoli dei connettori



- Ciascun connettore può occuparsi di una o più delle seguenti responsabilità
 - *comunicazione* – trasferimento di dati tra componenti
 - *coordinamento* – relativo al trasferimento del controllo tra componenti
 - *facilitazione* – per mediare l'interazione tra componenti
 - *conversione* – di formati e di interazioni



Tipologie di connettori



- Due tipologie principali di connettori
 - *chiamata di procedura*
 - può essere locale o remota, sincrona o asincrona
 - *eventi o messaggi*
 - per consentire la notifica di eventi o lo scambio di messaggi tra componenti, e abilitarne la successiva elaborazione
 - a-uno oppure a-molti, basata su polling (sincrono) o sottoscrizione (asincrona), con diversi livelli di affidabilità



- Ulteriori tipologie di connettori
 - connettori per l'*accesso ai dati*
 - connettori per *stream* – per trasmettere grandi quantità di dati tra componenti
 - connettori di *collegamento* (binding)
 - *arbitri* – facilitatori per risolvere possibili conflitti nell'interazione tra componenti
 - *adattatori* – per consentire la comunicazione tra componenti che non sono stati progettati per interoperare direttamente
 - *distributori* – per effettuare il routing dei dati e del controllo



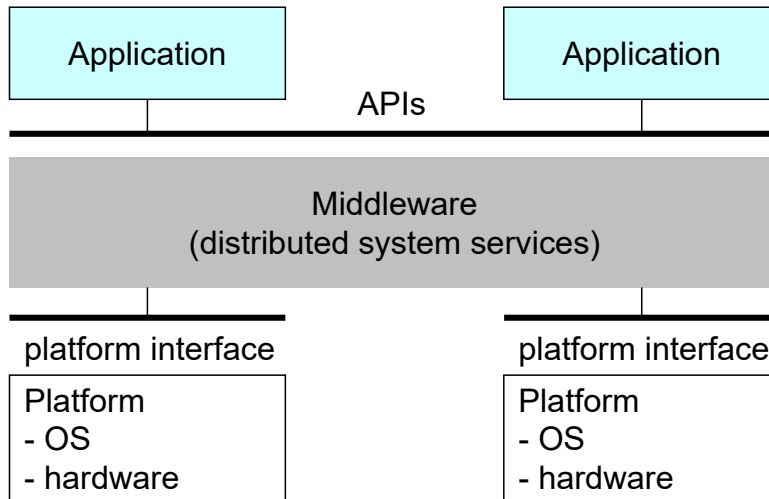
* Middleware

- In pratica, lo sviluppo dei connettori per i sistemi software distribuiti è sostenuto dal *middleware*
 - un insieme di tecnologie e soluzioni software sviluppate per aiutare gli sviluppatori nella gestione della complessità e della eterogeneità presenti nei sistemi distribuiti
 - uno strato software “in mezzo”, che supporta lo sviluppo dei connettori



Middleware

- Un **servizio di middleware** è [Bernstein]
 - un servizio distribuito, general-purpose e multi-piattaforma
 - che si colloca tra piattaforme e applicazioni
 - fornisce un'astrazione di programmazione distribuita
 - per aiutare a risolvere problemi di eterogeneità e distribuzione



Middleware e connettori

- Il middleware sostiene lo sviluppo dei sistemi distribuiti
 - i servizi di middleware semplificano lo sviluppo dei connettori richiesti dalle applicazioni distribuite
 - ci sono diverse famiglie o tecnologie di middleware – ciascuna tecnologia implementa uno o più paradigmi di interazione e comunicazione distribuita
 - ogni servizio di middleware affronta e risolve alcuni problemi comuni nei sistemi distribuiti



Tecnologie di middleware

- Ci sono diverse famiglie di tecnologie di middleware – ad esempio
 - middleware per chiamate di procedure remote (RPC, remote procedure call)
 - middleware per oggetti distribuiti (RMI, remote method invocation)
 - middleware per la comunicazione asincrona (message broker)
 - middleware per componenti distribuiti
 - middleware orientato ai servizi
 - ... *in continua evoluzione* ...



Middleware e trasparenza

- Ogni servizio di middleware affronta e risolve alcuni problemi comuni nei sistemi distribuiti
 - spesso consentono di mascherare qualche tipo di eterogeneità che può essere presente nei sistemi distribuiti
 - ad es., nelle reti e nell'hardware, nei sistemi operativi e/o nei linguaggi di programmazione
 - le astrazioni di programmazione offerte dal middleware possono fornire trasparenza rispetto ad aspetti come posizione, concorrenza, replicazione, fallimenti e mobilità
 - in alternativa, il programmatore dovrebbe farsi esplicitamente carico di questi aspetti



□ Uso efficace del middleware

- se utilizzato in modo opportuno, il middleware consente di affrontare e risolvere diverse problematiche significative nello sviluppo dei sistemi distribuiti
- consente di concentrarsi sullo sviluppo dei componenti e della logica di business
- per aumentare effettivamente la produttività, il middleware deve essere selezionato e utilizzato in modo corretto
- malgrado i molti benefici offerti dal middleware, il middleware non è una panacea per i sistemi distribuiti – non risolve “magicamente” i problemi derivanti da decisioni di progetto povere



- ## □ Sviluppare middleware per la comunicazione nei sistemi distribuiti è un'attività complessa
- fortunatamente, sono disponibili un'ampia varietà di servizi di middleware, standard e commerciali, già usati con successo in moltissimi sistemi distribuiti
 - solo raramente c'è la necessità di progettare e implementare nuovi stili o paradigmi di interazione distribuita



- Relazioni tra servizi di middleware e pattern architetturali
 - l'applicazione di alcuni pattern architetturali è sostenuta, dal punto di vista tecnologico, da opportuni servizi di middleware
 - altri pattern architetturali, invece, descrivono l'architettura (dell'infrastruttura) di alcuni servizi di middleware
 - è utile conoscere e comprendere queste relazioni



* Stili di comunicazione e pattern architetturali POSA

- Le tecnologie di middleware sostengono pochi stili di comunicazione distribuita
 - due stili fondamentali di comunicazione sono
 - *invocazione remota*
 - *comunicazione asincrona*
 - esistono anche altre tipologie di connettori e stili di comunicazione – ad es., le basi di dati condivise e lo streaming di dati
 - in questo corso ci concentriamo soprattutto su questi due stili di comunicazione fondamentali



Stili di comunicazione e pattern architetturali POSA

- I due stili di comunicazione principali sono rappresentati da due (anzi tre) pattern architetturali [POSA] fondamentali
 - invocazione remota – *Broker* [POSA]
 - comunicazione asincrona – *Messaging* [POSA4] e *Publisher-Subscriber* [POSA4]
- Una precisazione
 - per semplicità, ignoreremo la distinzione tra *Messaging* e *Publisher-Subscriber*
 - anzi, li considereremo un singolo pattern architetturale (che li generalizza) che chiameremo (un po' impropriamente) *Messaging*

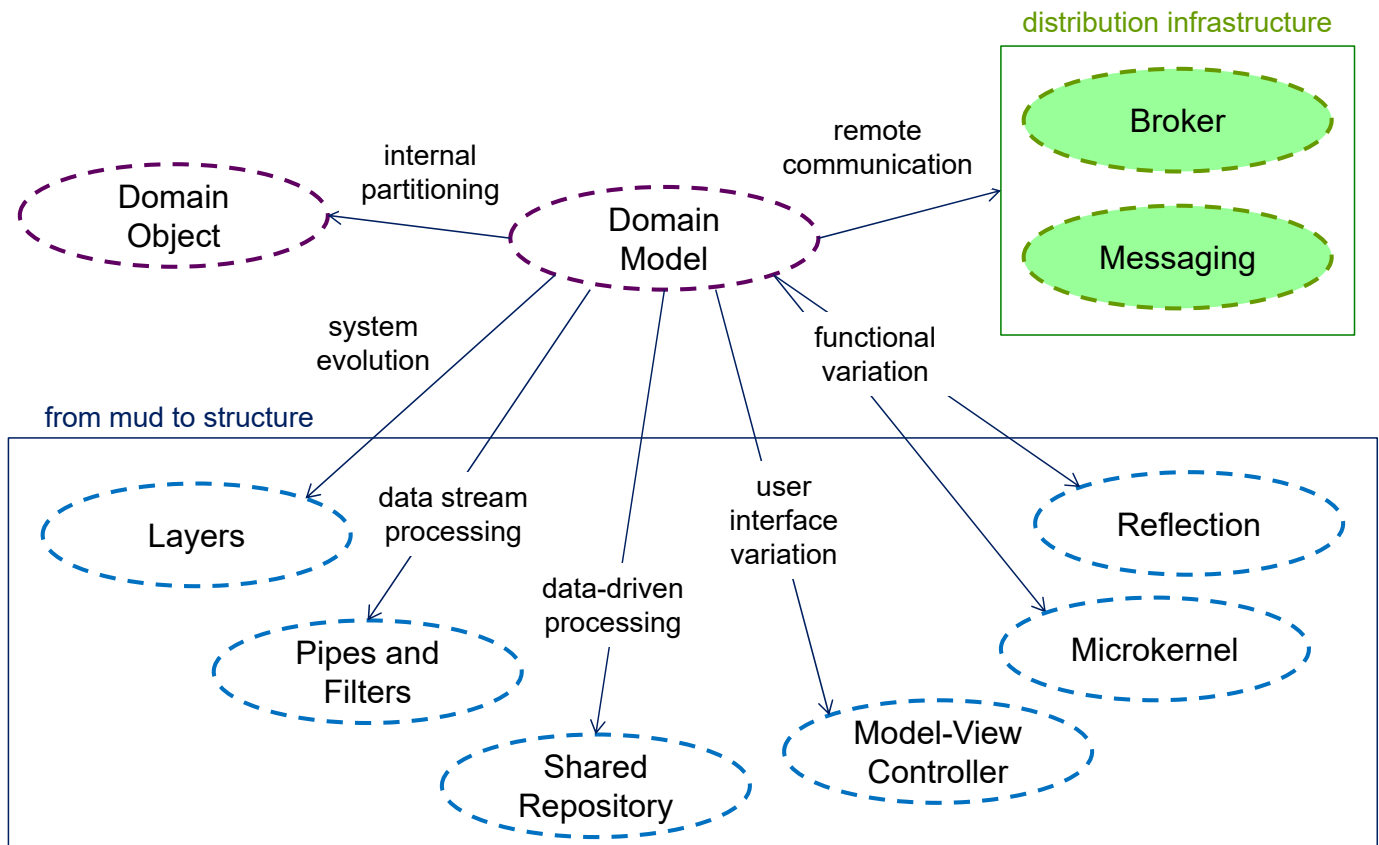


Stili di comunicazione e pattern architetturali POSA

- *Broker* [POSA]
 - organizza il sistema distribuito in un insieme di componenti che interagiscono sulla base di invocazioni remote
- *Messaging* [POSA4]
 - organizza il sistema in componenti che interagiscono sulla base dello scambio di messaggi, in modo asincrono



Stili di comunicazione e pattern architetturali POSA



31

Introduzione ai sistemi distribuiti

Luca Cabibbo ASW



* Discussione

- Abbiamo introdotto brevemente
 - i sistemi distribuiti – con i benefici e i rischi associati, e la sfida posta dalla distribuzione
 - i connettori – gli elementi architetturali che consentono la comunicazione tra i componenti software nei sistemi distribuiti
 - il middleware – che semplifica lo sviluppo dei connettori nei sistemi distribuiti, sulla base di opportune astrazioni di programmazione distribuite, che affronta e risolve alcuni problemi comuni nei sistemi distribuiti
 - due stili principali di comunicazione distribuita – nonché i pattern architetturali che li supportano
- Questi argomenti verranno ripresi e discussi nel seguito del corso
 - questa parte del corso presenta anche alcuni pattern architetturali fondamentali per sistemi distribuiti

32

Introduzione ai sistemi distribuiti

Luca Cabibbo ASW