Luca Cabibbo

Architettura dei Sistemi Software

Introduzione all'architettura del software

dispensa asw110

ottobre 2024

Every software system is constructed to satisfy an organization's business goals, and the architecture of a system is a bridge between those (often abstract) business goals and the final (concrete) resulting system.

Len Bass, Paul Clements, and Rick Kazman

- Riferimenti

- Luca Cabibbo. **Architettura del Software**: Strutture e Qualità. Edizioni Efesto, 2021.
  - Capitolo 1, Introduzione all'architettura del software
- [SAP] Len Bass, Paul Clements, Rick Kazman. **Software Architecture in Practice**. Addison Wesley, fourth edition, 2022
- [SSA] Nick Rozanski, Eoin Woods. **Software Systems Architecture**: Working with Stakeholders Using Viewpoints and Perspectives. Addison Wesley, second edition, 2012
- Maier, M.W. and Rechtin, E. **The Art of Systems Architecting**, third edition. CRC Press, 2009.

- Obiettivi e argomenti

- Obiettivi
  - introdurre l'architettura del software e motivare la sua importanza
  - introdurre l'approccio dell'architettura del software
- Argomenti
  - introduzione
  - sulla progettazione di sistemi complessi
  - introduzione all'architettura del software
  - l'architettura del software come disciplina
  - architettura monolitica (e perché evitarla)
  - esempio: cloud-native software
  - una disciplina in continua evoluzione
  - discussione

* Introduzione

- Il panorama dell'informatica e dell'IT (Information Technology) è cambiato in modo significativo nell'ultima decina di anni – e sta continuando a cambiare ancora ☺
  - i computer e Internet – e soprattutto il software che li fa funzionare – hanno reso il mondo oggi molto più connesso
  - l'IT sta avendo un ruolo sempre più importante nella vita delle persone e di molte organizzazioni
- I grandi sistemi software di oggi sono tra le strutture più complesse mai costruite dagli uomini
  - sono composti da milioni di linee di codice, centinaia di componenti software, centinaia di tabelle nelle basi di dati, e sono eseguiti su dozzine di computer (fisici o virtuali)
  - sono sviluppati, amministrati e fatti evolvere da team grandi, che spesso sono distribuiti e operano su periodi di tempo estesi

Esempio: Facebook

- Alcuni numeri su Facebook (dati del 2019)
  - in Italia ha circa 32 milioni di utenti attivi mensilmente (almeno un collegamento al mese) – 25 milioni lo utilizzano ogni giorno – di cui oltre 24 milioni da dispositivi mobili
  - nel mondo ha oltre 2.45 miliardi di utenti attivi mensilmente – 1.62 miliardi sono attivi giornalmente – di cui circa il 75% da dispositivi mobili – ogni minuto vengono inviati circa 290 mila messaggi, 510 mila commenti e vengono caricate circa 130 mila foto
  - Facebook si conferma ancora nel 2019 la piattaforma social più popolare
  - anche se le sue funzionalità sono piuttosto semplici, questi numeri (e l'obiettivo di farli crescere ancora) pongono una sfida molto difficile

Esempio: Facebook

- Alcuni numeri su Facebook (dati non ufficiali)
  - il sistema software è composto da oltre 60 milioni di linee di codice sorgente – scritto da oltre 1000 sviluppatori
  - lato server
    - il sistema è eseguito su circa 60000 nodi – di cui diverse migliaia per le basi di dati, usando una decina di DBMS (relazionali e non) differenti
    - il software viene aggiornato e rilasciato migliaia di volte al giorno
  - lato client
    - il sistema è acceduto dai dispositivi (fissi o mobili) degli utenti – tanti e di tanti tipi diversi: PC, tablet e smartphone – tramite uno dei tanti browser web o un client specifico

Esempio: Facebook

- Alcuni numeri su Facebook (dati non ufficiali)
  - il sistema software è composto da oltre 60 milioni di linee di codice sorgente – scritto da oltre 1000 sviluppatori

Si tratta di un sistema estremamente complesso, in cui il software svolge un ruolo dominante

utenti – tanti e di tanti tipi diversi: PC, tablet e smartphone – tramite uno dei tanti browser web o un client specifico

Il software è molto complesso

- I sistemi software di oggi sono molto complessi, perché
  - devono soddisfare gli interessi di numerose parti interessate
  - devono offrire funzionalità ricche e complete
  - devono essere caratterizzati da buone qualità – come prestazioni, disponibilità, scalabilità, sicurezza, modificabilità, …
  - in particolare, le qualità del software costituiscono un fattore critico di successo per molti sistemi software

Le sfide dello sviluppo del software

- Inoltre, la complessità dei sistemi software è anche via via crescente
  - questo presenta delle sfide formidabili a chi sviluppa il software
  - se queste sfide non vengono affrontate in modo opportuno, i sistemi realizzati si rivelano dei fallimenti
  - come affrontare al meglio queste sfide?
  - per gestire questa complessità, molti riconoscono oggi l'importanza di un approccio sistematico alla progettazione del software, guidato dall'architettura del software

Architettura del software – strutture

- Che cos'è l'**architettura del software**?
  - l'*architettura di un sistema software* è costituita da un insieme di *strutture*
  - ciascuna struttura comprende a sua volta un certo numero di *elementi* (software e non) – e di *relazioni* tra elementi (anche di strutture diverse)
  - l'architettura del software riguarda dunque gli aspetti di strutturazione e organizzazione del software

Architettura del software – qualità

- Perché l'architettura del software è importante?
  - l'esperienza ha mostrato che l'architettura di un sistema software è il vettore principale delle *qualità* di quel sistema software – che sono un interesse fondamentale per il successo di molti sistemi software
  - ovvero, la strutturazione e l'organizzazione "interna" di un sistema software determinano le qualità "esterne" del sistema

Architettura del software

- La disciplina dell'**architettura del software** è interessata ai sistemi software complessi
  - agli interessi – sia funzionali che di *qualità* – delle diverse parti interessate a un sistema
  - alla *strutturazione fondamentale interna* del sistema
    - quali elementi? con che responsabilità? quali collaborazioni?
  - a comprendere le *relazioni tra la struttura interna di un sistema e le sue qualità esterne*
  - questa conoscenza è infatti fondamentale ai fini dell'analisi, progettazione, valutazione ed evoluzione dei sistemi software
    - in modo che la *progettazione di un sistema software* possa essere guidata dalle qualità fondamentali che esso deve esibire – sulla base dell'applicazione di principi, linee guida e soluzioni derivanti da questa conoscenza

Importanza dell'architettura del software

- Perché questa conoscenza è importante?
  - la strutturazione interna di un sistema software ha un impatto significativo sulle qualità esterne del sistema
  - inoltre, le decisioni in merito alla strutturazione interna sono tra le prime da prendere – e sono anche le più difficili e costose da modificare più tardi

* Sulla progettazione di sistemi complessi

- Abbiamo detto di essere interessati ai sistemi (software) complessi
  - un sistema complesso è un sistema che deve soddisfare numerosi interessi, di diverse parti interessate – interessi funzionali, sociali e ambientali, funzionamento in condizioni specifiche, sicurezza delle persone, ... – nonché altri interessi pragmatici – tempi, costi, ...
  - ma come progettare e realizzare un sistema complesso?

Sistemi e complessità

- *Sistema*
  - un insieme di elementi distinti
  - che sono connessi o correlati
  - e lavorano assieme per realizzare una combinazione significativa di funzionalità e qualità
  - questa combinazione di funzionalità e qualità non può essere realizzata individualmente dai singoli elementi
- *Complesso*
  - composto di parti interconnesse o intrecciate

Affrontare la complessità con le strutture

- Come è possibile affrontare problemi/progettare sistemi caratterizzati da un alto livello di complessità?
  - partizionare (ovvero, decomporre) il problema/sistema via via in unità/parti più piccole e più “semplici”
  - ma questo non basta! – anche le interconnessioni tra le parti devono essere “semplici”
- La scelta delle parti e delle interconnessioni tra di esse definisce la “*struttura*” di un sistema

Interessi e utilità

- In un sistema complesso, la struttura è importante – ma la sua *utilità* è ancora più importante
  - questa utilità dipende dagli obiettivi desiderati – che derivano dalle parti interessate al sistema e dai loro *interessi* – sono importanti soprattutto gli interessi di qualità
  - bisogna saper
    - identificare questi interessi
    - progettare per garantire questi interessi
    - valutare se il sistema progettato può raggiungere effettivamente gli obiettivi desiderati
  - inoltre, la caratterizzazione del problema da affrontare spesso non è fissata o ben definita fin dall'inizio
    - bisogna saper identificare una coppia problema-soluzione per massimizzare la soddisfazione delle parti interessate

- Architettura e ingegneria

- Le due discipline che si occupano della progettazione e costruzione di sistemi complessi sono l'*architettura* e l'*ingegneria*
  - si tratta di due ruoli estremi – in uno spettro continuo – della pratica della progettazione di sistemi complessi
  - è spesso necessario un approccio in parte ingegneristico e in parte architetturale – in parte scienza, in parte arte

Che cosa è l'ingegneria

- **Engineering** [Wikipedia, 2016] is the application of mathematics, empirical evidence and scientific, economic, social, and practical knowledge in order to invent, innovate, design, build, maintain, research, and improve structures, machines, tools, systems, components, materials, and processes.
- A possible definition of "engineering" is:
  - The creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination; or to construct or operate the same with full cognizance of their design; or to forecast their behavior under specific operating conditions; all as respects an intended function, economics of operation or safety to life and property.
- Engineering is a broad discipline which is often broken down into several sub-disciplines. These disciplines concern themselves with differing areas of engineering work.

Che cosa è l'architettura

- **Architecture** [Wikipedia, 2016] is both the process and product of planning, designing and constructing buildings and other physical structures. Architectural works, in the material form of buildings, are often perceived as cultural symbols and as works of art. Historical civilizations are often identified with their surviving architectural achievements.
- "Architecture" can mean:
  - A general term to describe buildings and other physical structures.
  - The art and science of design buildings and some (nonbuilding) structures.
  - The style of design and method of construction of buildings and other physical structures.
  - The knowledge of art, science & technology and humanity.
  - The practice of an architect, ...
  - The design activity of the architect, from the macro-level (urban design, landscape architecture) to the micro-level (construction details and furniture).
- Architecture has to do with planning and designing form, space and ambience to reflect functional, technical, social, environmental and aesthetic considerations. It requires the creative manipulation and coordination of materials and technology, and of light and shadow. Often, conflicting requirements must be resolved. The practice of Architecture also encompasses the pragmatic aspects of realizing buildings and structures, including scheduling, cost estimation and construction administration. Documentation produced by architects, typically drawings, plans and technical specifications, defines the structure and/or behavior of a building or other kind of system that is to be or has been constructed.
- **The word "architecture" has also been adopted to describe other designed systems, especially in information technology.**

Ingegneria e architettura a confronto

- *Ingegneria*
  - riguarda principalmente quantità misurabili
  - uso di strumenti analitici, derivati da matematica e fisica
  - processo deduttivo – procede da postulati e principi primi verso determinazioni più particolari
  - il contesto/problema è compreso
  - ambisce all'ottimizzazione tecnica
  - interessata a costi quantificabili
  - più scienza
- *Architettura*
  - riguarda principalmente qualità non misurabili
  - uso di strumenti e linee guida, basati su esperienze apprese in pratica
  - processo induttivo – procede dall'esperienza, per elaborare leggi astratte e universali
  - il contesto/problema è inizialmente mal strutturato
  - ambisce alla soddisfazione del cliente
  - interessata al valore qualitativo
  - più arte

Progettazione di sistemi complessi

- Metodologie/approcci ingegneristico/architetturali
  - normativo (scienza)
    - basato su soluzioni pre-esistenti – "deve essere così"
    - ad es., realizzare un'ulteriore semplice applicazione web
  - razionale (scienza)
    - basato su principi – consente di produrre soluzioni "innovative"
    - ad es., accoppiamento e coesione
  - partecipativo (arte)
    - riconosce la complessità dovuta alla presenza di una molteplicità di parti interessate – l'obiettivo è il consenso – spesso sono necessari compromessi
    - ad es., modello a tre picchi
  - euristico (arte)
    - basato su euristiche che codificano del "buon senso comune" motivato da esperienza collettiva
    - ad es., pattern, architetture di riferimento, tattiche, …

- Perché un corso di architettura del software?

- Perché un corso di Architettura del software – anziché un corso di Ingegneria del software?
  - questo corso enfatizza più gli aspetti “architetturali” che quelli “ingegneristici”
  - in ogni caso, la disciplina dell’Architettura del software è considerata oggi una branca della disciplina più ampia dell’Ingegneria del software

* Introduzione all’architettura del software

- I grandi sistemi software di oggi sono delle strutture molto complesse – e la loro complessità è via via crescente
  - per gestire questa complessità, molti riconoscono oggi l’importanza di un approccio sistematico alla progettazione del software, guidato dall’architettura del software

Funzionalità e qualità del software

- Gli interessi (le caratteristiche desiderate) per i sistemi software riguardano sia le *funzionalità* che le proprietà che ne riflettono la *qualità* – ad esempio
  - affidabilità
  - disponibilità
  - prestazioni e scalabilità
  - sicurezza
  - modificabilità
  - usabilità, verificabilità, interoperabilità, economicità, risparmio energetico, ...

Qualità del software

- List of system quality attributes [Wikipedia, 2022]
  - accessibility
  - accountability
  - accuracy
  - adaptability
  - administrability
  - affordability
  - agility
  - auditability
  - autonomy
  - availability
  - compatibility
  - composability
  - condifentiality
  - configurability
  - correctness
  - credibility
  - customizability
  - debuggability
  - degradability
  - determinability
  - demonstrability
  - dependability
  - deployability
  - discoverability
  - distributability
  - durability
  - effectiveness
  - efficiency
  - evolvability
  - extensibility
  - failure transparency
  - fault-tolerance
  - fidelity
  - flexibility
  - inspectability
  - installability
  - integrity
  - interchangeability
  - interoperability
  - learnability
  - localizability
  - maintainability
  - manageability
  - mobility
  - modifiability
  - modularity
  - observability
  - operability
  - orthogonality
  - portability
  - precision
  - predictability
  - process capabilities
  - producibility
  - provability
  - recoverability
  - relevance
  - reliability
  - repeatability
  - reproducibility
  - resilience
  - responsiveness
  - reusability
  - robustness
  - safety
  - scalability
  - seamlessness
  - self-sustainability
  - serviceability (a.k.a. supportability)
  - securability
  - simplicity
  - stability
  - standards compliance
  - survivability
  - sustainability
  - tailorability
  - testability
  - timeliness
  - traceability
  - transparency
  - ubiquity
  - understandability
  - upgradability
  - (non) vulnerability
  - usability

Importanza delle qualità del software

- Un fattore critico per il successo di un'organizzazione è la costruzione di sistemi software in grado di soddisfare i requisiti – non solo funzionali, ma anche di qualità – non solo correnti, ma anche futuri – di quell'organizzazione
  - oggi, l'IT non viene usata più solo per automatizzare il funzionamento interno delle organizzazioni, ma anche per ottenere un vantaggio competitivo e per sostenere un'innovazione continua
  - ci sono organizzazioni di successo strettamente legate all'IT, il cui business è centrato sui propri sistemi software – come Amazon, Google e Facebook, ma anche Netflix, Spotify e Uber
  - ma anche molte altre organizzazioni di successo sono "basate sul software", anche se operano in domini applicativi diversi dall'IT – come Ford, FedEx, DreamWorks, …
  - in particolare, i requisiti di qualità rivestono un ruolo fondamentale nel successo di questi sistemi

Che cos'è l'architettura del software?

- Alcune possibili definizioni di "architettura del software"
  - l'*architettura software* di un sistema è l'insieme delle strutture del sistema, necessarie per ragionare su di esso, che comprendono elementi software, le relazioni tra di essi, e le loro proprietà [SAP]
  - l'*architettura software* è il fulcro per i sistemi altamente complessi, su scala grandissima e altamente interoperabili, di cui abbiamo bisogno ora e nel futuro [Rolf Siegers]
  - ma qual è la relazione tra queste due definizioni – che sono apparentemente molto lontane tra di loro?

Benefici dell'architettura del software

- L'architettura del software è importante per molteplici ragioni
  - l'architettura è il vettore principale delle qualità di un sistema software – come prestazioni, modificabilità e sicurezza – nessuna delle quali può essere ottenuta senza una visione architetturale unificante [http://www.sei.cmu.edu/architecture/]
  - l'architettura è il collante concettuale che tiene assieme ogni fase del progetto, per ciascuna delle sue parti interessate
    - nelle fasi iniziali di un progetto, l'analisi dell'architettura consente di garantire che l'approccio di progettazione prescelto conduca a un sistema accettabile
    - l'architettura serve da progetto sia per il sistema che per il piano/progetto relativo al suo sviluppo, e definisce le assegnazioni di lavoro per i team di sviluppo
    - l'architettura ha un ruolo chiave anche per le attività di manutenzione successive al rilascio del sistema

* L'architettura del software come disciplina

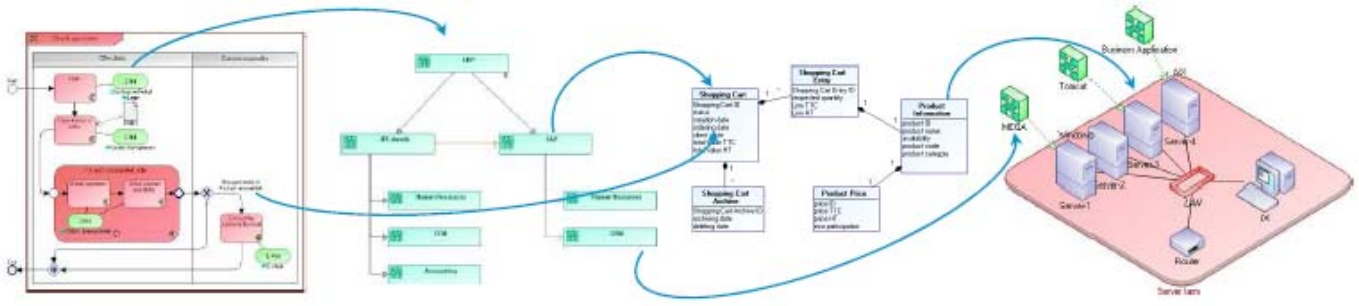
- La disciplina dell'architettura del software
  - riconosce l'importanza degli interessi – sia funzionali che, soprattutto, di *qualità* – di tutte le parti interessate allo sviluppo di un sistema software
  - si occupa della *strutturazione fondamentale* del sistema – ovvero, della decomposizione del sistema in elementi e delle loro inter-relazioni
  - si occupa di comprendere e studiare le *relazioni tra la struttura interna del sistema e le sue qualità esterne*
    - affinché la conoscenza distillata dalle esperienze passate possa essere sfruttata per progettare e sviluppare sistemi che possiedono le qualità richieste

Approccio dell'architettura del software

- Ecco alcune buone "esperienze passate" che è possibile applicare nell'architettura software
  - viste e punti di vista architetturali
  - pattern architetturali e architetture di riferimento
  - tattiche architetturali e prospettive architetturali

Viste e punti di vista architetturali

- L'architettura di un sistema software
  - è costituita da più strutture (o *viste architetturali*) – indipendenti ma correlate



  - ciascuna vista è relativa a un diverso *punto di vista* e si occupa di un insieme distinto di interessi (principio di separazione degli interessi)

Pattern architetturali

- I pattern sono una modalità diffusa di condivisione di esperienze progettuali, in varie discipline
  - in generale, un *pattern software* ha lo scopo di condividere una soluzione provata ed ampiamente applicabile a un particolare problema di progettazione, descritta in una forma standard che possa essere facilmente riusata
  - in particolare, un *pattern architetturale* (o *stile architetturale*) codifica un'esperienza significativa nella realizzazione di un'architettura software

Tattiche e prospettive architetturali

- I pattern architetturali sono utili per identificare una decomposizione *iniziale* del sistema per soddisfare *alcune* qualità più importanti
  - il progetto iniziale deve essere raffinato, in modo che il sistema esibisca *tutte* le qualità richieste
  - la progettazione per alcune qualità richiede spesso delle considerazioni specializzate, che possono riguardare anche più viste
- A tal fine, esistono delle ulteriori opzioni per la progettazione per qualità specifiche di un sistema – tra cui
  - una *tattica architetturale* [SAP] è una decisione di progetto che influenza il controllo di un attributo di qualità
  - una *prospettiva architetturale* [SSA] è una collezione di attività, tattiche e linee guida per far sì che un sistema esibisca un insieme di proprietà di qualità correlate

* Architettura monolitica (e perché evitarla)

- Per avere un'intuizione del perché la strutturazione interna di un sistema software può avere un impatto sulle qualità esterne del sistema, è utile pensare ai sistemi "monolitici" (che non hanno una strutturazione interna)
- Un sistema software è *monolitico* – ha un'*architettura monolitica* – se tutti i suoi aspetti funzionali (come la logica di business, l'interfaccia utente, la gestione dei dati) sono intrecciati e gestiti mediante un singolo elemento – anziché mediante elementi architetturali distinti
  - questa monoliticità può riguardare una o più viste/strutture architetturali – oppure l'intero sistema

Architettura monolitica – conseguenze

- Un sistema monolitico può presentare dei vantaggi
  - nel caso di semplici applicazioni, può essere semplice da sviluppare e rilasciare
  - molte applicazioni vengono inizialmente realizzate in modo monolitico (rispetto a una o più strutture architetturali)
- Tuttavia, l'architettura (completamente) monolitica non è adatta a sistemi complessi – perché può essere difficile o impossibile garantire molte qualità
  - come prestazioni, disponibilità, scalabilità, modificabilità, …
  - quali le intuizioni?

L'architettura monolitica è un anti-pattern

- L'architettura monolitica è considerata un anti-pattern – chiamato **Big Ball of Mud** ("Grossa Palla di Fango")
  - una "Grossa Palla di Fango" è una giungla di "spaghetti code", strutturata a caso, scomposta, sciatta, che si tiene insieme con nastro isolante e filo da imballo
  - questi sistemi mostrano segni inequivocabili di crescita irregolare e di riparazioni ripetute e basate su espedienti
  - le informazioni sono condivise in modo promiscuo tra elementi distanti del sistema, spesso ad un punto tale che quasi tutte le informazioni importanti sono globali o duplicate
  - la struttura complessiva del sistema potrebbe non essere mai stata ben definita – se lo fosse, essa potrebbe essere corrosa e non più riconoscibile
  - i programmatori con un briciolo di sensibilità architetturale fuggono da questi acquitrini: solo coloro che sono indifferenti all'architettura e, forse, si sentono a proprio agio con l'inerzia del compito quotidiano di rattoppare i buchi in queste dighe fallite, si accontentano di lavorare su tali sistemi

Scappare dall'inferno del monolito

- Come evitare questi problemi – e perseguire un insieme di qualità desiderate?
  - è possibile adottare un'architettura non monolitica, ovvero usare un sistema con un'opportuna strutturazione interna
  - ecco alcuni esempi di modalità di organizzazione interna del software, basate su diverse tipologie di elementi software
    - architettura a strati per il codice
    - architettura a plug-in (ad es., per i browser web e gli IDE)
    - architettura a componenti (per applicazioni di tipo enterprise)
    - architettura a microservizi (per applicazioni su scala globale)

* Esempio: Cloud-Native Software

- Da una dozzina di anni, alcune organizzazioni – come Google, Amazon e Microsoft – forniscono dei servizi di *cloud computing*
  - ovvero l'acceso in rete a risorse computazionali – come CPU, storage, reti, sistemi operativi, piattaforme applicative, servizi e/o applicazioni

  - oggi il cloud è una piattaforma di rilascio delle applicazioni sempre più diffusa

Cloud-Native Software

- Il rilascio di un sistema software nel cloud offre diverse opportunità – ma solleva anche delle sfide e dei rischi
  - in particolare, il cloud è una piattaforma di esecuzione scalabile e disponibile
    - ma questo è sufficiente per rendere un sistema software scalabile e disponibile?
  - per perseguire le opportunità offerte dal cloud e affrontare le sfide che pone, è utile progettare e sviluppare il software appositamente per il cloud
    - ad es., applicando un certo numero di pattern software specifici per il cloud
    - si parla in questo caso di *cloud-native software*

Cloud-Native Software

- Ecco alcuni requisiti che possono essere soddisfatti dal software eseguito sul cloud
  - alta disponibilità: nessuna interruzione di servizio
  - scalabilità: accettare un numero di utenti o di richieste crescenti
  - modificabilità: cicli di sviluppo e feedback rapidi (continui)
  - supporto per client mobili e dispositivi di accesso multipli
  - supporto per IoT (Internet of Things)
  - supporto per Big Data

- Esempio – Netflix e microservizi

- Netflix è una società operante nella distribuzione via Internet di film, serie televisive e altri contenuti d'intrattenimento

NETFLIX

  - Netflix ha oltre 163 milioni di abbonati in tutto il mondo (2019) – oltre il 50% di questi sono al di fuori degli USA – negli USA, Netflix detiene oltre il 50% degli abbonati a servizi di streaming
    - questo richiede dei livelli elevati di scalabilità e disponibilità – si pensi che nel 2016 gli abbonati erano circa 75 milioni
  - l'architettura software di Netflix è
    - basata su microservizi
    - rilasciata in container
    - nel cloud (di Amazon AWS)
  - altre organizzazioni che hanno adottato con successo un'architettura simile sono Amazon, Ebay, Groupon, Spotify, Uber, Zalando e Zoom

* Una disciplina in continua evoluzione

- Il concetto di “architettura del software” risale alla fine degli anni Sessanta – la disciplina dell’architettura del software (sia scientifica che pratica) ha poi origini negli anni Novanta
  - nei decenni successivi questa disciplina è certamente maturata – ed è anche evoluta in modo continuo
    - ad es., l’architettura a microservizi sarebbe stata inconcepibile all’inizio degli anni Duemila
    - in quegli anni il cloud computing, i container e la continuous delivery non esistevano ancora, e una decina di anni dopo erano considerati ancora delle soluzioni di nicchia – o fuori dall’interesse di questa disciplina
    - oggi invece sono considerate tecnologie fondamentali – di centrale importanza per questa disciplina

* Discussione

- Nella progettazione dei sistemi software complessi
  - i sistemi devono esibire una certa combinazione di qualità
  - la struttura interna di un sistema ha un’influenza significativa sulle qualità del sistema
  - la disciplina dell’architettura del software studia le relazioni tra la struttura interna dei sistemi software e le loro qualità esterne
  - questa conoscenza è una guida utile all’analisi, progettazione, valutazione ed evoluzione dei sistemi software complessi

Benefici dell'architettura del software

- L'architettura del software è importante per molteplici ragioni
  - un'architettura inibirà o abiliterà il controllo degli attributi di qualità di un sistema
  - l'architettura è un vettore delle decisioni più importanti e più difficili da cambiare
  - l'analisi di un'architettura abilita una valutazione precoce delle qualità di un sistema
  - le decisioni prese in un'architettura permettono di ragionare sui cambiamenti e di gestirli, mentre il sistema evolve
  - un'architettura documentata migliora la comunicazione tra le parti interessate
  - un'architettura definisce un insieme di vincoli sulla successiva implementazione
  - l'architettura influenza la struttura di un'organizzazione, e viceversa
  - un'architettura può definire le basi per una prototipazione evoluzionaria
  - un'architettura è l'elaborato chiave che consente all'architetto e al project manager di ragionare su costi e tempi di sviluppo
  - è possibile creare un'architettura come un modello trasferibile e riutilizzabile che forma il cuore di una famiglia di prodotti
  - lo sviluppo basato sull'architettura focalizza l'attenzione sull'assemblaggio di componenti, piuttosto che semplicemente della loro creazione
  - riducendo le alternative di progettazione, l'architettura veicola la creatività degli sviluppatori, riducendo la complessità del progetto e del sistema
  - un'architettura può essere la base per formare un nuovo membro di un team