

Luca Cabibbo  
Architettura  
dei Sistemi  
Software

# Architettura del software: definizione e concetti

**dispensa asw120**  
ottobre 2022

*Architecture is about the important stuff...  
whatever that is.*

*Ralph Johnson*



## - Riferimenti

- ❑ Luca Cabibbo. **Architettura del Software: Strutture e Qualità**. Edizioni Efestò, 2021.
  - Capitolo 2, **Architettura del software: definizione e concetti**
- ❑ [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
- ❑ Perry, D.E. and Wolf, A.L. **Foundations for the Study of Software Architecture**. ACM Software Engineering Notes, 1992.
- ❑ ISO/IEC/IEEE 42010:2011. **Systems and Software Engineering – Architecture Description**. 2011.
- ❑ Conway, M. E. **How do committees invent?** Datamation, 1968.



## - Obiettivi e argomenti

### □ Obiettivi

- introdurre i concetti fondamentali dell'architettura del software

### □ Argomenti

- architettura del software – definizioni preliminari
- architettura del software – definizione e concetti
- ulteriori concetti
- discussione



## \* Architettura del software – definizioni preliminari

- L'architettura del software riguarda le strutture e le qualità dei sistemi software

- ma come definire l'“architettura del software”?

- Il sito del SEI (Software Engineering Institute@Carnegie Mellon) elenca diverse dozzine di definizioni del termine *software architecture* – *architettura del software* o *architettura software*

- <http://www.sei.cmu.edu/architecture/start/glossary/index.cfm>
- ne presentiamo ora più di una, per enfatizzare diversi aspetti salienti dell'architettura del software



## Software architecture – alcune definizioni

- Taylor, Medvidovic, and Dashofy
  - a **software system's architecture** is the set of principal design decisions made about the system
  - design decisions encompass every facet of the system
  - “principal” implies a degree of importance that grants a design decision “architectural status” – it implies that not all design decisions are architectural
- Grady Booch
  - **architecture** represents the significant design decisions that shape a system – where “significant” is measured by cost of change
- Eoin Woods
  - **software architecture** is the set of design decisions which, if made incorrectly, may cause your project to be cancelled



## Software architecture – alcune definizioni

- [Perry and Wolf, 1992]
  - **software architecture** = { *elements*, *form*, *rationale* }
  - un'architettura software è
    - un insieme di *elementi* architeturali – di tre tipi
      - *data elements* – contengono le informazioni da gestire
      - *processing elements* – implementano le funzionalità e le trasformazioni desiderate
      - *connecting elements* – il collante che tiene insieme i diversi pezzi dell'architettura
    - utilizzati secondo una particolare *forma* (nel senso di organizzazione, strutturazione)
    - insieme a una *giustificazione logica* che ha lo scopo di cogliere e rendere esplicita la motivazione per la scelta degli elementi e della forma



## Software architecture – alcune definizioni

### □ [SAP]

- the **software architecture** of a system is the set of structures needed to reason about the system; these structures comprise software elements, relations among them, and properties of both



## Software architecture – alcune definizioni



### □ [ISO-42010], adottata anche da [SSA]

- **architecture** – the fundamental concepts or properties of a system in its environment, embodied in its elements, relationships, and in the principles of its design and evolution

### □ [Boehm et al., 1995]

- **a software system architecture** comprises
  - a collection of software and system components, connections, and constraints
  - a collection of system stakeholders' need statements
  - a rationale which demonstrates that the components, connections, and constraints define a system that, if implemented, would satisfy the collection of system stakeholders' need statements



## \* Architettura del software – definizione e concetti

- Facciamo riferimento principalmente alla definizione di [SAP]
  - nella definizione, *sistema* va inteso come *sistema software* – o *sistema software intensive*
    - un sistema in cui il software ha un ruolo essenziale nella progettazione, costruzione, rilascio ed evoluzione del sistema nel suo complesso
- 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



## - Elementi

- *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*



# Elementi

- Un'architettura software comprende/definisce/è composta da un insieme di **elementi**
  - l'alternativa è un sistema **monolitico** – con tutti i suoi inconvenienti
  - un'architettura è un'astrazione del sistema – che enfatizza gli elementi del sistema e le loro relazioni
    - un **elemento architetturale (elemento)** è una “parte” fondamentale che costituisce un sistema
  - gli elementi possono essere di varia natura e di diversi tipi
    - **elementi software** – ad es., un modulo, un processo, un componente software, un servizio, una base di dati, ...
    - **elementi non software** – ad es., un'unità di deployment fisica o virtuale, un team di sviluppo, ...



# Caratteristiche degli elementi software

- Caratteristiche rilevanti di un **elemento software**
  - un insieme ben definito di **responsabilità** – con un **confine** (o **portata**) ben definito
    - quello che l'elemento fa e come lo fa
  - un'**interfaccia** (o un insieme di **interfacce**) ben definita
    - i **servizi** che l'elemento fornisce (oppure richiede) agli altri elementi
    - il livello di **qualità** con cui ciascun servizio viene fornito



## Componenti e connettori

- È comune distinguere tra due tipi principali di elementi software
  - **componenti** – responsabili dell’implementazione di **funzionalità** e della gestione di **dati** (responsabilità di business)
  - **connettori** – responsabili delle **interazioni** tra componenti (responsabilità infrastrutturali)
- ci sono buoni motivi per trattare separatamente i connettori dai componenti – lo capiremo meglio nel seguito del corso
- in alcune architetture, componenti e connettori sono elementi architeturali di “primo livello” – in altre architetture, più moderne, ci sono elementi software di “primo livello” che sono internamente decomposti in componenti e connettori, che sono dunque elementi architeturali di “secondo livello”
  - in ogni caso, la distinzione tra componenti e connettori è in genere rilevante



## - Strutture

- *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*



# Strutture

- Due nozioni distinte ma correlate: struttura e vista
  - una **struttura** comprende un insieme di *elementi* e di *relazioni* tra questi elementi
  - una **vista** è la *descrizione* di una struttura – ovvero, un *modello* che descrive gli elementi della struttura e le loro relazioni
    - ad es., un diagramma di (UML) – package, componenti, interazione, di deployment, ...
- I termini *struttura* e *vista* vengono però usati spesso in pratica in modo intercambiabile



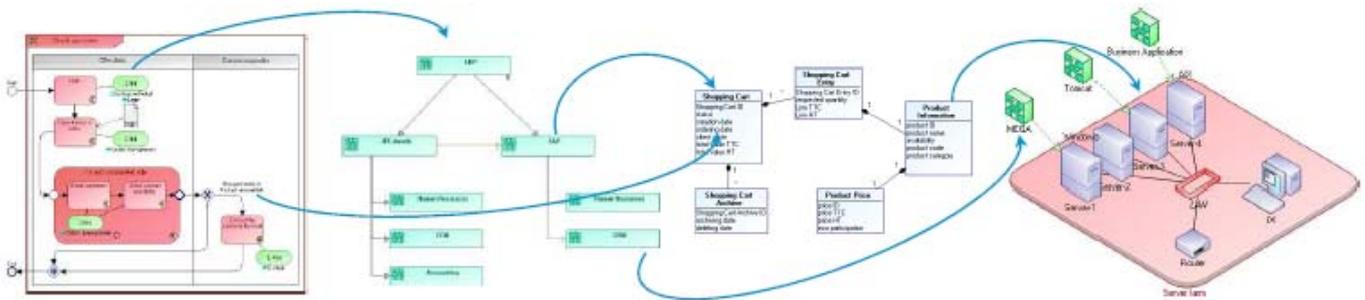
# Strutture

- Un'architettura comprende di solito *più strutture*
  - ad esempio
    - una struttura statica
    - una struttura dinamica
    - una struttura di deployment
    - una struttura di sviluppo
  - alcune strutture comprendono solo elementi software, mentre altre strutture comprendono sia elementi software che elementi non software



# Strutture

- Un'architettura comprende di solito *più strutture*



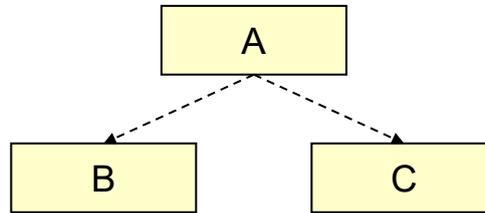
# Strutture

- Un'architettura comprende di solito *più strutture* (e non una sola)
  - una singola struttura non è in grado di descrivere, in modo completo, l'intera architettura di un sistema complesso
  - ciascuna struttura comprende solo alcuni elementi (e solo di alcuni tipi di elementi) e ne definisce un particolare arrangiamento
  - strutture diverse comunicano informazioni differenti
  - ciascuna struttura affronta aspetti o interessi diversi



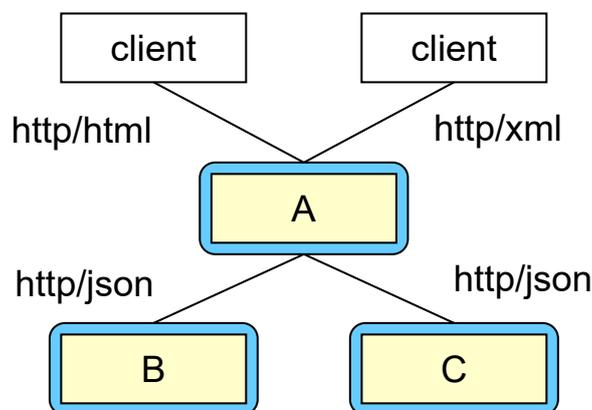
## - Esempio

- Consideriamo una semplice applicazione web composta da tre moduli A, B e C



## Esempio

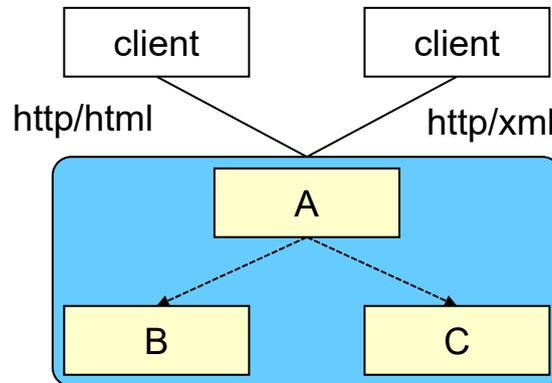
- L'applicazione web è organizzata in più *processi* separati – che comunicano in vari modi





## Esempio

- L'applicazione web è organizzata in più *processi* separati – che comunicano in vari modi



21

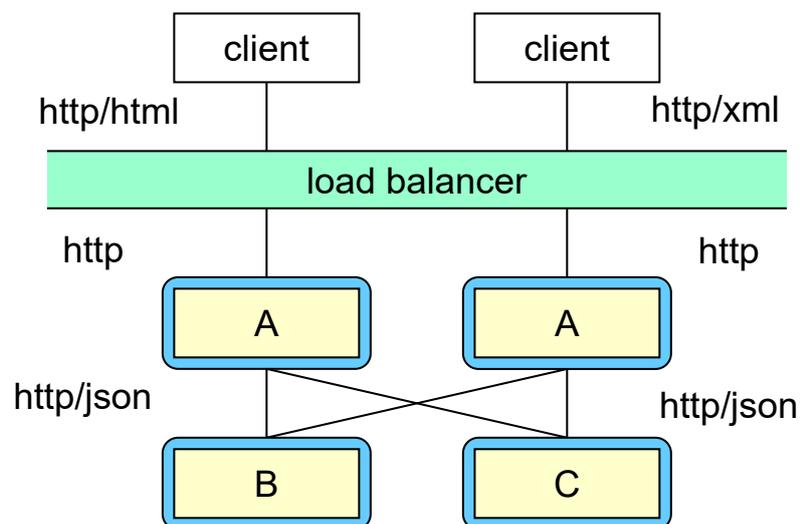
Architettura del software: definizione e concetti

Luca Cabibbo ASW



## Esempio

- L'applicazione web è organizzata in più *processi* separati – che comunicano in vari modi



22

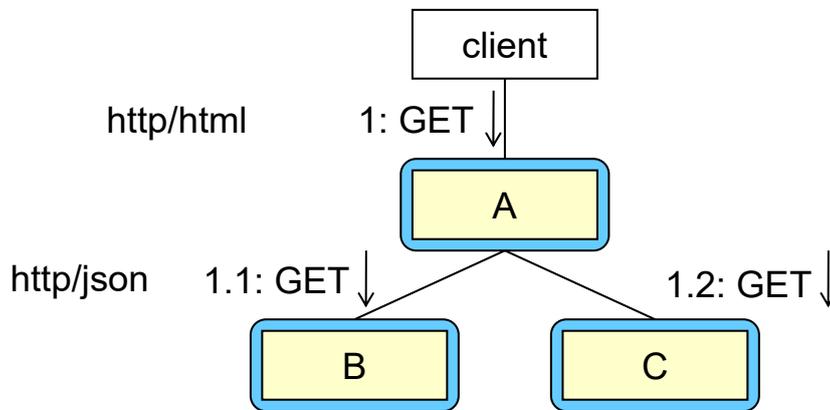
Architettura del software: definizione e concetti

Luca Cabibbo ASW



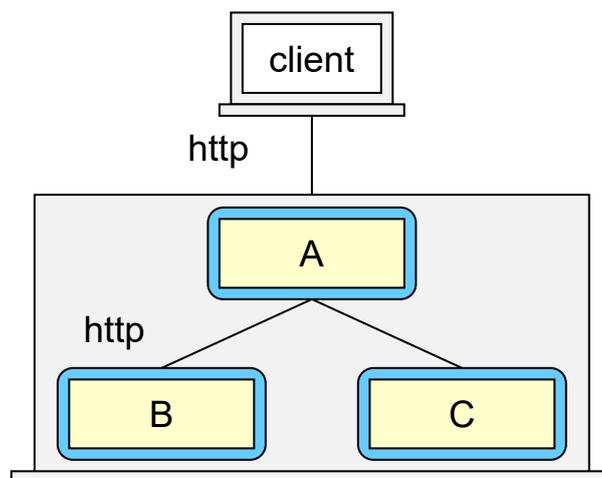
## Esempio

- Un possibile scenario nell'esecuzione dell'applicazione web
  - come viene gestita una richiesta da parte di un client?



## Esempio

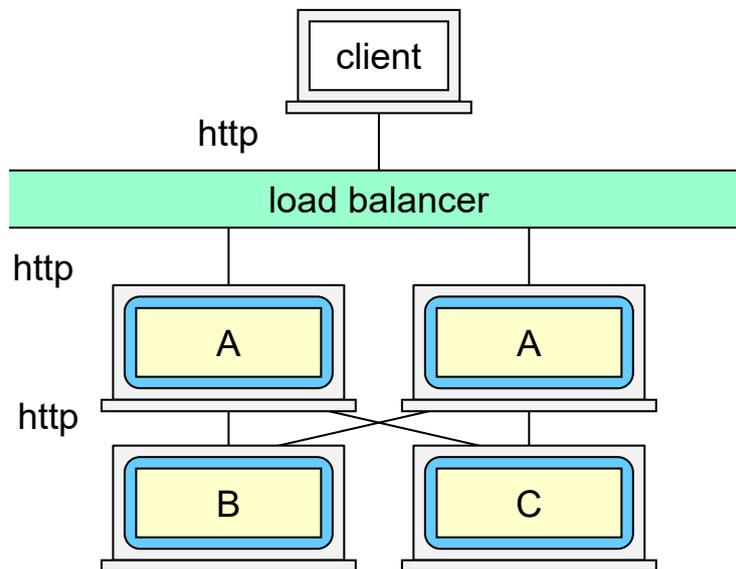
- Una possibile struttura di deployment – in cui i processi sono distribuiti sui *nod*i di elaborazione





## Esempio

- Un'altra possibile struttura di deployment – i processi sono distribuiti sui *nod*i di elaborazione in modo differente



## - Relazioni

- 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*



# Relazioni

- Relazioni di interesse in un'architettura
  - i connettori (sono un tipo di elementi software) hanno lo scopo di descrivere le interazioni (sono un tipo di relazione) tra componenti software (sono elementi software)
  - una vista/struttura comprende sempre le relazioni tra gli elementi che vi compaiono
  - relazioni tra strutture, ovvero tra elementi presenti in strutture diverse (comprese le relazioni tra elementi di natura diversa)



# - Proprietà

- *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***



## Proprietà

- L'architettura del software è interessata non solo agli elementi – ma anche alle relazioni/interazioni/collaborazioni tra gli elementi
  - queste collaborazioni sono basate esclusivamente sul comportamento *all'interfaccia* degli elementi – ovvero sulle loro “proprietà” (che sono visibili esternamente dagli elementi)
- Le **proprietà** sono le ipotesi che ciascun elemento può fare circa gli altri elementi
  - due tipologie di proprietà degli elementi software
    - il *comportamento (funzionalità)*, che definisce le interazioni funzionali tra un elemento software e il suo ambiente
    - le *proprietà di qualità (qualità)*, ovvero le proprietà non funzionali di un elemento che sono percepite dagli altri elementi
  - queste proprietà possono essere relative sia ai componenti che ai connettori



## - Ragionare sul sistema

- *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*



## Ragionare sul sistema

- Abbiamo parlato delle proprietà degli elementi interni di un sistema
  - ma quello che interessa davvero sono soprattutto le **proprietà complessive del sistema**
  - l'architettura del software è interessata a comprendere come gli elementi interni del sistema, con certe loro proprietà e sulla base di una certa organizzazione interna (le strutture), contribuiscono alle proprietà (complessive, esterne) dell'intero sistema



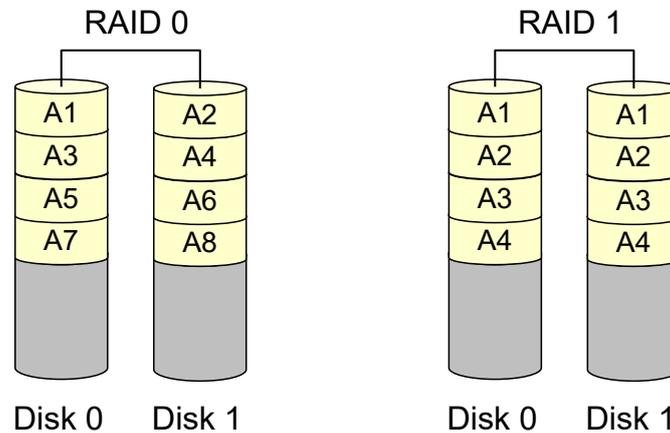
## Organizzazione interna e proprietà esterne

- Relazione tra organizzazione interna e proprietà esterne
  - il **comportamento complessivo** di un sistema è determinato dal comportamento funzionale combinato dei suoi elementi interni
  - anche le **qualità complessive** di un sistema derivano dalle qualità dei suoi elementi interni
  - spesso, uno stesso insieme di elementi può essere organizzato in modi diversi (eventualmente aggiungendo qualche elemento e/o cambiando le relazioni tra elementi) – e dar luogo a un sostegno differente alle qualità del sistema



## Organizzazione interna e proprietà esterne

- Esulando un momento dal software, pensiamo alle configurazioni RAID per i dischi
  - le configurazioni RAID sono ottenute dagli stessi elementi – un certo numero di dischi e un controller RAID
  - configurazioni diverse danno luogo a qualità esterne diverse



## Organizzazione interna e proprietà esterne

- Anche nei sistemi software sussistono relazioni significative tra organizzazione interna e proprietà esterne dei sistemi software
  - ecco alcune intuizioni
    - se il sistema richiede tolleranza ai guasti, allora gli elementi che possono guastarsi vanno replicati
    - se la modificabilità del sistema è importante, allora le responsabilità vanno assegnate agli elementi in modo tale che ciascun cambiamento atteso abbia effetto su uno o pochi elementi
    - se il sistema deve essere altamente sicuro, allora la comunicazione tra gli elementi va gestita e protetta opportunamente
    - se sono importanti le prestazioni e la scalabilità, allora il lavoro da svolgere va decomposto tra più elementi, eventualmente replicati



# Architettura del software

- La disciplina dell'architettura del software
  - ha come obiettivo fondamentale lo studio e la comprensione delle relazioni tra le strutture interne e le qualità esterne dei sistemi software – e, più in generale, dell'impatto delle decisioni di progetto sulle qualità complessive di un sistema software
  - questa comprensione può essere usata
    - per guidare attivamente la progettazione di un'architettura, perseguendo un certo insieme di obiettivi di qualità
    - più in generale, per guidare l'analisi, la progettazione, la valutazione e l'evoluzione dei sistemi software complessi



## - Ulteriori osservazioni



- [ISO-42010] adottata anche da [SSA]
  - **architecture** – the fundamental concepts or properties of a system in its environment, embodied in its elements, relationships, and in the principles of its design and evolution
- [Boehm et al., 1995]
  - **a software system architecture** comprises
    - a collection of software and system components, connections, and constraints
    - a collection of system stakeholders' need statements
    - a rationale which demonstrates that the components, connections, and constraints define a system that, if implemented, would satisfy the collection of system stakeholders' need statements



- Non tutte le architetture sono buone architetture
  - la definizione di architettura software non distingue tra architetture buone e non buone
  - è importante anche saper valutare le architetture
  
- Ogni sistema software ha un'architettura
  - anche se questa non è stata documentata
  - in alcuni casi, l'architettura effettiva di un sistema è diversa da quella che è stata documentata
  - questi sono di solito dei cattivi indicatori – a cui si può ovviare mediante un'attività di ricostruzione dell'architettura



## \* Ulteriori concetti

- È utile introdurre brevemente degli ulteriori concetti correlati all'architettura del software
  - parti interessate
  - interessi
  - architetturelmente significativo
  - processo di definizione dell'architettura
  - qualità e compromessi
  - descrizioni architetturelali
  - l'architetto e il suo ruolo
  - team
  - legge di Conway (Conway's Law)



## - Parti interessate

- Ogni sistema software viene realizzato per soddisfare gli interessi di un certo numero di parti interessate
  - una **parte interessata** (**stakeholder**, “portatore di interessi”) è un individuo, un gruppo di persone o un’organizzazione che ha interessi nel sistema [ISO-42010, SSA]
  - le parti interessate sono importanti, perché
    - il sistema software viene creato solo per soddisfare i loro bisogni
    - solo loro possono definire il successo (o meno) del sistema



## - Interessi

- Ogni sistema software viene realizzato per soddisfare gli interessi di un certo numero di parti interessate
  - un **interesse** (*concern*) su un sistema (o su un’architettura) è un requisito, un obiettivo, un vincolo, un’intenzione o un’aspirazione che una parte interessata ha per quel sistema (o architettura) [SSA]
  - si tratta di una caratterizzazione piuttosto ampia
  - alcuni interessi sono specifici, non ambigui e misurabili – in tal caso vengono chiamati *requisiti*
  - l’architettura si dovrebbe concentrare soprattutto sugli interessi più rilevanti – e non solo sui requisiti più specifici



## - Architetaturalmente significativo

- In un sistema software, non tutti gli interessi e non tutte le decisioni di progetto sono ugualmente importanti per la sua architettura – alcuni sono più rilevanti, altri meno
  - l'architettura deve focalizzare l'attenzione su ciò che è significativo per l'architettura – ovvero, su ciò che viene detto “architetturalmente significativo”
  - un interesse, un requisito, un problema, un elemento del sistema o una decisione di progetto è **architetturalmente significativo/a** [SSA] se ha un impatto ampio sulle strutture del sistema oppure sulle sue qualità più importanti
  - un **requisito** è **architetturalmente significativo** [SAP] – **ASR**, *architetturally significant requirement* – se ha/avrà un effetto profondo sull'architettura



## - Processo di definizione dell'architettura

- La *definizione dell'architettura* [SSA] è un **processo** con cui
  - vengono colti gli interessi e i bisogni delle parti interessate,
  - viene progettata un'architettura che soddisfa questi interessi,
  - e l'architettura viene descritta in modo chiaro e non ambiguo mediante una descrizione architeturale
- Non è solo progettazione
  - è un'attività a cavallo tra la comprensione degli interessi e la progettazione – che si influenzano reciprocamente
  - adotta un processo di natura iterativa
  - richiede di saper valutare l'architettura



## - Qualità e compromessi

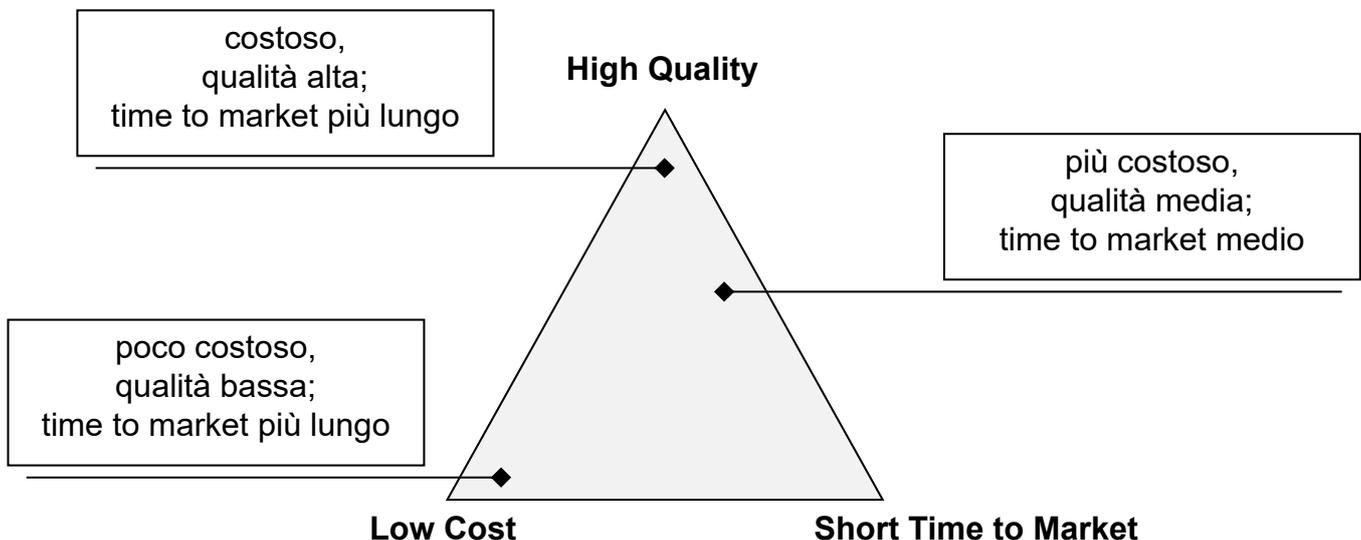
- Una fonte di complessità nella definizione dell'architettura è legata al fatto che alcuni interessi possono essere tra loro contrastanti
  - molte decisioni architettoniche richiedono degli opportuni **compromessi** (*tradeoff*) nella gestione delle qualità
  - è comune che i requisiti di qualità vengano rinegoziati durante il processo di definizione dell'architettura



## Qualità contrastanti e compromessi

- Il triangolo della qualità

Faster, better, cheaper – choose two  
(you can't have all three at once)





## - Descrizioni architetture

- Un risultato del processo di definizione dell'architettura è una descrizione dell'architettura del sistema software
  - una *descrizione architetture* è un insieme di prodotti che documentano un'architettura
    - un insieme di modelli architetture (viste)
    - una descrizione degli interessi, dei vincoli, dei principi rilevanti e delle scelte di progettazione
    - una giustificazione logica dell'architettura
- Una buona descrizione architetture
  - è la base per le decisioni iniziali di progetto e la loro analisi
  - sostiene la comunicazione con le parti interessate
  - è una guida per lo sviluppo e l'evoluzione del sistema



## - L'architetto e il suo ruolo

- L'*architetto* è chi progetta, documenta e guida la costruzione di un sistema – in modo che esso soddisfi gli interessi di tutte le sue parti interessate [SSA]
  - un buon architetto deve saper
    - identificare le parti interessate e cogliere i loro interessi
    - operare ove necessario per riconciliare interessi contrastanti – mediante opportuni compromessi
    - prendere decisioni
    - comunicare le proprie decisioni alle parti interessate
    - promuovere un cambiamento architetture, tecnologico, metodologico e culturale



## - Team

- L'architettura è interessata anche alla gestione delle persone e dei team – gli elementi “umani” dell'architettura
  - sono di interesse soprattutto Dev e Ops
  - *Dev (development)* indica gli *sviluppatori (developer)* del software
  - *Ops (operations)* indica gli *operatori (operator)* o “amministratori” – che gestiscono l'ambiente di produzione e il rilascio del software in questo ambiente
    - il termine *operations*, praticamente privo di traduzione in italiano, si riferisce a tutte quelle funzioni nella messa a disposizione per il cliente di un certo prodotto o servizio [Wikipedia]



## Team

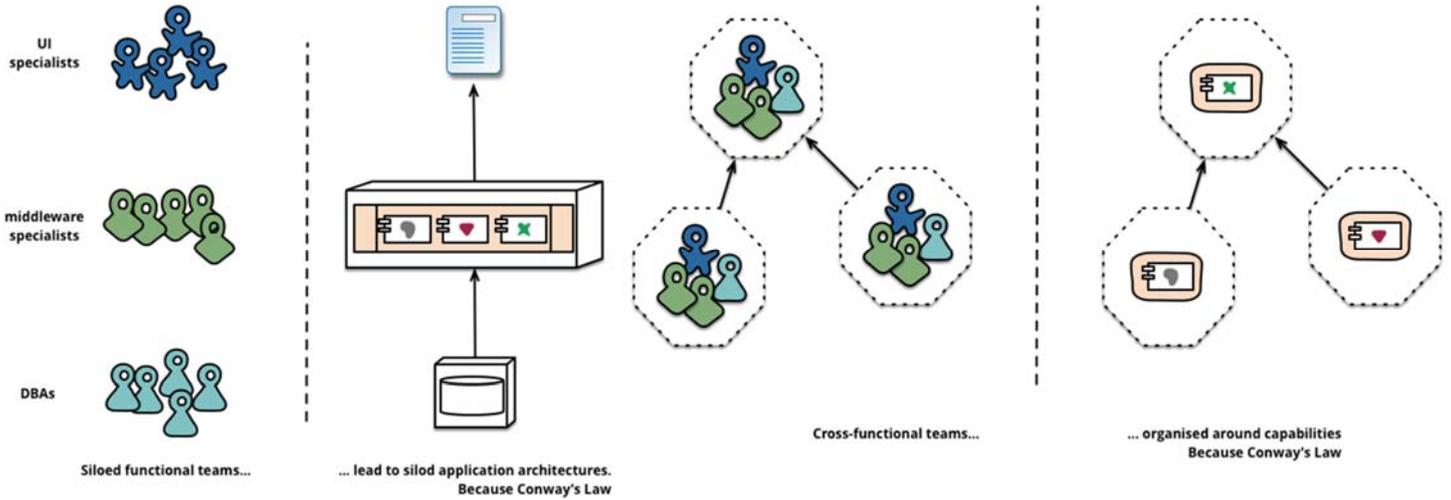
- Organizzazione dei team, assegnazione di lavoro ai team e coordinamento tra team sono parte dell'architettura
  - ci sono diverse modalità di organizzazione dei team di sviluppo
    - team mono-funzionali
    - team cross-funzionali
  - l'assegnazione di lavoro degli elementi software ai team deve essere coerente con l'organizzazione dei team
  - le dipendenze tra elementi software assegnati ai team costituiscono anche dipendenze tra team – poiché implicano delle necessità di coordinamento tra team
  - il costo e lo sforzo richiesti dal coordinamento inter-team è di solito (molto) più alto che nel coordinamento intra-team – pertanto, è in genere opportuno mantenere basso l'accoppiamento tra team



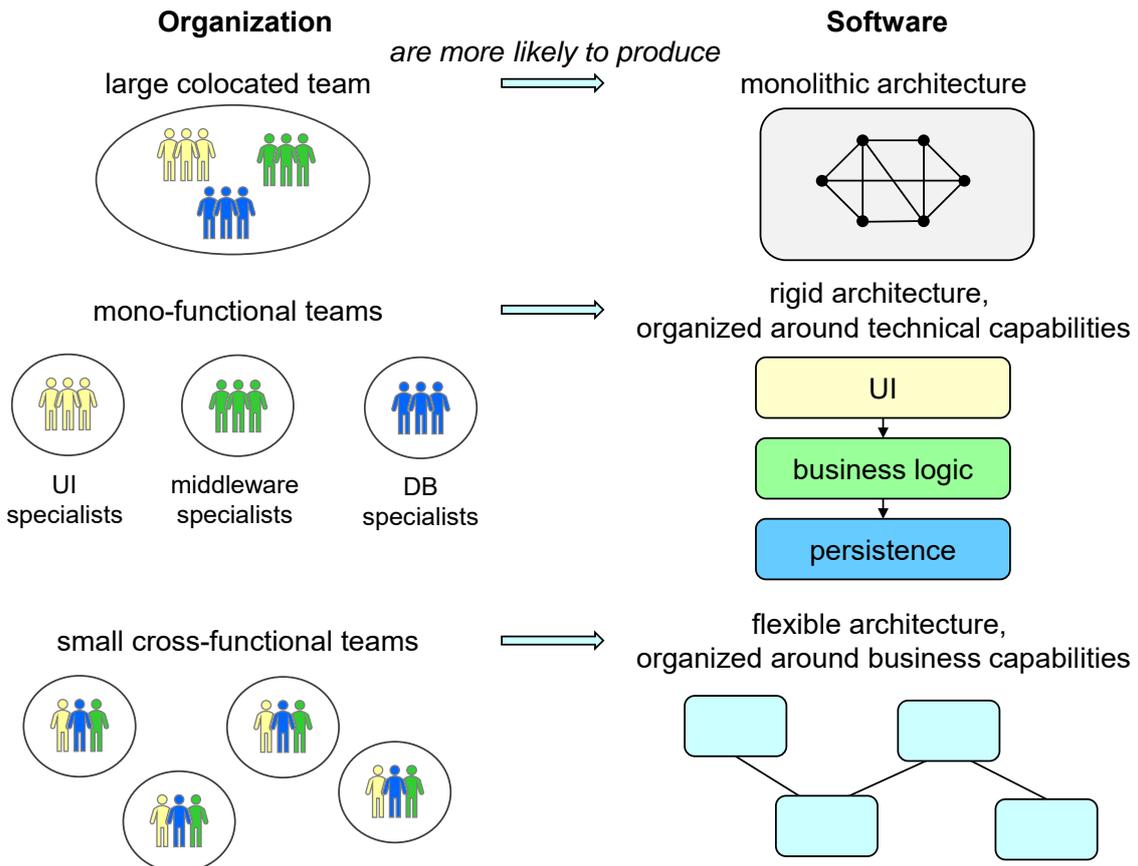
# - Legge di Conway (Conway's Law)

## ▣ Legge di Conway (Conway's Law, 1968)

- ▣ ogni organizzazione che progetta sistemi (in senso più generale che i soli sistemi software) produrrà inevitabilmente dei progetti la cui struttura è una copia della struttura di comunicazione dell'organizzazione



# Legge di Conway (Conway's Law)





## Legge di Conway (Conway's Law)

- Si tratta di una legge empirica, che è tornata recentemente di moda
  - il modo in cui un'organizzazione organizza i propri team per lo sviluppo del software ha un effetto significativo sul software che viene prodotto e sulla sua architettura – e dunque sulle qualità del software
  - c'è anche una legge “inversa” di Conway – la struttura dei team di un'organizzazione è determinata dall'architettura dei sistemi che produce
  - è bene cercare di usare la legge di Conway (e la sua inversa) a proprio vantaggio, per avere team che lavorino in modo indipendente tra di loro



## \* Discussione

- La seguente definizione – adattata da [McGovern, 2004] – riassume molti dei concetti presentati finora
  - the **software architecture** of a system consists of all the *important design decisions* about the software *structures* and *elements* and the *interactions* between those structures and elements that comprise the systems
  - the design decisions support a desired set of *qualities* that the system should support to be successful
  - the design decisions provide a conceptual basis for system development, support, and maintenance



## Discussione



- Alcune attività legate all'architettura del software
  - comprensione degli interessi e dei requisiti
  - creazione/progettazione dell'architettura
  - descrizione dell'architettura
  - analisi e valutazione dell'architettura
  - comunicazione dell'architettura
  - implementazione del sistema – guidati dall'architettura
  - ricostruzione dell'architettura



## Discussione



- Vantaggi nel progettare esplicitamente un'architettura software
  - controllare attributi di qualità – presto nel ciclo di vita del sistema
  - comprendere i compromessi tra qualità ed effettuare scelte di compromesso
  - avere una giustificazione logica per le scelte di progetto
  - ridurre la probabilità di fallimento del progetto
  - comunicare con tutte le parti interessate
  - ragionare sui cambiamenti e gestirli
  - predire e mitigare rischi
  - ...



# Concetti e relazioni fondamentali

