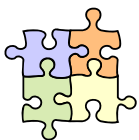


Architetture Software

Architetture software: Concetti

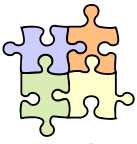
Dispensa ASW 120
ottobre 2014

*Come raggiungere un traguardo?
Senza fretta ma senza sosta.
Johann Wolfgang Goethe*



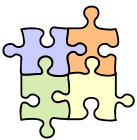
- Fonti

- [SAP] Chapter 1, What is Software Architecture?
- [SSA] Chapter 2, Software Architecture Concepts
- [Perry&Wolf] D.E. Perry, A.L. Wolf, Foundations for the Study of Software Architecture, ACM Software Engineering Notes, 1992
- [ISO-42010] Systems and Software Engineering – Architecture Description, ISO/IEC/IEEE, 2011



* Architetture software

- Le architetture software riguardano le qualità e le strutture dei grandi sistemi software
 - le parti interessate a un sistema desiderano alcune qualità
 - il sistema può essere decomposto in elementi/parti/sottosistemi
 - si tratta di elementi a grana grossa
 - ogni sistema può essere opportunamente decomposto in un numero (relativamente piccolo) di elementi
 - in effetti, per un sistema sono spesso possibili (e di interesse) più decomposizioni – relative a strutture diverse
 - la definizione dell'architettura è interessata (almeno) a
 - gli elementi del sistema
 - le relazioni tra questi elementi
 - le relazioni tra le strutture
 - l'architettura influisce sul raggiungimento delle qualità desiderate dalle parti interessate al sistema



Architettura software - alcune definizioni

- Il sito del SEI (Software Engineering Institute@Carnegie Mellon) elenca diverse dozzine di definizioni del termine “architettura software”
 - <http://www.sei.cmu.edu/architecture/start/glossary/community.cfm>
 - ne presentiamo alcune tratte dalla letteratura



Architettura software - alcune definizioni

- Una delle prime caratterizzazioni (1992)

[Perry&Wolf]

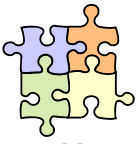
- **software architecture** = { *elements*, *form*, *rationale* }
- un'architettura software è
 - un insieme di *elementi* architeturali
 - utilizzati secondo una particolare *forma* (nel senso di organizzazione, strutturazione)
 - insieme a una *giustificazione logica* che coglie la motivazione per la scelta degli elementi e della forma
- ad esempio, che cosa sapete dell'architettura a strati?
- che cosa invece non sapete dell'architettura a strati?



Architettura software - alcune definizioni

- Inoltre, secondo [Perry&Wolf]

- tre tipi di *elementi architeturali*
 - *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 – ad es., chiamate di procedure locali, chiamate di procedure remote, o scambio di messaggi
- la *forma* cattura il modo in cui gli elementi sono organizzati nell'architettura – come sono composti, e le relazioni e interazioni tra di essi
- la *giustificazione logica* ha lo scopo di rendere esplicite le motivazioni per la scelta degli elementi e della forma – in particolare, con riferimento al modo in cui queste scelte consentono di soddisfare i requisiti/interessi del sistema



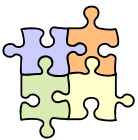
Architettura software - alcune definizioni

 [SAP]

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

 [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



Architettura software - alcune definizioni

 Eoin Woods

- **software architecture** is the set of design decisions which, if made incorrectly, may cause your project to be cancelled

 Grady Booch

- **architecture** represents the significant design decisions that shape a system – where *significant* is measured by cost of change

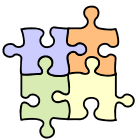
 Taylor, Medvidovic, Dashofy

- a **software system's architecture** is the set of principal design decisions made about the system
- design decisions encompass every facet of the system – structure, behavior, interaction, non-functional properties, ...
- “principal” implies a degree of importance that grants a design decision “architectural status” – it implies that not all design decisions are architectural



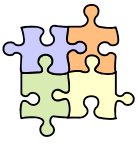
* Architettura software - concetti

- Nel seguito faremo riferimento principalmente alla definizione di [SAP], con una precisazione
 - nella definizione, “sistema” va inteso come *sistema software intensive*
 - ovvero, 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



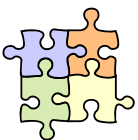
- Elementi

- 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



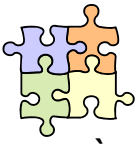
Elementi

- Un'architettura software comprende/definisce degli **elementi**
 - un'architettura è un'astrazione del sistema – che enfatizza gli elementi del sistema e le loro interazioni
 - un **elemento architetturale** (**elemento**) è un pezzo fondamentale che costituisce un sistema
 - gli elementi di interesse per un'architettura possono essere di varia natura
 - **elementi software** – ad es., un modulo (ovvero, codice), un processo, un componente EJB o .NET, un web service, una base di dati, ...
 - **elementi non software** – di varia natura, ad es., un'unità di deployment, un team di sviluppo, ...



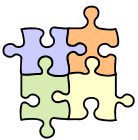
Caratteristiche degli elementi software

- Alcune caratteristiche rilevanti di un **elemento software** sono
 - un insieme di **responsabilità**
 - una caratterizzazione di quello che l'elemento fa – e di quello che non fa – e di come lo fa
 - un'**interfaccia** – definisce i **servizi** che l'elemento fornisce agli altri elementi
 - in termini di operazioni fornite o richieste, oppure di messaggi che è in grado di accettare o produrre
 - anche con riferimento al livello di **qualità** con cui ciascun servizio viene fornito



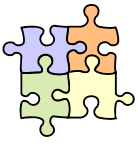
Componenti e connettori

- È comune distinguere tra due tipi principali di elementi software
 - **componenti** – elementi responsabili dell'implementazione di **funzionalità** e della gestione di **dati**
 - ad es., un modulo, un processo, una base di dati, ...
 - **connettori** – elementi responsabili delle **interazioni** tra componenti
 - caratterizzano assemblaggio e integrazione di componenti
 - ad es., una chiamata di procedura remota, un canale di messaging, un protocollo, ...
- come vedremo nel seguito del corso, ci sono buoni motivi per trattare separatamente i connettori dai componenti
 - ad es., la responsabilità per alcune qualità del sistema può essere attribuita ai connettori – inoltre i connettori sono riutilizzabili in più contesti, sotto forma di middleware



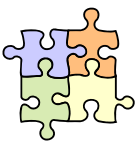
- Strutture del sistema

- 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



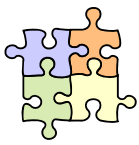
Strutture del sistema

- [SAP] identifica due nozioni distinte ma correlate: struttura e vista
 - una **struttura** comprende un insieme di *elementi*, nonché le *relazioni* tra questi elementi
 - ad es., un insieme di processi – e la relativa modalità di comunicazione interprocesso
 - ad es., un insieme di moduli – e le relazioni tra moduli
 - attenzione – in alcuni casi i connettori sono considerati “elementi”, in altri casi “relazioni tra elementi”
 - una **vista** è la *descrizione* di una struttura – ovvero, un modello – in termini dei suoi elementi e delle relazioni tra di essi
 - ad es., un diagramma UML – in cui ciascun elemento viene rappresentato mediante un opportuno simbolo grafico
- Tuttavia, in pratica i termini *struttura* e *vista* vengono spesso usati in modo intercambiabile



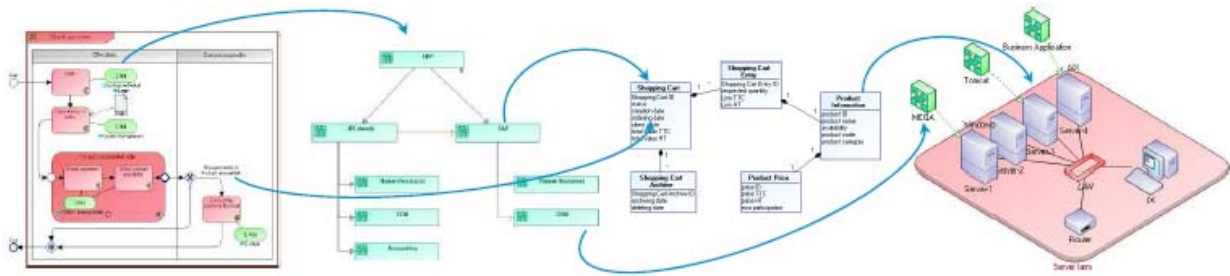
Strutture del sistema

- Un'architettura può comprendere (e di solito comprende) *più strutture*
 - ad esempio
 - una struttura (statica) può riguardare l'organizzazione del codice (moduli)
 - un'altra struttura (dinamica) può riguardare i processi in esecuzione
 - un'altra struttura (deployment) può riguardare i nodi di calcolo in cui sono esecuzione tali processi
 - un'altra struttura ancora (sviluppo), può riguardare l'assegnazione di lavoro dei moduli ai team di sviluppo
 - si noti che alcune strutture comprendono solo elementi software, mentre altre strutture possono comprendere anche elementi non software

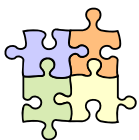


Strutture del sistema

- Un'architettura può comprendere (e di solito comprende) **più strutture**

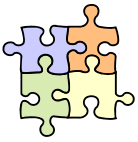


- la figura mostra le viste relative a quattro strutture: processi di business, funzionale, informazioni, deployment
- da notare, all'interno di ogni vista, l'uso di elementi e di relazioni tra elementi
- da notare anche l'importanza delle relazioni tra elementi di viste diverse



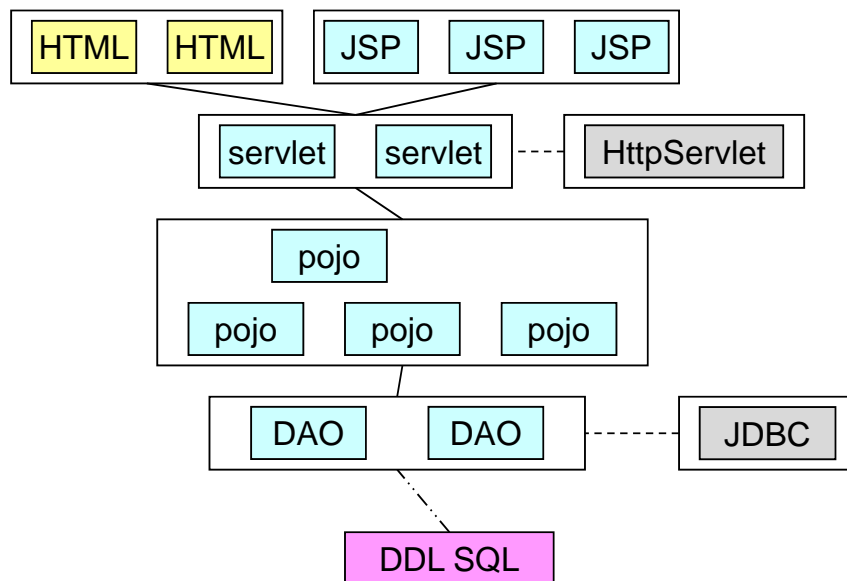
Strutture del sistema

- Un'architettura comprende di solito **più strutture** (e non una sola)
 - in generale, per i sistemi complessi, una singola struttura non è in grado di descrivere, in modo completo, l'intera architettura del sistema
 - ciascuna struttura comprende solo alcuni elementi (e solo di alcuni tipi di elementi) e ne definisce un particolare arrangiamento – una configurazione o topologia
 - strutture diverse comunicano informazioni differenti
 - ad es., l'interazione tra due processi che comunicano tra loro (ad es., tramite pipe) potrebbe non essere affatto evidente nella struttura statica del codice
 - ciascuna struttura è affronta aspetti diversi
 - ad es., la struttura statica è importante per la modificabilità del sistema, quella dinamica è importante per le sue prestazioni, e quella di deployment per la sua disponibilità

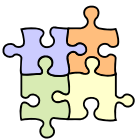


- Esempio

- Un'applicazione web per basi di dati potrebbe essere realizzata scrivendo/utilizzando i seguenti *moduli*

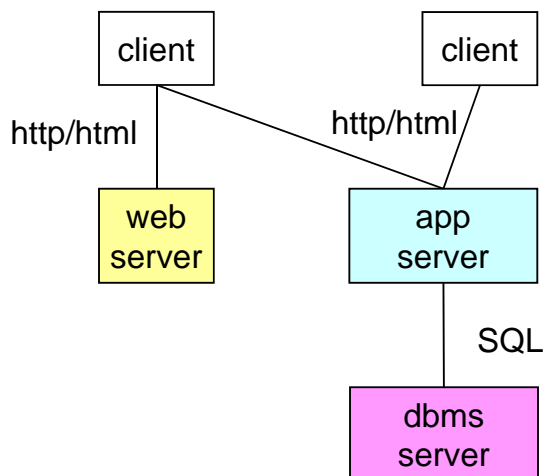


- questa è una struttura statica
- gli elementi sono moduli



Esempio

- L'esecuzione dell'applicazione web mediante diversi *processi*, in esecuzione su più *computer*, che comunicano in vari modi

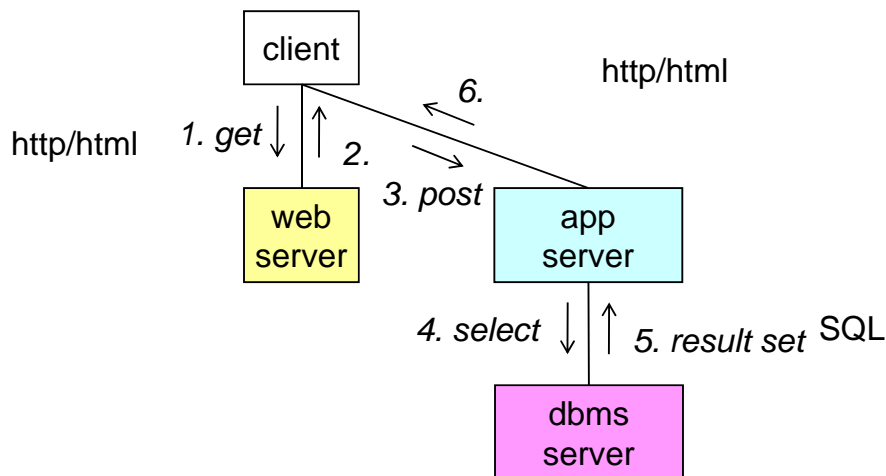


- questa è una struttura dinamica, perché gli elementi sono processi
- tuttavia, non si parla ancora della dinamica delle interazioni
- si noti anche che la struttura di deployment non è specificata

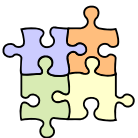


Esempio

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

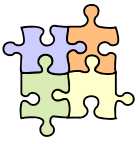


- questa è una struttura dinamica
- gli elementi sono processi
- anche i messaggi scambiati sono elementi significativi
- sarebbe interessante anche una discussione sulle istanze dei moduli



- Relazioni

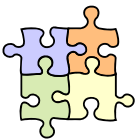
- L'architettura software di un sistema è l'insieme delle strutture del sistema, necessarie per ragionare su di esso, che comprendono elementi software, le *relazioni tra di essi*, e le loro proprietà



Relazioni

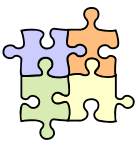
- Abbiamo già illustrato l'importanza delle relazioni tra elementi parlando sia di elementi che di strutture
 - i connettori (sono un tipo particolare di elementi software) hanno lo scopo di descrivere le interazioni (sono un tipo di relazione) tra componenti software
 - una vista/struttura comprende sempre le relazioni tra gli elementi che vi compaiono

- Altre relazioni di interesse nelle architetture
 - relazioni tra strutture, ovvero tra elementi presenti in strutture diverse – o, in generale, tra elementi di natura diversa
 - ad es., un modulo – un processo che è istanza di quel modulo – il nodo su cui è in esecuzione quel processo – il team responsabile dell'implementazione di quel modulo
 - nell'esempio precedente, descritte tramite l'uso di colori



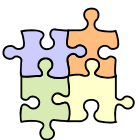
- Proprietà

- 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*



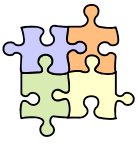
Proprietà

- L'architettura 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)
 - l'architettura non è interessata all'implementazione interna degli elementi
 - il comportamento di ciascun elemento è parte dell'architettura solo nella misura in cui quel comportamento può essere osservato dal punto di vista di un altro elemento
- Le *proprietà* sono le ipotesi che ciascun elemento può fare circa gli altri elementi – in termini di servizi forniti e di qualità dei servizi forniti



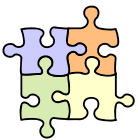
Proprietà

- Due tipologie di **proprietà** di un elemento software
 - il *comportamento* (*funzionalità*), che definisce le interazioni funzionali tra l'elemento software e il suo ambiente
 - ovvero, quali servizi offre un elemento? che cosa gli posso chiedere? come glielo posso chiedere? come mi risponde?
 - queste funzionalità sono di solito descritte a scatola nera, mediante, ad es., l'interfaccia dell'elemento (le sue operazioni pubbliche), un protocollo che descrive l'ordine in cui chiamare le varie operazioni, i contratti per queste operazioni (precondizioni e postcondizioni)
 - le *proprietà di qualità* (*qualità*), ovvero le proprietà non funzionali che sono percepite dagli altri elementi
 - ovvero, come un elemento fa le cose – ad esempio, in termini di prestazioni, sicurezza, affidabilità, ...



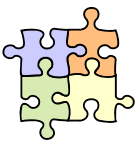
- Ragionare sul sistema

- 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



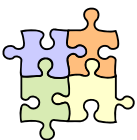
Ragionare sul sistema

- Abbiamo parlato delle proprietà degli elementi interni di un sistema
 - ma quello che interessa sono soprattutto le *proprietà complessive del sistema*
 - la disciplina delle architetture software è in particolare 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



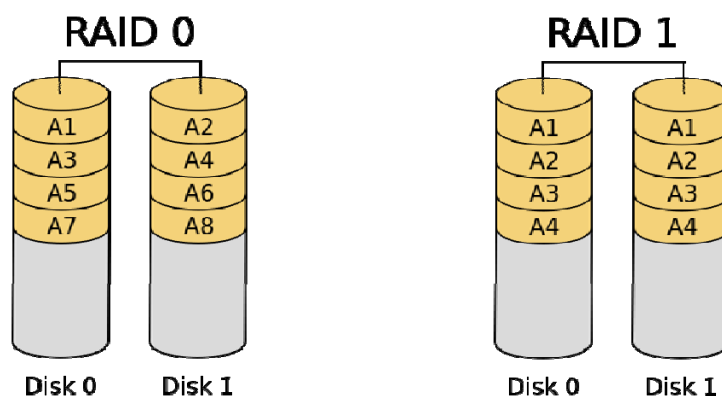
Organizzazione interna e proprietà esterne

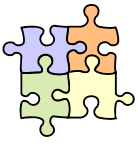
- Relazione tra organizzazione interna e proprietà esterne
 - il **comportamento complessivo** di un sistema è determinato dal comportamento funzionale combinato dei suoi elementi interni
 - anche le **qualità complessive** di un sistema derivano dalle qualità dei suoi elementi interni
 - di solito (se non si progetta opportunamente), una qualità complessiva del sistema è buona come la qualità del suo elemento interno peggiore o più debole
 - è però spesso possibile costruire sistemi di qualità migliore di ciascuno dei suoi elementi interni presi individualmente
- Attenzione, uno stesso insieme di elementi può essere organizzato per sostenere qualità diverse in modi differenti – questo dipende dalla struttura scelta



Organizzazione interna e proprietà esterne

- Ad esempio, esulando dal software, si pensi alle configurazioni RAID per i dischi
 - tutte queste configurazioni sono ottenute dagli stessi elementi – un certo numero di dischi e un controller RAID
 - tuttavia, configurazioni diverse danno luogo a qualità esterne diverse

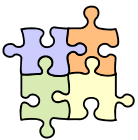




Organizzazione interna e proprietà esterne

□ Due domande

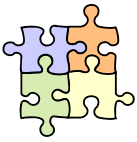
- ma siamo sicuri che c'è una relazione significativa tra organizzazione interna e proprietà esterne?
- come facciamo a comprendere le qualità esterne complessive di un sistema organizzato secondo una certa architettura?



Organizzazione interna e proprietà esterne

□ Alcuni esempi (intuizioni)

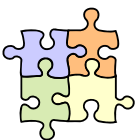
- per ottenere alte prestazioni, bisogna gestire le caratteristiche temporali del comportamento degli elementi, nonché frequenza e volume della comunicazione tra elementi
- se la modificabilità è importante, le responsabilità vanno assegnate agli elementi in modo tale che ciascun cambiamento atteso abbia effetto su uno o pochi elementi
- se il sistema deve essere altamente sicuro, la comunicazione tra gli elementi va gestita e protetta opportunamente – potrebbe essere richiesta l'introduzione di elementi specializzati
- se la scalabilità è importante, l'uso delle risorse deve essere opportunamente localizzato, in modo da poter facilitare l'introduzione di risorse con maggior capacità
- se si vuole riusabilità, l'accoppiamento tra elementi deve essere opportunamente ridotto
- ...



Organizzazione interna e proprietà esterne

Alcuni esempi (intuizioni)

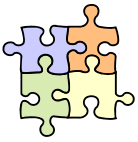
- per ottenere certe proprietà, le caratteristiche tempo, frequenza e volume sono importanti
 - se l'architettura è sbagliata, il sistema non funziona
- In questi esempi, si può notare che la gestione di queste qualità avviene soprattutto a livello architetturale.
- Infatti, in tutti questi esempi, si parla di elementi, proprietà degli elementi, relazioni tra elementi – e delle caratteristiche che essi devono possedere per poter ottenere una certa qualità.
- opportuno l'introduzione di un elemento che deve essere opportunamente ridotto
 - ...



Architetture software

La disciplina delle architetture software

- uno degli obiettivi fondamentali della disciplina delle architetture software è proprio lo studio e la comprensione delle relazioni tra struttura, qualità interne e qualità esterne di un sistema 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
 - ma anche per valutare un'architettura, sempre con riferimento a delle qualità desiderate



- Esempio

- Un **sistema di prenotazioni aeree** deve consentire un certo numero di tipi di transazioni (funzionalità)
 - prenotare un posto, modificare o annullare una prenotazione, ...
- e offrire alcune qualità desiderate
 - tempo medio di risposta ad una transazione (in certe condizioni di carico), throughput massimo, disponibilità, tempo di ripristino in caso di fallimento, ...
- A fronte di questi requisiti (funzionali e di qualità), il sistema potrebbe essere organizzato secondo diverse possibili architetture (architetture candidate)
 - un'architettura candidata per un sistema è un particolare arrangiamento di strutture statiche e dinamiche che ha il potenziale per esibire le proprietà visibili esternamente (funzionali e di qualità) del sistema
 - ad es., architettura client/server a due livelli oppure a tre livelli

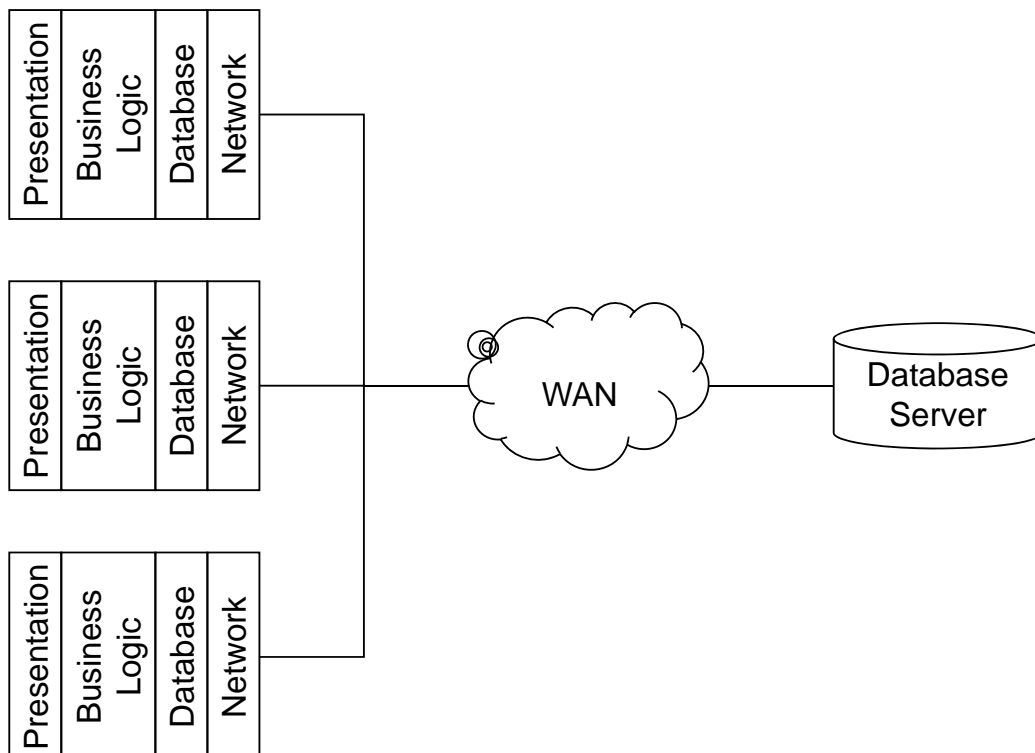
37

Architetture software: Concetti

Luca Cabibbo - ASw



Architetture client/server a due livelli



38

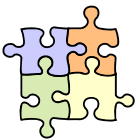
Architetture software: Concetti

Luca Cabibbo - ASw

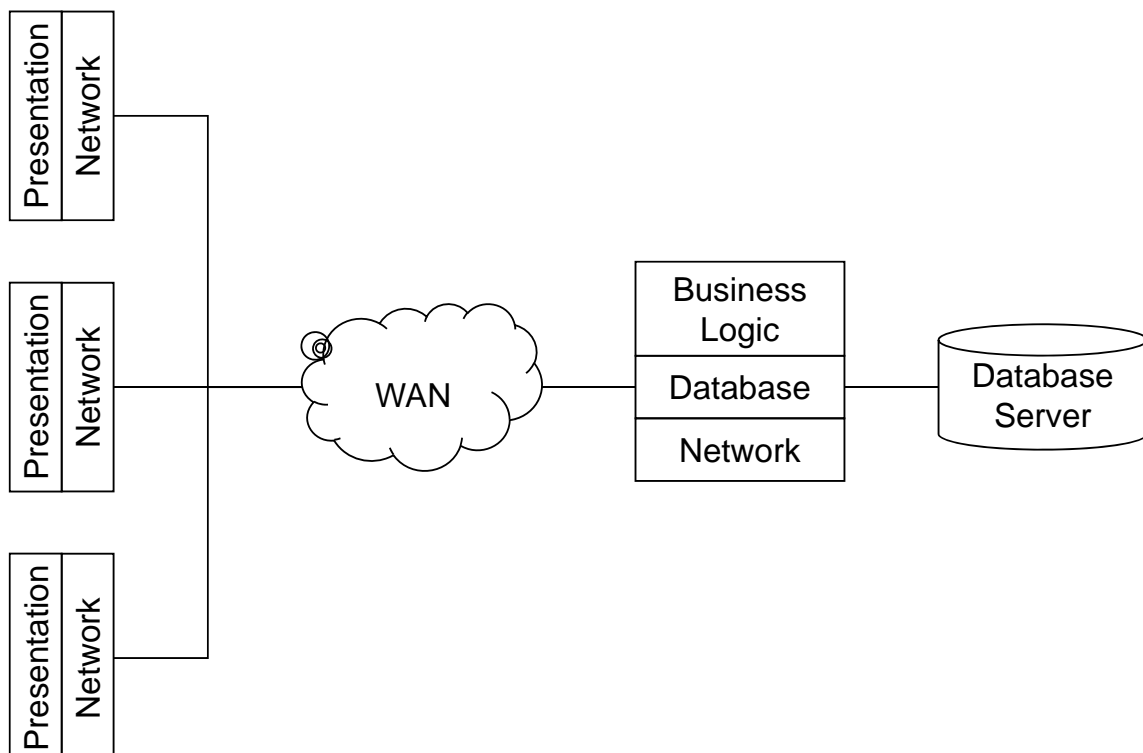


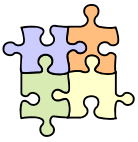
Architettura client/server a due livelli

- *Client/server* è uno *stile architetturale*
- Struttura statica
 - programmi client (thick client) – un'architettura logica a strati, con strati Presentation, Business Logic, Database e Network
 - database server – schema della base di dati
 - connessioni tra di essi
- Struttura dinamica
 - modello request/response – il client (thick client) effettua delle richieste (sulla base di un opportuno formato/protocollo) al database server e riceve delle risposte – tramite la WAN



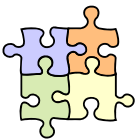
Architetture client/server a tre livelli





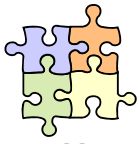
Architettura client/server a tre livelli

- Una variante – architettura *client/server a tre livelli*
- Struttura statica
 - programmi client (thin client) – un'architettura logica a strati, con strati Presentation e Network
 - programma sul server applicativo – con un'architettura a strati, con gli strati Business Logic, Database e Network
 - database server – ad es., schema della base di dati
 - connessioni tra di essi
- Struttura dinamica
 - modello request/response a tre livelli
 - il client (thin client) effettua delle richieste all'application server...
 - l'application server effettua delle richieste al database server...




Confronto

- Entrambe queste architetture candidate *dovrebbero* consentire di soddisfare i requisiti
 - ma come faccio a saperlo? come scelgo tra le due?
attenzione, le due architetture potrebbero soddisfare i requisiti diversi in misura diversa
 - l'architettura C/S a due livelli è più semplice e veloce da realizzare e gestire, ...
 - l'architettura C/S a tre livelli ha minori esigenze sull'hardware dei client, una migliore scalabilità, migliori opzioni per la gestione della sicurezza, ...
- L'architetto deve essere in grado di
 - identificare le architetture candidate
 - comprendere le loro qualità
 - scegliere l'architettura "migliore" per il particolare sistema in discussione

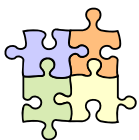


- Ulteriori osservazioni



 [ISO-42010]

- **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
- Questa definizione enfatizza l'importanza per le architetture
 - dell'ambiente (contesto) del sistema
 - comprende sia l'ambiente di esecuzione – ma anche quello di sviluppo, operativo, politico, sociale, ...
 - dei principi che guidano la progettazione e l'evoluzione
 - ad es., lo stile architeturale



Ulteriori osservazioni

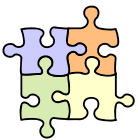


- Non tutte le architetture sono buone architetture
 - la definizione di architettura software non distingue tra architetture buone e non buone
 - è importante saper valutare le architetture
- Ogni sistema software ha un'architettura
 - anche se questa non è stata documentata
 - inoltre, in alcuni casi, 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 talvolta ovviare mediante un'attività di ricostruzione dell'architettura



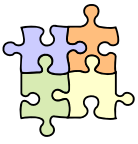
* Ulteriori concetti

- Ulteriori concetti importanti sulle architetture software sono relativi al contesto in cui le architetture software vengono definite
 - processo di definizione dell'architettura
 - parti interessate e interessi
 - qualità e compromessi
 - l'architetto e il suo ruolo



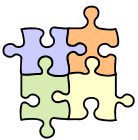
- Processo di definizione dell'architettura

- 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



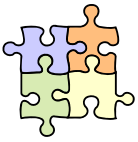
Definizione dell'architettura

- In prima approssimazione, la definizione dell'architettura
 - ha in ingresso gli interessi delle parti interessate
 - produce come risultato una descrizione dell'architettura
 - la descrizione delle strutture dell'architettura (viste) – insieme alle relative giustificazioni logiche
- In realtà, questo processo
 - è a cavallo tra la comprensione degli interessi e la progettazione – che si influenzano reciprocamente
 - adotta un processo di natura iterativa
 - basata su una progettazione iniziale – per sostenere gli interessi più importanti – e raffinamenti successivi – per sostenere gli ulteriori interessi
 - richiede di saper valutare l'architettura
 - ad es., per capire se sono necessari ulteriori raffinamenti



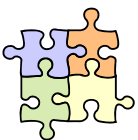
- Parti interessate e interessi

- La realizzazione di un sistema software è sempre soggetta agli interessi di un certo numero di parti interessate
 - una *parte interessata* è un individuo, un gruppo di persone o un'organizzazione che ha interessi nel sistema
 - ad es., chi userà il sistema, chi lo deve pagare, chi lo deve amministrare, chi lo deve sviluppare e mantenere, la legislazione vigente, ...
 - gli *interessi* sono relativi agli obiettivi che le varie parti interessate desiderano e vogliono perseguire
 - ad es., funzionalità e qualità del sistema, tempi di realizzazione, sull'organizzazione interna del sistema e sul modo di lavorare del team di sviluppo, ...



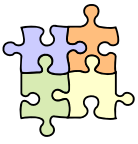
Parti interessate

- Una **parte interessata (stakeholder)** [ISO-42010] in un sistema è una persona, un gruppo o un'organizzazione che ha un interesse circa la realizzazione del sistema
 - cliente/acquirente – paga il sistema
 - valutatori/periti – ne valutano la conformità alle leggi o standard
 - supporto – forniscono supporto agli utenti, quando il sistema è in uso
 - amministratori di sistema – amministrano il sistema
 - utenti – usano il sistema – possono definire i requisiti funzionali del sistema
 - comunicatori – spiegano il sistema alle altre parti interessate
 - sviluppatori – costruiscono e rilasciano il sistema
 - tester – verificano il sistema
 - manutentori – gestiscono l'evoluzione del sistema dopo che è stato rilasciato
 - fornitori – costruiscono o forniscono hardware o software



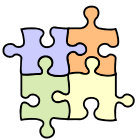
Importanza delle parti interessate

- Le parti interessate sono importanti perché
 - sono le persone che possono dire quello che serve
 - sono le persone che possono dire se quello che è stato costruito/progettato è soddisfacente (o meno)
 - il successo è definito dall'osservatore, non dall'architetto
 - l'architetto deve evitare giudizi sul valore del sistema
- Dunque, le parti interessate guidano (direttamente o indirettamente) l'intera forma e organizzazione dell'architettura
 - l'architettura (il sistema) viene creata solo per soddisfare i bisogni delle parti interessate
 - senza parti interessate, non servirebbe sviluppare il sistema
 - sotto la guida dell'architetto
 - le parti interessate esprimono desideri circa gli interessi da perseguire, l'architetto decide sulla fattibilità



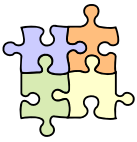
Interessi architetturali

- Un **interesse (concern)** [SSA] circa un'architettura
 - è un requisito, un obiettivo, un intento o un'aspirazione che una parte interessata ha per l'architettura
- Si tratta di una definizione molto generale
 - gli interessi comprendono i requisiti funzionali e non funzionali (di qualità)



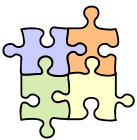
Che cos'è architetturalmente significativo?

- Alcuni interessi/requisiti per un sistema sono rilevanti per l'architettura – altri invece sono meno rilevanti
 - l'architettura deve focalizzare l'attenzione su ciò che è architetturalmente significativo
 - lasciando ciò che non è architetturalmente significativo alla lavorazione dei requisiti – oppure alla progettazione di dettaglio
- 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 sulla struttura del sistema o su una sua qualità importante – come prestazioni, scalabilità, sicurezza, affidabilità o modificabilità



- Qualità e compromessi

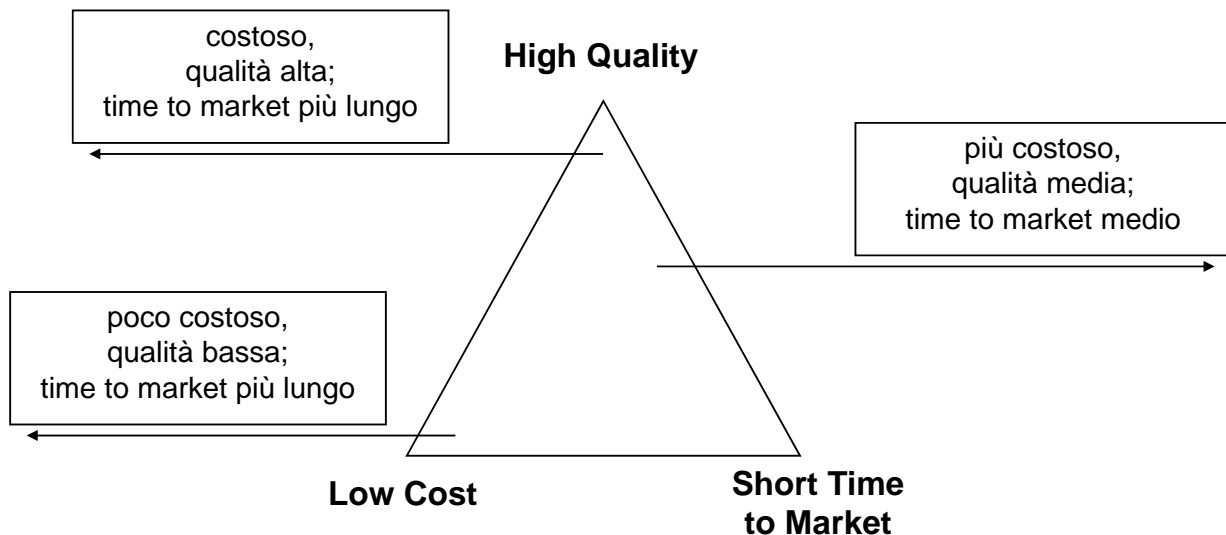
- Una fonte di complessità nella progettazione delle architetture è legata a questa osservazione
 - in linea di principio, dato un insieme F qualunque di funzionalità, è sempre possibile costruire un sistema che offre tutte le funzionalità in F
 - tuttavia, dato F e un insieme Q qualunque di qualità, non sempre è possibile costruire un sistema che offre le funzionalità in F ed esibisce tutte le qualità in Q
- In effetti, molte decisioni architettoniche richiedono comunemente di identificare degli opportuni *compromessi* (*tradeoff*) nella gestione delle qualità
 - è anche comune che i requisiti di qualità vengano rinegoziati durante la definizione dell'architettura – con la possibilità che alcuni requisiti vengano opportunamente limitati oppure ridefiniti



Qualità contrastanti e compromessi

- Il triangolo della qualità mostra tre interessi contrastanti
 - qualità, costo, time to market

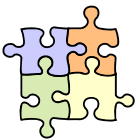
Faster, better, cheaper – choose two
(you can't have all three at once)



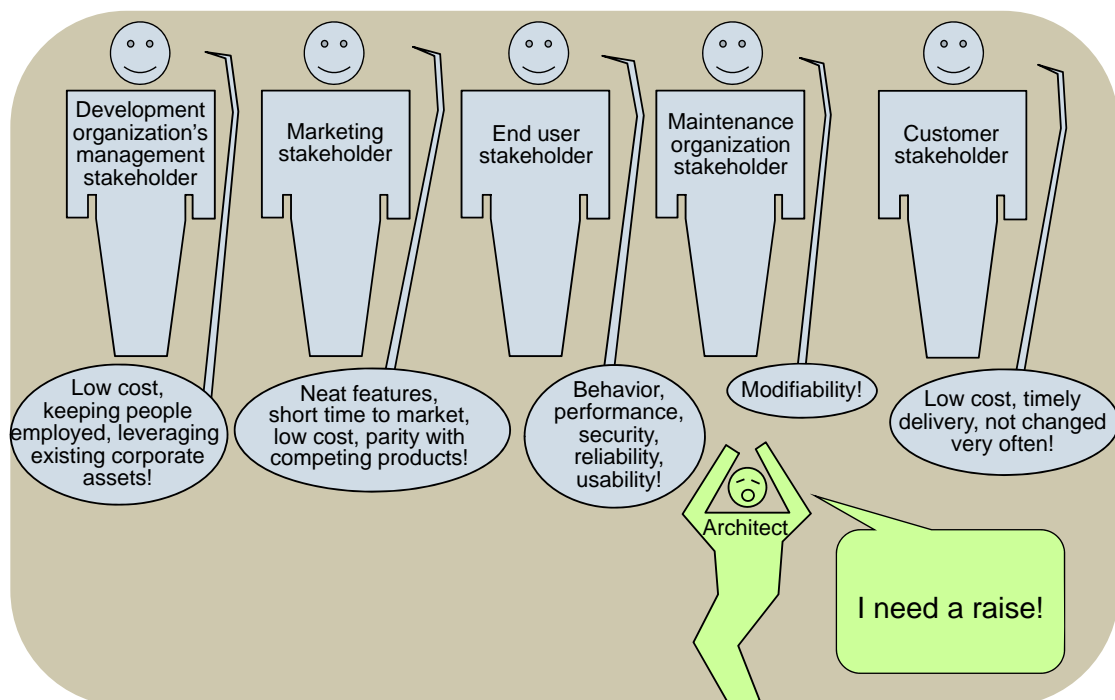


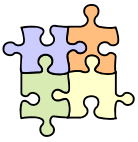
- L'architetto e il suo ruolo

- L'architetto – che è una delle parti interessate – deve saper
 - identificare le parti interessate
 - catturarne gli interessi
 - operare ove necessario per riconciliare interessi contrastanti – mediante opportuni compromessi
 - prendere decisioni
 - comunicare le proprie decisioni alle parti interessate
- Un buon architetto è uno che sa cogliere con successo gli obiettivi, gli scopi e i bisogni delle sue parti interessate



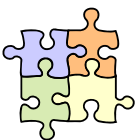
L'architetto e il suo ruolo





* Discussione

- La seguente definizione (adattata da una definizione di McGovern) consente di riassumere 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



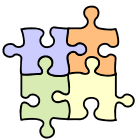
Discussione

- Vantaggi nel progettare e documentare 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 su e gestire i cambiamenti
 - predire e mitigare rischi
 - ...



Discussione

- Alcune attività legate alle architetture software
 - comprensione degli interessi e dei requisiti
 - creazione o scelta dell'architettura
 - descrizione dell'architettura
 - analisi e valutazione dell'architettura
 - comunicazione dell'architettura
 - implementazione del sistema – guidati dall'architettura
 - ricostruzione dell'architettura



Relazioni tra concetti fondamentali

