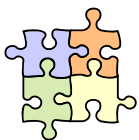


# Architetture Software

## Introduzione alle architetture software

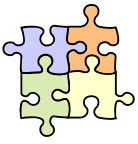
Dispensa ASW 110

ottobre 2011



### - Fonti

- [SAP] Chapter 2, What Is Software Architecture?
- [SSA] Chapter 1, Introduction
- [Maier&Rechtin] The Art of Systems Architecting, Third Edition, CRC Press, 2009



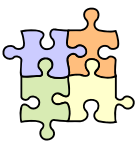
## - Obiettivi e argomenti

### □ Obiettivi

- motivare l'approccio e l'importanza delle architetture software

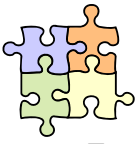
### □ Argomenti

- che cos'è l'architettura
- introduzione alle architetture software
- alcuni esempi
- letture consigliate



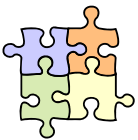
## \* Che cosa è l'architettura

- **Architecture** [Wikipedia, ottobre 2011] is both the process and product of planning, designing and construction. Architectural works, in the material form of buildings, are often perceived as cultural and political symbols and as works of art. Historical civilizations are often identified with their surviving architectural achievements.
- "Architecture" can mean:
  - The art and science of design and erecting buildings and other physical structures.
  - A general term to describe buildings and other infrastructures.
  - A style and method of design and construction of buildings and other physical structures.
  - The practice of an architect, ...
  - Design activity, from the macro-level (urban design, landscape architecture) to the micro-level (construction details and furniture).
  - The term "architecture" has been adopted to describe the activity of designing any kind of system, and is commonly used in describing information technology.
- In relation to buildings, architecture has to do with the planning, designing and constructing form, space and ambience that reflect functional, technical, social, environmental, and aesthetic considerations. It requires the creative manipulation and coordination of material, technology, light and shadow. Architecture also encompasses the pragmatic aspects of realizing buildings and structures, including scheduling, cost estimating and construction administration. As documentation produced by architects, typically drawings, plans and technical specifications, architecture defines the structure and/or behavior of a building or any other kind of system that is to be or has been constructed.



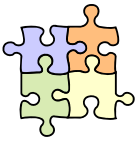
## Che cosa è l'ingegneria

- **Engineering** [Wikipedia, ottobre 2011] is the discipline, art, skill and profession of acquiring and applying scientific, mathematical, economic, social, and practical knowledge, in order to design and build structures, machines, devices, systems, materials and processes that safely realize improvements to the lives of people.
- 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 and safety to life and property.
- One who practices engineering is called an engineer, and those licensed to do so may have more formal designations such as Professional Engineer, Chartered Engineer, Incorporated Engineer, Ingenieur or European Engineer. The broad discipline of engineering encompasses a range of more specialized subdisciplines, each with a more specific emphasis on certain fields of application and particular areas of technology.



## Progettazione di sistemi complessi

- Sia l'architettura che l'ingegneria si occupano della progettazione e costruzione di sistemi complessi
  - come costruzioni, strutture, sistemi, ...
  - che devono soddisfare gli interessi di numerose parti interessate – considerazioni funzionali, sociali e ambientali, funzionamento in condizioni specifiche, sicurezza delle persone, ...
  - con riferimento anche ad aspetti pragmatici – tempi, costi, ...
- In pratica, l'architettura e l'ingegneria rappresentano due ruoli estremi – in uno spettro continuo – della pratica della progettazione di sistemi complessi
  - la progettazione di sistemi complessi richiede spesso un approccio sia ingegneristico che architeturale
  - in parte scienza, in parte arte



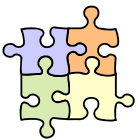
## Ingegneria e architettura a confronto

### □ *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
- interessata a costi quantificabili
- ambisce all'ottimizzazione tecnica
- 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
- interessata al valore qualitativo
- ambisce alla soddisfazione del cliente
- più arte



## Sistemi e complessità

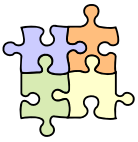
### □ *Sistema*

- 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*

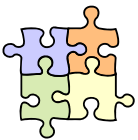
- composto di parti interconnesse o intrecciate

- Si noti come entrambe le definizioni parlano *parti* o *elementi*, nonché di *interrelazioni* e *interconnessioni* tra le parti



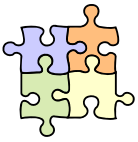
## Affrontare la complessità

- In che modo è possibile affrontare/progettare problemi/sistemi caratterizzati da un alto livello di complessità?
  - partizionare (ovvero, decomporre) il sistema/problema via via in unità più piccole e più semplici
- Tuttavia, affinché la complessità venga ridotta, non è sufficiente che le varie parti siano “semplici”
  - è necessario che anche le interconnessioni siano “semplici”
  - ovvero, vanno utilizzati criteri di partizionamento/decomposizione che, per il problema in esame, siano in grado di ridurre opportunamente la complessità delle interconnessioni tra le parti



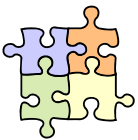
## Progettazione di sistemi complessi

- Metodologie/approcci ingegneristico/architetturali
  - normativo (scienza)
    - basato su soluzioni pre-esistenti – “deve essere così”
  - razionale (scienza)
    - basato su principi – consente (in parte) di produrre soluzioni “nuove” – per affrontare cambiamenti nelle necessità, nelle preferenze o nelle circostanze
  - partecipativo (arte)
    - riconosce la complessità dovuta alla presenza di una molteplicità di parti interessate – l’obiettivo è il consenso
  - euristico (arte)
    - basato su euristiche che codificano del “buon senso comune” motivato da esperienza collettiva



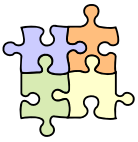
## \* Introduzione alle architetture software

- I grandi sistemi software di oggi sono tra le strutture costruite dagli uomini più complesse – e la loro complessità è via via crescente
  - contengono milioni di linee di codice, migliaia di tabelle nelle basi di dati, e sono eseguiti da dozzine di calcolatori
  - richiedono il coinvolgimento e la negoziazione tra numerose parti interessate – committente, utenti, sviluppatori, ... – di cui devono soddisfare gli interessi – sia funzionali che di qualità
  - i team di sviluppo del software sono grandi e spesso distribuiti – ed operano su periodi di tempo estesi
  - ciò presenta a questi team delle sfide formidabili – e se queste sfide non vengono affrontate presto ed in modo opportuno, i sistemi realizzati si rivelano dei fallimenti
    - ad es., sono consegnati in ritardo, costano più del previsto, ed hanno un livello di qualità inaccettabilmente povero
- Per affrontare queste sfide, molti riconoscono oggi l'importanza di un approccio guidato dall'architettura software



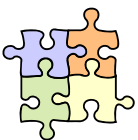
## Funzionalità e qualità del software

- Gli interessi (ovvero, le caratteristiche desiderate) per i sistemi software riguardano sia le **funzionalità** che altre importanti proprietà che ne riflettono la **qualità** – ad esempio
  - **affidabilità** – il software deve effettivamente fornire le funzionalità richieste
  - **disponibilità** – il software deve essere funzionante in modo continuato nel tempo
  - **prestazioni, scalabilità, sicurezza, usabilità, ...**
  - **verificabilità, manutenibilità, ...**
  - **eterogeneità** – facendo interagire sistemi eterogenei (per piattaforma, rappresentazione dei dati, ...)
  - **agilità di business** – poter creare o modificare dinamicamente processi di business – per poter agire in modo competitivo nel proprio settore di business
  - **economicità, risparmio energetico, ...**



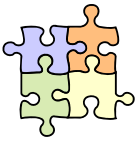
## Importanza delle qualità del software

- In generale, un fattore critico per il successo di un'organizzazione – o di una sua linea di attività – è
  - la costruzione di sistemi software (o comunque software-intensive) in grado di soddisfare i requisiti – non solo funzionali, ma anche di qualità – non solo correnti, ma anche futuri – di quell'organizzazione
- I requisiti di qualità rivestono un ruolo significativo nel successo di un sistema software
  - il mancato raggiungimento di alcuni livelli richiesti minimi di qualità può rendere il sistema inaccettabile e dunque inutilizzabile – anche se le funzionalità sono realizzate in modo impeccabile



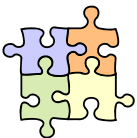
## Architetture software

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



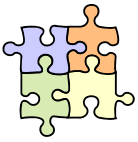
## Che cosa sono le architetture software?

- Ecco alcune possibili definizioni di architettura software
  - the *software architecture* of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [SAP]
  - *architecture* is the linchpin for the highly complex, massively large-scale, and highly interoperable systems that we need now and in the future [Rolf Siegers]
- Qual è la relazione tra queste due definizioni – che sono apparentemente molto lontane tra di loro?
  - **l'architettura software si occupa della struttura interna di un sistema software – perché questa struttura interna ha una influenza significativa sulle qualità del sistema**



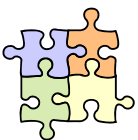
## - L'approccio delle architetture software

- L'approccio delle architetture software
  - riconosce l'importanza di tutte le parti interessate e dei loro interessi – soprattutto delle qualità del sistema
    - come prestazioni, sicurezza, verificabilità, modificabilità, ...
  - l'architettura viene definita a cavallo tra requisiti (che cosa deve fare il sistema?) e progettazione (come è fatto il sistema?)
    - da una parte, guida la negoziazione e la scelta dei requisiti di qualità
    - dall'altra, guida la progettazione, l'implementazione e l'evoluzione del sistema
  - si occupa della decomposizione del sistema in elementi e delle loro inter-relazioni, per fornire al sistema le qualità richieste
    - sulla base di soluzioni pre-esistenti, principi ed euristiche
  - come disciplina, si occupa della comprensione delle relazioni tra la struttura interna del sistema e le sue qualità esterne



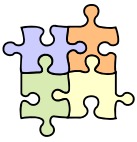
## L'approccio delle architetture software

- In pratica, la definizione di un'architettura software può ruotare attorno a
  - parti interessate e interessi
    - chi ha interessi nel sistema – tra cui le persone per cui il sistema viene costruito, e le persone che lo costruiscono
  - l'applicazione di punti di vista e viste
    - un approccio per la strutturazione dell'architettura del sistema, basato sul principio della separazione degli interessi, con riferimento agli interessi principali del sistema
  - l'applicazione di stili architeturali
    - esperienze significative nella realizzazione di architetture
  - l'applicazione di tattiche e prospettive
    - un approccio per valutazione e la (ri)strutturazione dell'architettura del sistema, con riferimento ad ulteriori interessi del sistema



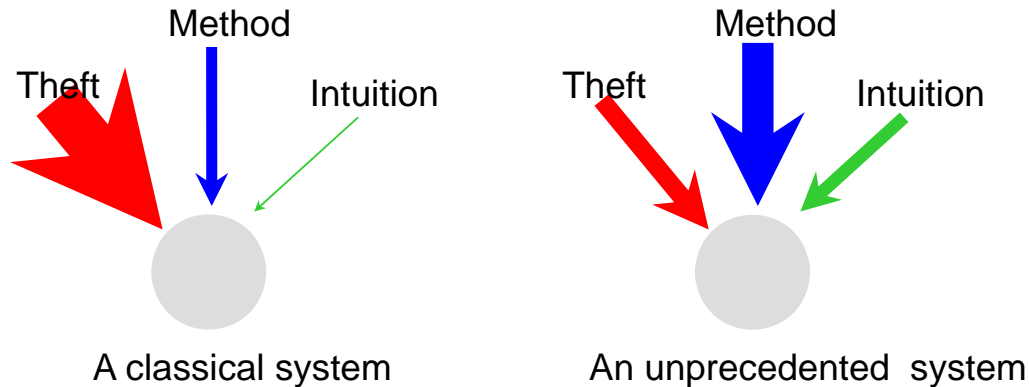
## Parti interessate ed interessi

- I requisiti (che cosa deve fare il sistema?) di un sistema derivano dalle varie *parti interessate* al sistema
  - utenti – chi userà il sistema
  - amministratori – chi gestisce il sistema
  - acquirente – chi paga per il sistema
  - sviluppatori – creano, verificano, mantengono il sistema
  - ...
- Ciascuna parte interessata ha degli *interessi* – funzionali e/o non funzionali (di qualità)
  - l'architettura deve tenere in considerazione e sostenere gli interessi – spesso contrastanti – delle parti interessate
    - ad es., prestazioni, sicurezza e modificabilità
  - l'architetto guida la definizione e la risoluzione dei requisiti più significativi

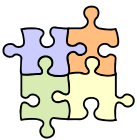


## Le sorgenti delle architetture software

- Da: Mommy, Where Do Software Architectures Come From? [Kruchten, 1995]

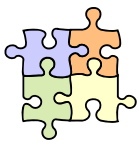


- theft – elementi derivati da un sistema precedente dello stesso tipo
  - method – un approccio sistematico per derivare l'architettura dai requisiti
  - intuition – riflette l'esperienza dell'architetto software
- Come in ogni attività di progettazione, è fondamentale poter sfruttare l'esperienza passata per produrre progetti migliori



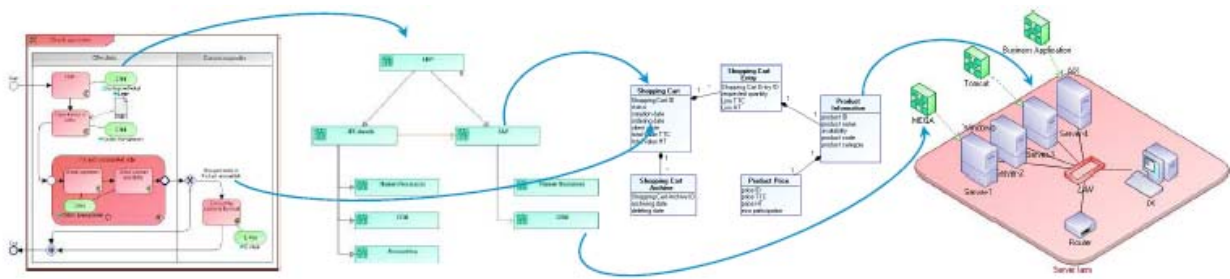
## Viste e punti di vista

- L'architettura software di un sistema è interessata alla decomposizione del sistema in parti e alle relazioni tra queste parti
  - un singolo criterio di partizionamento/decomposizione non è di solito sufficiente per dominare la complessità di un sistema
    - utile decomporre e descrivere il sistema secondo un insieme di *viste architettonali* – indipendenti ma correlate
  - ciascuna vista architettonale definisce un modello (una descrizione parziale) del sistema – che si occupa di un insieme coeso di interessi
    - le viste sono utili per comprendere, progettare, validare ed attuare l'architettura – nonché per descrivere e comunicare l'architettura alle varie parti interessate
  - la selezione e realizzazione delle viste può essere guidata da opportuni *punti di vista* – ad es., funzionale, delle informazioni, della concorrenza, di deployment,...

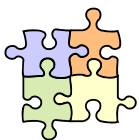


## Viste e punti di vista

- Un esempio di descrizione architeturale (parziale) – basata su più viste

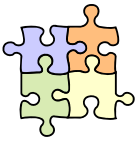


- qui sono utilizzate quattro viste: dei processi di business, funzionale, delle informazioni, di deployment
- da notare, all'interno di ogni vista, l'uso di elementi e di relazioni tra elementi
- da notare anche l'importanza delle relazioni tra elementi di viste diverse



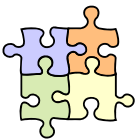
## Stili architeturali

- I pattern sono un modo per condividere esperienze progettuali
  - in generale, un **pattern software** ha lo scopo di condividere una soluzione provata ed ampiamente applicabile ad un particolare problema di progettazione, descritta in una forma standard che possa essere facilmente riusata
  - un **pattern (o stile) architeturale** [POSA] codifica un'esperienza significativa nella realizzazione di un'architettura software
- Uno stile architeturale
  - affronta un problema – un tipo di sistema, con una certa combinazione di requisiti di qualità da raggiungere
  - propone una soluzione – in termini di elementi, relazioni tra elementi e criteri di decomposizione – spesso nell'ambito di una singola vista
  - nonché una discussione della soluzione – in termini delle diverse qualità su cui la soluzione ha impatto



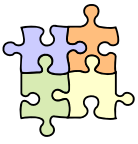
## Tattiche e prospettive

- Gli stili sono di solito applicati per identificare una decomposizione iniziale del sistema che soddisfa alcune qualità fondamentali
  - il progetto iniziale del sistema deve solitamente evolvere, per far sì che il sistema esibisca tutte le qualità richieste (o scelte)
  - la progettazione per le qualità richiede spesso considerazioni specializzate e che riguardano più punti di vista
- Le tattiche e le prospettive architettoniche sono delle ulteriori guide per la progettazione per le qualità di un sistema
  - una **tattica architettonica** [SAP] è una decisione di progetto che influenza il controllo di un attributo di qualità
  - una **prospettiva architettonica** [SSA] è una collezione di attività, tattiche e linee guida per far sì che un sistema esibisca un insieme di proprietà di qualità correlate
  - si applicano per verificare il livello di qualità raggiunto, nonché per guidare il cambiamento delle viste (quando richiesto)



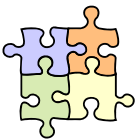
## - Software architecture = ?

- Qual è la traduzione corretta dall'inglese della disciplina **software architecture**?
  - **architetture software** o **architettura del software**?
- Secondo Google
  - “architetture software” – circa 55.000 hit
  - “architettura del software” – circa 24.000 hit
- Dunque questo corso si chiama “architetture software”



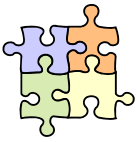
## - Architetonico o architettuale?

- Qual è l'aggettivo corretto associato al nome *architettura*?
  - *architetonico*, *architettuale*, oppure entrambi?
- De Mauro
  - *architetonico* – relativo all'architettura
  - *architettuale* – *non ho trovato occorrenze*
- Treccani
  - *architetonico* – dell'architettura, che concerne l'architettura
  - *architettuale* – sinonimo poco comune di *architetonico*
- In questo corso, userò “architettuale” come aggettivo per “architettura”



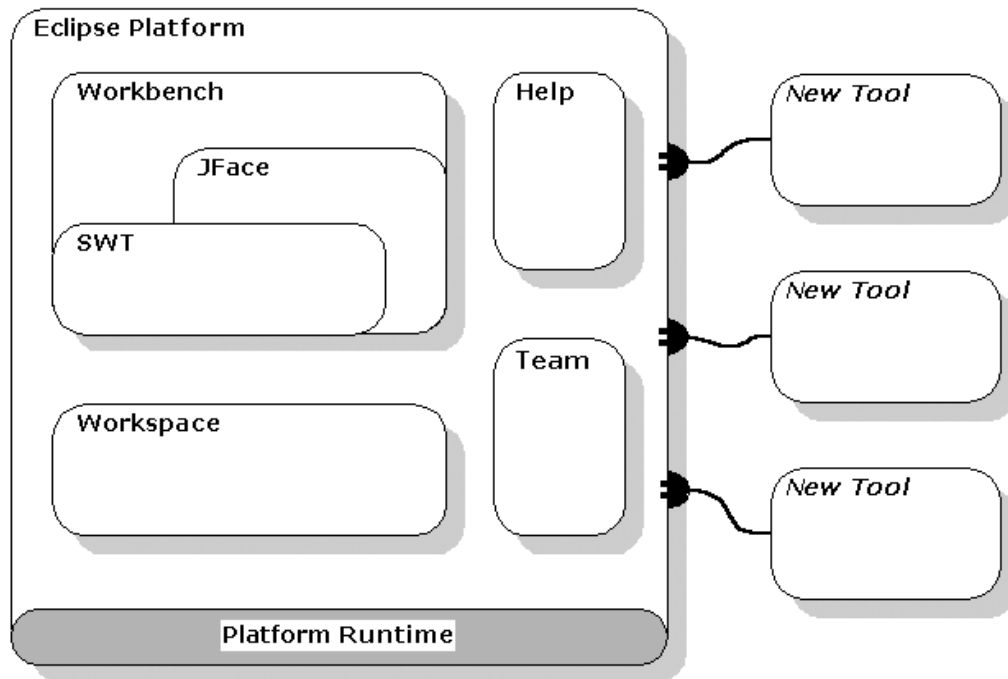
## \* Alcuni esempi

- Seguono alcuni esempi di architetture software
  - l'enfasi è sulle qualità, gli elementi architettureali e le loro relazioni
  - alcuni di questi casi saranno analizzati meglio nel seguito del corso



## - Esempio - Piattaforma Eclipse

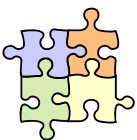
<http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.pdf>



27

Introduzione alle architetture software

Luca Cabibbo - ASw



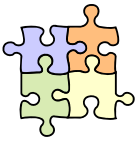
## Piattaforma Eclipse - obiettivi

- The Eclipse Platform is designed and built to meet the following requirements:
  - Support the construction of a variety of tools for application development.
  - Support an unrestricted set of tool providers, including independent software vendors (ISVs).
  - Support tools to manipulate arbitrary content types (e.g., HTML, Java, C, JSP, EJB, XML, and GIF).
  - Facilitate seamless integration of tools within and across different content types and tool providers.
  - Support both GUI and non-GUI-based application development environments.
  - Run on a wide range of operating systems, including Windows®, Linux™, Mac OS X, Solaris, AIX, and HP-UX.
  - Capitalize on the popularity of the Java programming language for writing tools.
- The Eclipse Platform's principal role is to provide tool providers with mechanisms to use, and rules to follow, that lead to seamlessly-integrated tools. These mechanisms are exposed via well-defined API interfaces, classes, and methods. The Platform also provides useful building blocks and frameworks that facilitate developing new tools.

28

Introduzione alle architetture software

Luca Cabibbo - ASw

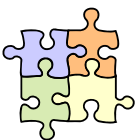


## Piattaforma Eclipse - obiettivi

- The Eclipse Platform is designed and built to meet the following requirements:
  - Support the construction of a variety of tools for application development.
  - Support an unrestricted set of tool providers, including independent software vendors (ISVs).
  - Support tools for developing applications in Java, C, JSP, EJB, XML, etc.
  - Facilitate the development of content types.
  - Support development on various platforms.
  - Run on various operating systems, including Linux™, Mac OS X, and Windows.
  - Capitalize on the Eclipse Platform as a language for writing tools.
- The Eclipse Platform's principal role is to provide tool providers with mechanisms to use, and rules to follow, that lead to seamlessly-integrated tools. These mechanisms are exposed via well-defined API interfaces, classes, and methods. The Platform also provides useful building blocks and frameworks that facilitate developing new tools.

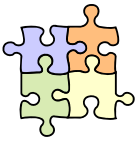
è la descrizione degli obiettivi della piattaforma Eclipse

si noti che si tratta principalmente di requisiti di qualità



## Piattaforma Eclipse - architettura a plug-in

- A plug-in is the smallest unit of Eclipse Platform function that can be developed and delivered separately. Usually a small tool is written as a single plug-in, whereas a complex tool has its functionality split across several plug-ins. Except for a small kernel known as the Platform Runtime, all of the Eclipse Platform's functionality is located in plug-ins.
- Plug-ins are coded in Java. A typical plug-in consists of Java code in a Java Archive (JAR) library, some read-only files, and other resources such as images, web templates, message catalogs, native code libraries, etc. ...
- Each plug-in has a plug-in manifest declaring its interconnections to other plug-ins. The interconnection model is simple: ...
- On start-up, the Platform Runtime discovers the set of available plug-ins, reads their manifests, and builds an in-memory plug-in registry. ...
- A plug-in is activated when its code actually needs to be run. Once activated, ...
- By determining the set of available plug-ins up front, and by supporting a significant exchange of information between plug-ins without having to activate any of them, the Platform can provide each plug-in with a rich source of pertinent information about the context in which it is operating. ...



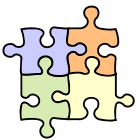
## Piattaforma Eclipse - architettura a plug-in

- A plug-in is the smallest unit of Eclipse Platform functionality that is developed and delivered separately from the Eclipse Platform. A plug-in, except for a small number of special cases, is a self-contained unit of software that provides a specific function.
- Plug-in Architecture (PIA) describes the characteristics of a plug-in, the Platform Runtime (‘‘un piccolo kernel’’), and other elements called plug-in.
- PIA describes the external characteristics that a plug-in must provide, as well as the communication modalities between each plug-in and the Platform Runtime.
- PIA describes the achievement of quality objectives based on this organization.

questo è la descrizione di un'architettura in cui c'è un elemento software chiamato Platform Runtime (‘‘un piccolo kernel’’), nonché altri elementi chiamati plug-in

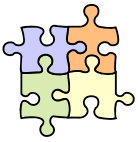
descrive le caratteristiche esterne che deve fornire un plug-in, nonché le modalità di comunicazione tra ciascun plug-in e Platform Runtime

descrive il raggiungimento degli obiettivi di qualità sulla base di questa organizzazione



## - Esempio - Java Enterprise Edition

- Da Java EE Tutorial – [java.sun.com/javasee/6/docs/tutorial/doc/](http://java.sun.com/javasee/6/docs/tutorial/doc/)
  - Developers today increasingly recognize the need for distributed, transactional, and portable applications that leverage the speed, security, and reliability of server-side technology. In the world of information technology, enterprise applications must be designed, built, and produced for less money, with greater speed, and with fewer resources.
  - With the Java™ Platform, Enterprise Edition (Java EE), development of Java enterprise applications has never been easier or faster. The aim of the Java EE platform is to provide developers a powerful set of APIs while reducing development time, reducing application complexity, and improving application performance.
  - ...

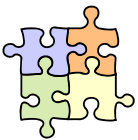


# - Esempio - Java Enterprise Edition

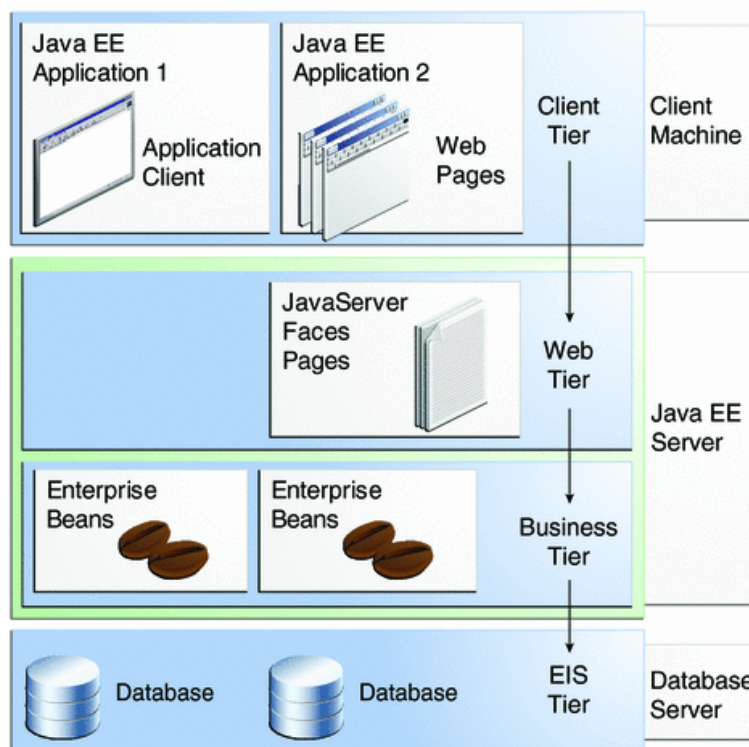
Da Java EE Tutorial – [java.sun.com/javaee/6/docs/tutorial/doc/](http://java.sun.com/javaee/6/docs/tutorial/doc/)

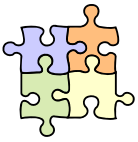
- Developers today increasingly recognize the need for distributed, transactional, and portable applications that leverage the speed, security, and portability of server-side technologies. Java Enterprise Edition provides a platform for building such applications.
- With Java Enterprise Edition, it has never been easier or more powerful to build applications. The goal is to provide developers a powerful set of tools while reducing development time, reducing application complexity, and improving application performance.
- ....

notare l'enfasi della piattaforma  
Java Enterprise Edition su  
requisiti di qualità



# Modello applicativo per Java EE





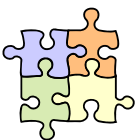
## Modello applicativo per Java EE

è una descrizione dell'architettura delle applicazioni che possono eseguite in un application server Java EE

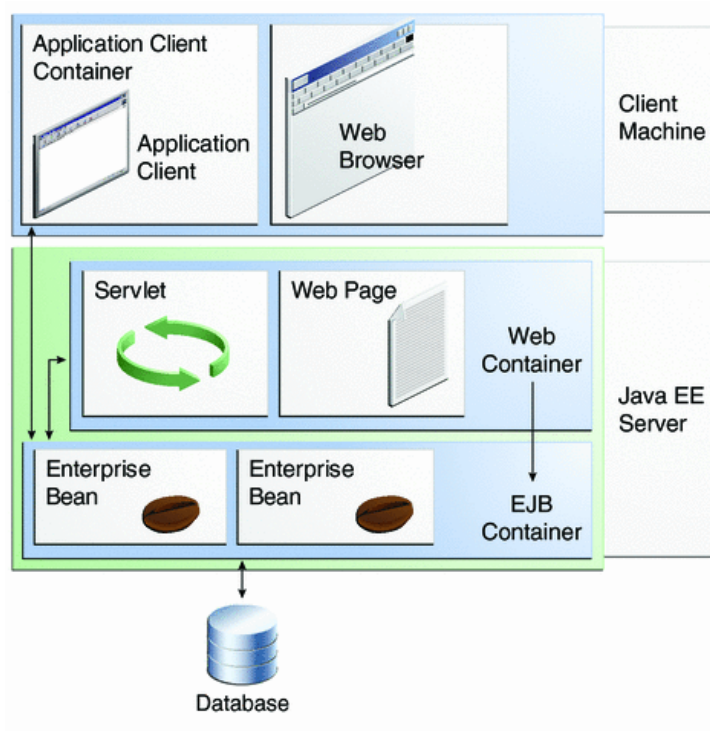
come effettuare la decomposizione architetturale di un'applicazione se la tecnologia target è Java EE?

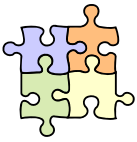
quali qualità possiedono le applicazioni per la piattaforma Java EE?

come sono assicurate queste qualità?

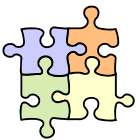
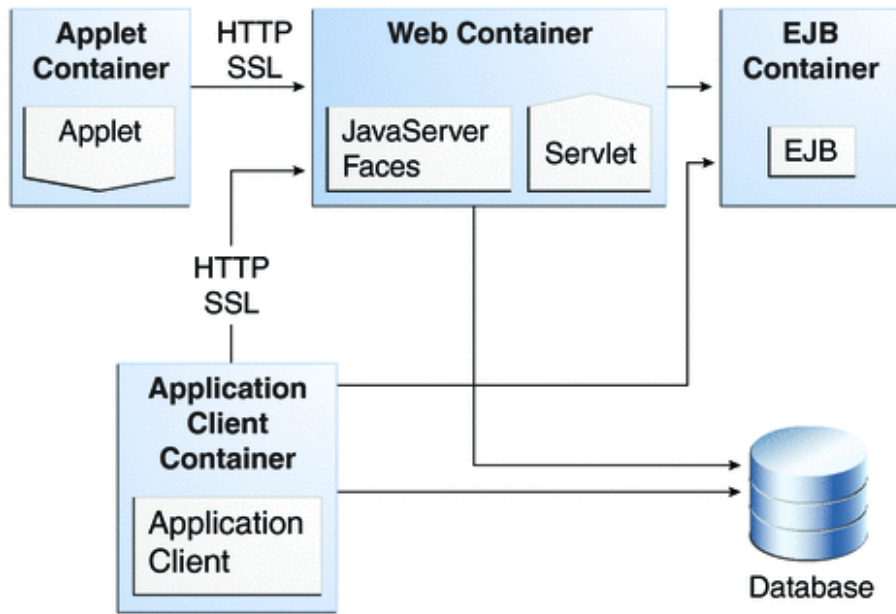


## Container Java EE





# Relazioni tra container Java EE



# Architettura delle API di Java EE

<b>Application Client Container</b>	Java Persistence	<b>Java SE</b>
	Management	
	WS Metadata	
	Web Services	
	Application Client	
	JSR 299	
	JMS	
	JAXR	
JAX-WS	SAAJ	
JAX-RPC	SAAJ	

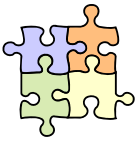
  New in Java EE 6

<b>Web Container</b>	JSR 330	<b>Java SE</b>
	Interceptors	
	Managed Beans	
	JSR 299	
	Bean Validation	
	EJB Lite	
	EL	
	Servlet	
	JavaMail	
	JSP	
	JavaServer Faces	
	Connectors	
	Java Persistence	
	JMS	
	Management	
	WS Metadata	
	Web Services	
JACC		
JASPIC		
JAX-RS		
JAX-WS	SAAJ	
JAX-RPC	SAAJ	

  New in Java EE 6

<b>EJB Container</b>	JSR 330	<b>Java SE</b>
	Interceptors	
	Managed Beans	
	JSR 299	
	Bean Validation	
	JavaMail	
	EJB	
	Java Persistence	
	JTA	
	Connectors	
	JMS	
	Management	
	WS Management	
	Web Services	
	JACC	
	JASPIC	
	JAXR	
JAX-RS		
JAX-WS	SAAJ	
JAX-RPC	SAAJ	

  New in Java EE 6

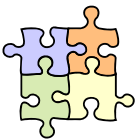
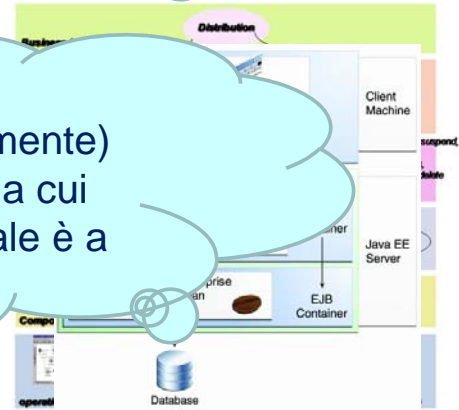


## - Esempio - Architetture a strati

- Molte architetture sono basate su un'organizzazione a strati (Layers)

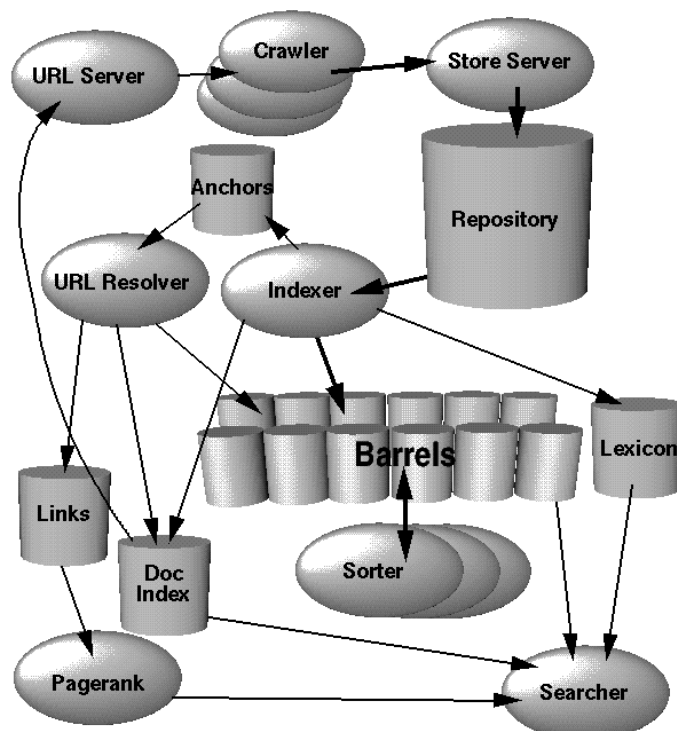
quali vantaggi ci sono ad adottare un'architettura a strati?  
perché? ci sono anche degli svantaggi?

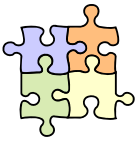
come decomporre (ulteriormente) gli strati di un'architettura la cui organizzazione fondamentale è a strati?



## - Esempio - architettura di Google

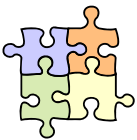
<http://infolab.stanford.edu/~backrub/google.html>





## Esempio - architettura di Google

- In Google, the web crawling (downloading of web pages) is done by several distributed crawlers. There is a URLserver that sends lists of URLs to be fetched to the crawlers. The web pages that are fetched are then sent to the storeserver. The storeserver then compresses and stores the web pages into a repository. Every web page has an associated ID number called a docID which is assigned whenever a new URL is parsed out of a web page. The indexing function is performed by the indexer and the sorter. The indexer performs a number of functions. It reads the repository, uncompresses the documents, and parses them. Each document is converted into a set of word occurrences called hits. The hits record the word, position in document, an approximation of font size, and capitalization. The indexer distributes these hits into a set of "barrels", creating a partially sorted forward index. The indexer performs another important function. It parses out all the links in every web page and stores important information about them in an anchors file. This file contains enough information to determine where each link points from and to, and the text of the link.
- The URLresolver reads the anchors file and converts relative URLs into absolute URLs and in turn into docIDs. It puts the anchor text into the forward index, associated with the docID that the anchor points to. It also generates a database of links which are pairs of docIDs. The links database is used to compute PageRanks for all the documents.
- The sorter takes the barrels, which are sorted by docID (this is a simplification, see Section 4.2.5), and resorts them by wordID to generate the inverted index. This is done in place so that little temporary space is needed for this operation. The sorter also produces a list of wordIDs and offsets into the inverted index. A program called DumpLexicon takes this list together with the lexicon produced by the indexer and generates a new lexicon to be used by the searcher. The searcher is run by a web server and uses the lexicon built by DumpLexicon together with the inverted index and the PageRanks to answer queries.

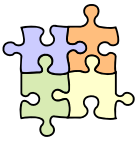


## Esempio - architettura di Google

- In Google, the web crawling (downloading of web pages) is done by several distributed crawlers. There is a URLserver that sends lists of URLs to be fetched to the crawlers. The web pages that are fetched are then sent to the storeserver. The storeserver then compresses and stores the web pages into a repository. Every web page has an associated ID number called a docID which is assigned whenever a new URL is parsed out of a web page. The indexing function is performed by the indexer and the sorter. The indexer performs a number of functions. It reads the repository, uncompresses the documents, and parses them. Each document is converted into a set of word occurrences called hits. The hits record the word, position in document, an approximation of font size, and capitalization. The indexer distributes these hits into a set of "barrels", creating a partially sorted forward index. The indexer performs another important function. It parses out all the links in every web page and stores important information about them in an anchors file. This file contains enough information to determine where each link points from and to, and the text of the link.
- The URLresolver reads the anchors file and converts relative URLs into absolute URLs and in turn into docIDs. It puts the anchor text into the forward index, associated with the docID that the anchor points to. It also generates a database of links which are pairs of docIDs. The links database is used to compute PageRanks for all the documents.
- The sorter takes the barrels, which are sorted by docID (this is a simplification, see Section 4.2.5), and resorts them by wordID to generate the inverted index. This is done in place so that little temporary space is needed for this operation. The sorter also produces a list of wordIDs and offsets into the inverted index. A program called DumpLexicon takes this list together with the lexicon produced by the indexer and generates a new lexicon to be used by the searcher. The searcher is run by a web server and uses the lexicon built by DumpLexicon together with the inverted index and the PageRanks to answer queries.

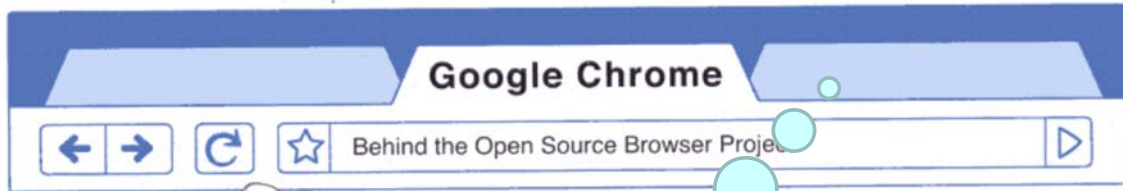
è una vista funzionale, che descrive la decomposizione di Google in termini di elementi funzionali, relativamente all'implementazione dell'algoritmo PageRank

ma come fa Google a soddisfare i requisiti relativi a prestazioni e scalabilità? com'è fatta la vista di deployment?

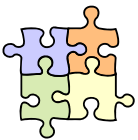


## - Esempio - Google Chrome

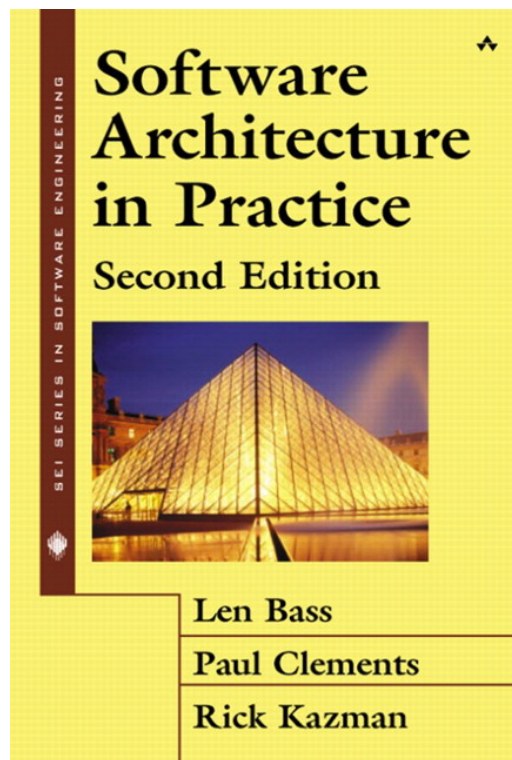
- Google Chrome è un browser progettato per rendere più veloce, facile e sicuro l'uso del Web con un design minimo che non intralcia la navigazione – <http://www.google.com/chrome>

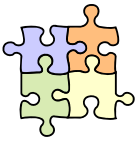


il fumetto di presentazione di Google Chrome descrive alcuni requisiti di qualità chiave affrontati dal progetto, nonché alcune scelte architettoniche significative

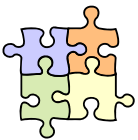


## \* Alcune letture consigliate Architetture software [SAP]

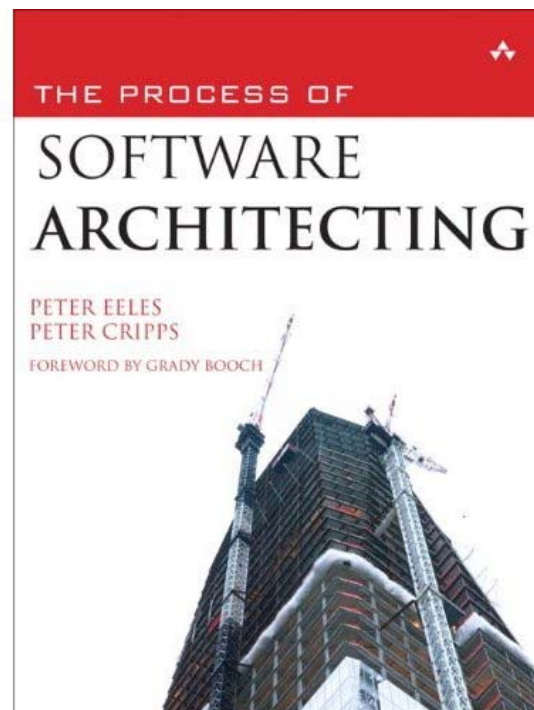


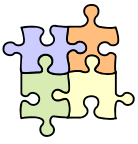


## Alcune letture consigliate Stili architetturali [POSA4]

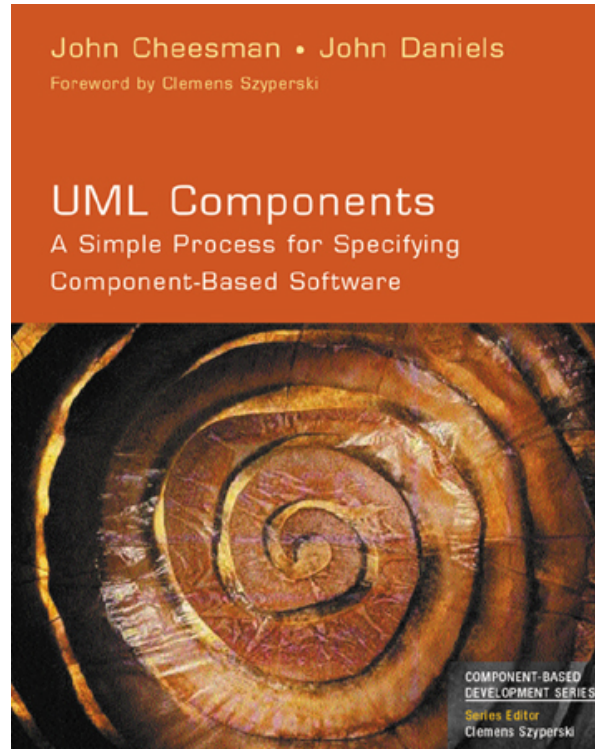


## Alcune letture consigliate Processo di def. dell'arch. sw. [Eeles, Cripps]





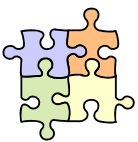
## Alcune letture consigliate Components [UML Components]



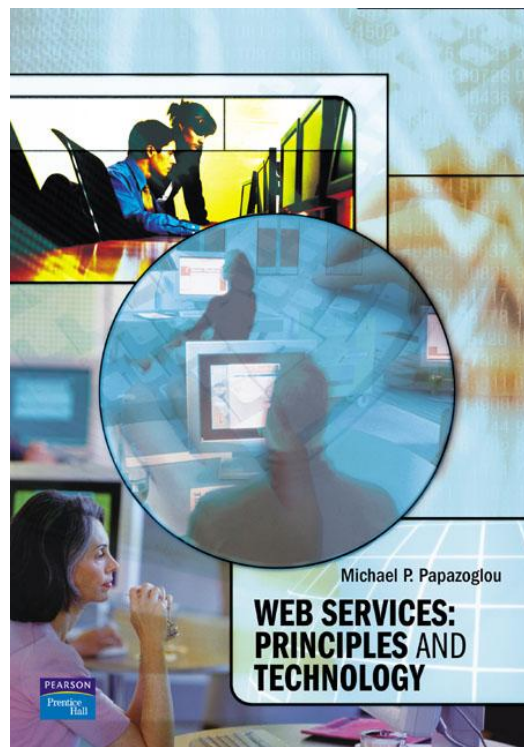
47

Introduzione alle architetture software

Luca Cabibbo - ASw



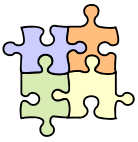
## Alcune letture consigliate Web Services e SOA [Papazoglou]



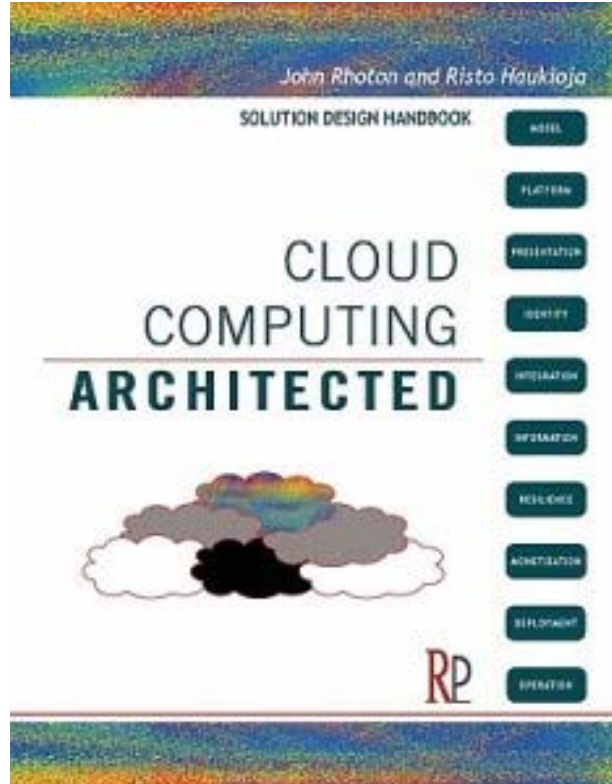
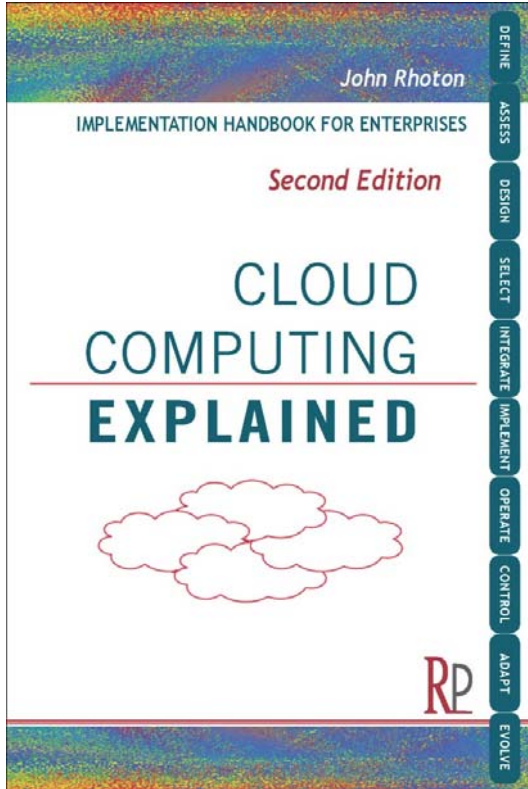
48

Introduzione alle architetture software

Luca Cabibbo - ASw



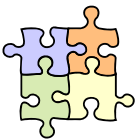
# Alcune letture consigliate Cloud Computing [Rhoton]



49

Introduzione alle architetture software

Luca Cabibbo - ASw



# Alcune letture consigliate Standard per arch. sw. [IEEE 1471-2000]

IEEE Std 1471-2000

## IEEE Recommended Practice for Architectural Description of Software-Intensive Systems

Sponsor  
**Software Engineering Standards Committee**  
of the  
**IEEE Computer Society**

Approved 21 September 2000  
**IEEE-SA Standards Board**

**Abstract:** This recommended practice addresses the activities of the creation, analysis, and sustainment of architectures of software-intensive systems, and the recording of such architectures in terms of architectural descriptions. A conceptual framework for architectural description is established. The content of an architectural description is defined. Annexes provide the rationale for key concepts and terminology, the relationships to other standards, and examples of usage.

**Keywords:** architectural description, architecture, software-intensive system, stakeholder concerns, system stakeholder, view, viewpoint

The Institute of Electrical and Electronics Engineers, Inc.  
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2000 by the Institute of Electrical and Electronics Engineers, Inc.  
All rights reserved. Published 9 October 2000. Printed in the United States of America.

Print: ISBN 0-7381-2518-0 SH94869  
PDF: ISBN 0-7381-2519-9 SS94869

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

50

Introduzione alle architetture software

Luca Cabibbo - ASw