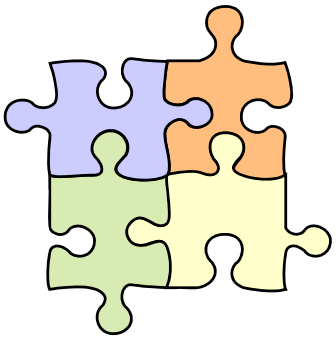
Luca Cabibbo



Architetture Software

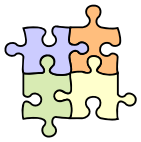
Introduzione alle architetture software

Dispensa ASW 110

ottobre 2014

Un'intuizione vale più di mille analisi.

Charles W. Sooter



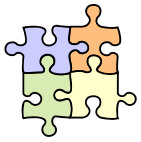
- Fonti

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

- Obiettivi e argomenti

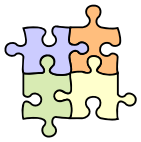
- Obiettivi
  - motivare l'approccio e l'importanza delle architetture software
- Argomenti
  - progettazione di sistemi complessi
  - introduzione alle architetture software
  - l'approccio delle architetture software
  - alcuni esempi
  - letture consigliate

- Wordle



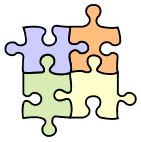
* Progettazione di sistemi complessi

- In questo corso siamo interessati alla progettazione e realizzazione di sistemi (software) complessi
  - questi sistemi devono soddisfare numerosi interessi, di diverse parti interessate – interessi funzionali, sociali e ambientali, funzionamento in condizioni specifiche, sicurezza delle persone, ... – nonché altri interessi pragmatici – tempi, costi, ...
  - in generale, le discipline che si occupano della progettazione e costruzione di sistemi complessi sono sia l'architettura che l'ingegneria
- 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 architetturale
  - in parte scienza, in parte arte



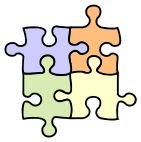
Che cosa è l'architettura

- **Architecture** [Wikipedia, ottobre 2012] is both the process and product of planning, designing and construction. Architectural works, in the material form of buildings, are often perceived as cultural symbols and as works of art. Historical civilizations are often identified with their surviving architectural achievements.
- "Architecture" can mean:
  - A general term to describe buildings and other physical structures.
  - The art and science of design and erecting buildings and other physical structures.
  - A style and method of design and construction of buildings and other physical structures.
  - The practice of an architect, ...
  - The design activity of the architect, from the macro-level (urban design, landscape architecture) to the micro-level (construction details and furniture).
  - 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.



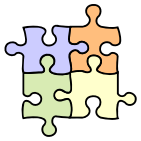
Che cosa è l'ingegneria

- **Engineering** [Wikipedia, ottobre 2012] is the science, skill and profession of acquiring and applying scientific, economic, social, and practical knowledge, in order to design and also build structures, machines, devices, systems, materials and processes.
- A possible definition of "engineering" is:
  - The creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination; or to construct or operate the same with full cognizance of their design; or to forecast their behavior under specific operating conditions; all as respects an intended function, economics of operation 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 sub disciplines, each with a more specific emphasis on certain fields of application and particular areas of technology.



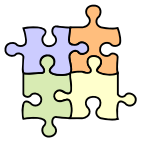
Ingegneria e architettura a confronto

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



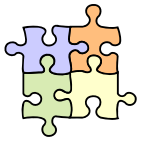
Progettazione di sistemi complessi

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



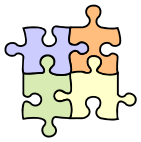
Sistemi e complessità

- *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
- Entrambe le definizioni parlano di *parti* o *elementi*, nonché di *interrelazioni* e *interconnessioni* tra le parti



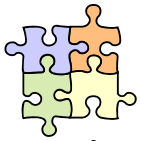
Affrontare la complessità

- 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”
  - ma 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
- La scelta delle parti e delle interconnessioni tra di esse definisce la “struttura” di un sistema



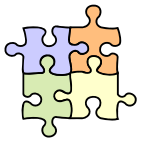
Non solo struttura

- L’architettura non è interessata *solo* alla struttura dei sistemi
  - è interessata soprattutto all’utilità dei sistemi realizzati
- Ciascun sistema è utile nella misura in cui consente di perseguire e ottenere dei risultati
  - da una parte, è importante comprendere quali sono i risultati desiderati – i motivi per cui un sistema viene costruito – per questo, la definizione dell’architettura deve essere basata sugli interessi delle parti interessate al sistema
  - dall’altra, è necessario comprendere le relazioni tra struttura e qualità
  - inoltre, la caratterizzazione del problema da affrontare spesso non è fissata o ben definita fin dall’inizio – il processo architetturale deve consentire di identificare una coppia problema-soluzione, per massimizzare la soddisfazione delle parti interessate



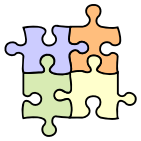
* Introduzione alle architetture software

- I grandi sistemi software di oggi sono tra le strutture più complesse mai costruite dagli uomini – e la loro complessità è via via crescente
  - contengono milioni di linee di codice, centinaia di componenti e di tabelle nelle basi di dati, e sono eseguiti da dozzine di computer
  - 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 – e operano su periodi di tempo estesi
  - ciò presenta a questi team delle sfide formidabili – se queste sfide non vengono affrontate 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



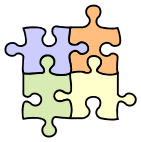
Funzionalità e qualità del software

- 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, ...*
  - *interoperabilità* – la capacità di far 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, ...*



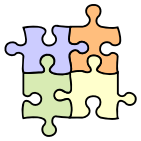
Importanza delle qualità del software

- 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
- Pertanto, 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



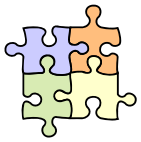
Architetture software

- 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 architetturale 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 a 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



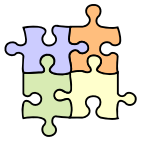
Che cosa sono le architetture software?

- Ecco un paio di possibili definizioni di “architettura software”
  - 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 [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 un’influenza significativa sulle qualità del sistema**



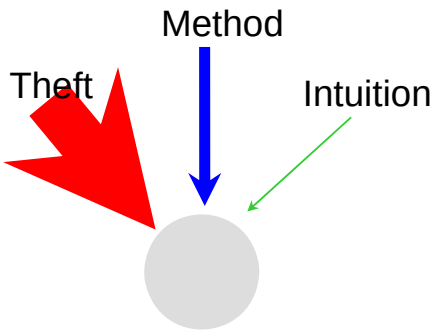
* L’approccio delle architetture software

- 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 scelta e la negoziazione 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

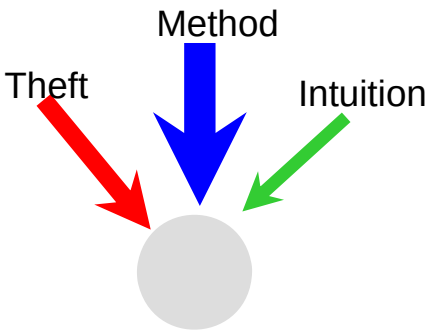


Le sorgenti delle architetture software

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

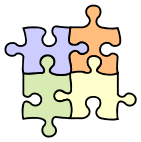


A classical system



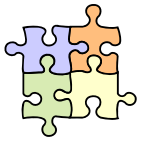
An unprecedented system

  - 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



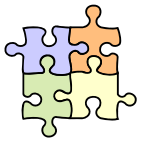
Progettazione di un'architettura software

- In pratica, la progettazione di un'architettura software può essere basata sull'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
  - stili architetturali
    - esperienze significative nella realizzazione di architetture
  - tattiche e prospettive architetturali
    - approcci per la valutazione e la (ri)strutturazione dell'architettura del sistema, con riferimento a qualità specifiche del sistema



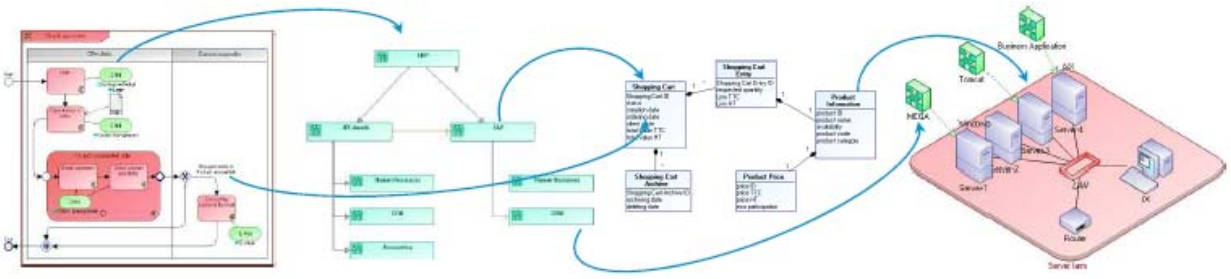
Viste e punti di vista

- 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 architetturali* – indipendenti ma correlate
  - ciascuna vista architetturale 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, ...

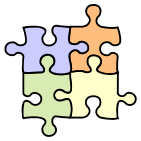


Viste e punti di vista

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

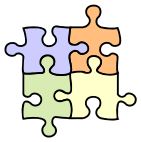


  - qui sono utilizzate quattro viste: dei processi di business, funzionale, delle informazioni, di deployment
  - da notare, all'interno di ogni vista, l'uso di elementi e di relazioni tra elementi
  - da notare anche l'importanza delle relazioni tra elementi di viste diverse



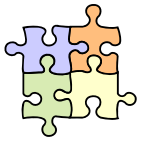
Non solo struttura

- Le viste architetturali consentono di descrivere in modo efficace la struttura di un sistema – ma l'architettura non è interessata *solo* alla struttura dei sistemi
  - è interessata soprattutto alle scelte di progetto che conducono alla definizione di una particolare struttura anziché di un'altra
  - è interessata anche alla "giustificazione" delle scelte di progetto e della struttura



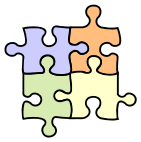
Stili architetturali

- 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*) *architetturale* [POSA] codifica un'esperienza significativa nella realizzazione di un'architettura software
- Uno stile architetturale
  - affronta un problema – un tipo di sistema, con una certa combinazione di requisiti (soprattutto 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



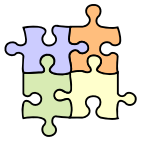
Tattiche e prospettive architetturali

- 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 architetturali sono delle ulteriori guide per la progettazione per le qualità di un sistema
  - una *tattica architetturale* [SAP] è una decisione di progetto che influenza il controllo di un attributo di qualità
  - una *prospettiva architetturale* [SSA] è una collezione di attività, tattiche e linee guida per far sì che un sistema esibisca un insieme di proprietà di qualità correlate
  - si applicano per verificare il livello di qualità raggiunto, nonché per guidare il cambiamento delle viste (quando richiesto)



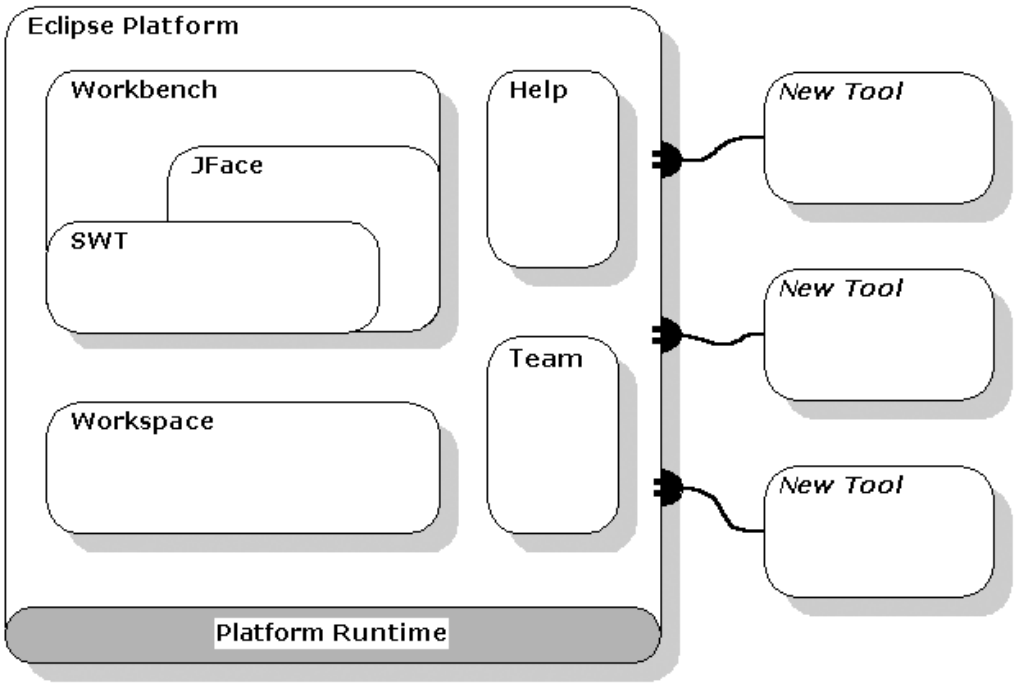
* Alcuni esempi

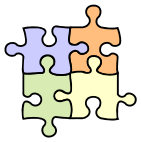
- Seguono alcuni esempi di architetture software
  - l'enfasi è sulle qualità, gli elementi architetturali e le loro relazioni
  - alcuni di questi casi saranno analizzati meglio nel seguito del corso



– Esempio – Piattaforma Eclipse

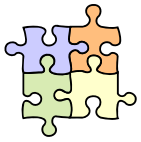
http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.pdf (2006)





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®, LinuxTM, 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.

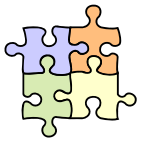


Piattaforma Eclipse – obiettivi

- 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 [illegible] Java, C, JSP, EJB, XM[illegible]
  - Facilitat[illegible] tent ty[illegible]
  - Su[illegible] ments.
  - Run[illegible] nuxTM, Mac[illegible]
  - Capitalize on [illegible] guage 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.

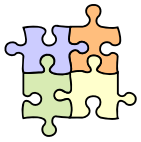
È la descrizione degli obiettivi della piattaforma Eclipse.

Notare 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. ...



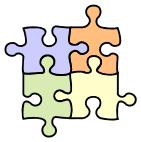
Piattaforma Eclipse – architettura a plug-in

- A plug-in is the smallest unit of Eclipse Pla… developed and delivered separately … g-in, whereas … Except for a s… 's functi…
- Plug… Arch… …es,
- …
- …d, …
- …g a … to activate any … of pertinent informatio…

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.

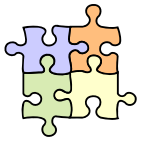
Descrive le caratteristiche esterne che deve fornire un plug-in, nonché le modalità di comunicazione tra ciascun plug-in e il Platform Runtime.

Descrive il raggiungimento degli obiettivi di qualità sulla base di questa organizzazione.



- Esempio – Java Enterprise Edition

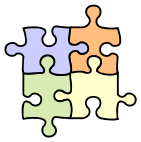
- Da Java EE Tutorial – docs.oracle.com/javaee/7/tutorial/doc/
  - Developers today increasingly recognize the need for distributed, transactional, and portable applications that leverage the speed, security, and reliability of server-side technology. Enterprise applications provide the business logic for an enterprise. They are centrally managed and often interact with other enterprise software. 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 shortening development time, reducing application complexity, and improving application performance.
  - …



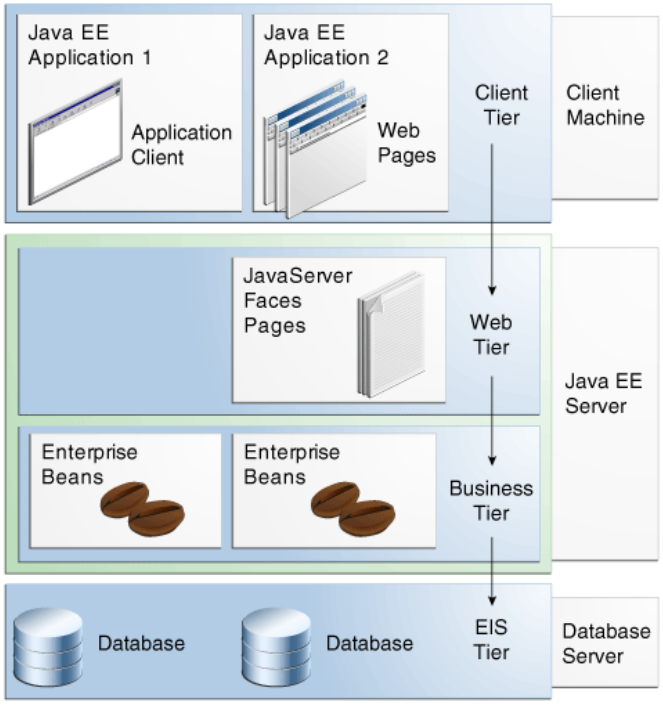
- Esempio – Java Enterprise Edition

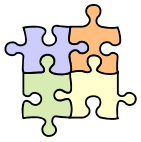
- Da Java EE Tutorial – docs.oracle.com/javaee/7/tutorial/doc/
  - Developers today increasingly recognize the need for distributed, transactional, and portable applications that leverage the speed, s… …ver-side technol… …ss logic for an… …n interact w… …n tec… …uilt, and pro… …with fewer resources…
  - With the Java Platform, E… …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 shortening 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





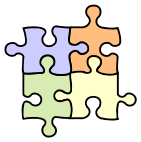
Modello applicativo per Java EE

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

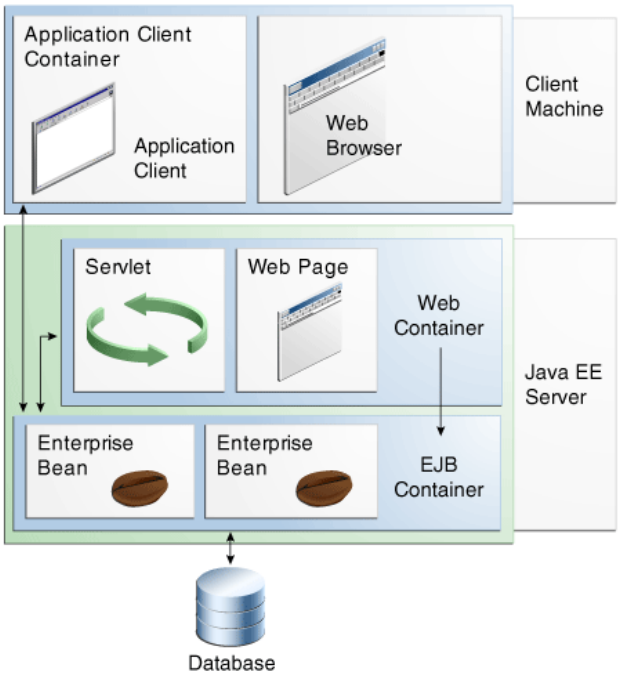
Ma come effettuare la decomposizione architetturale di un'applicazione se la tecnologia target è Java EE?

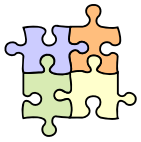
Quali qualità possiedono le applicazioni per la piattaforma Java EE?

Come sono assicurate queste qualità?

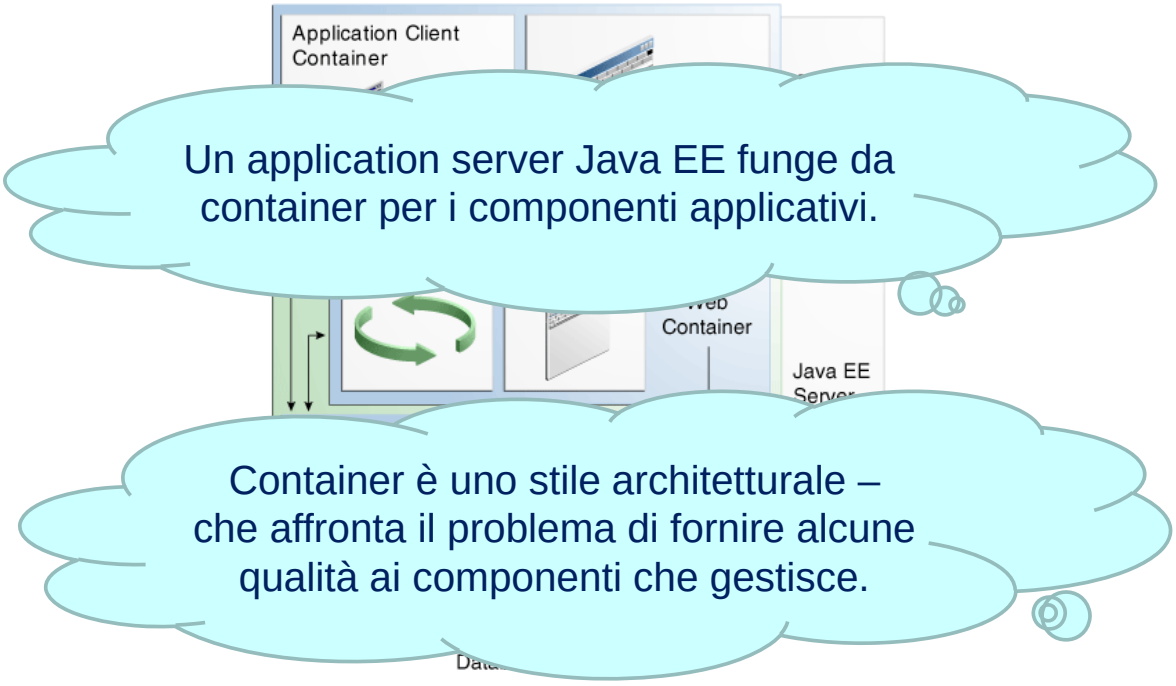


Container Java EE



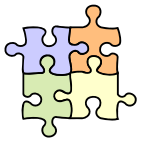


Container Java EE



Un application server Java EE funge da container per i componenti applicativi.

Container è uno stile architetturale – che affronta il problema di fornire alcune qualità ai componenti che gestisce.

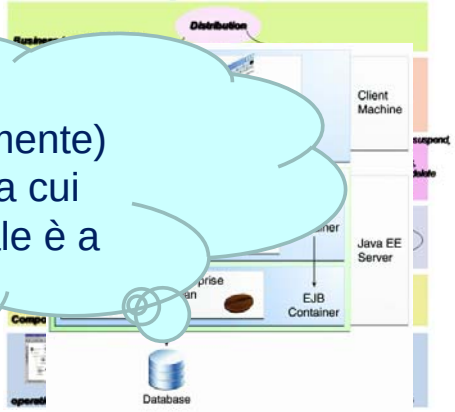


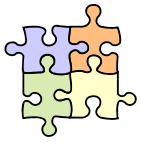
- Esempio – Architetture a strati

- 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?

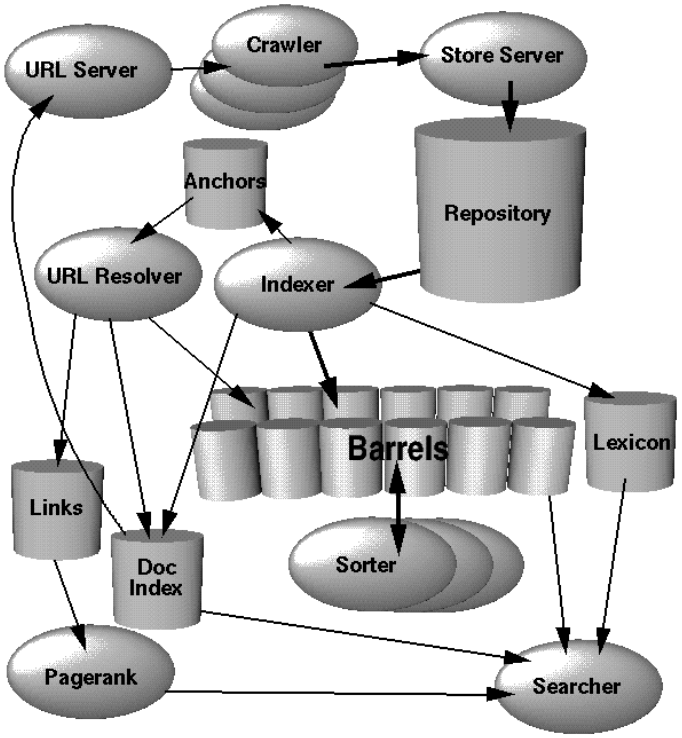
Come decomporre (ulteriormente) gli strati di un'architettura la cui organizzazione fondamentale è a strati?

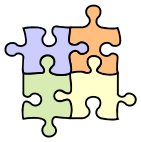




- Esempio – architettura di Google

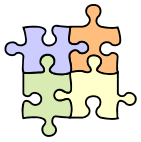
http://infolab.stanford.edu/~backrub/google.html





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.

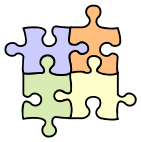


Esempio – architettura di Google

- In Google, the web ... rawlers. There is a URLserve... hed are then sent repositor... ver a ne... orter. ...nd ...the wo... th... another in... ormation about then... link points from and to, and th...
- The URLresolver reads ... RLs into absolut... s and in turn into docIDs. It puts the anchor text into the forward ..., 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 ta... 4.2.5), and resorts th... space is need... index. g... ne lex... nes.

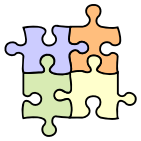
È una vista funzionale, che descrive la decomposizione di Google in termini di elementi funzionali, relativamente all'implementazione dell'algoritmo PageRank.

Ma come fa Google a soddisfare i requisiti relativi a prestazioni e scalabilità? Com'è fatta la vista di deployment? Come sono gestiti i dati?



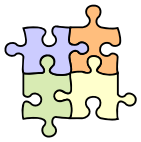
- Esempio – MapReduce

- PageRank è un *algoritmo* per l'indicizzazione di pagine web – che opera su pagine che sono state scaricate e memorizzate localmente in un grande cluster di computer
  - ma come sfruttare questo cluster per eseguire questo algoritmo in modo efficiente, scalabile e affidabile?
  - e questa capacità può essere trasferita ad un insieme più ampio di algoritmi?

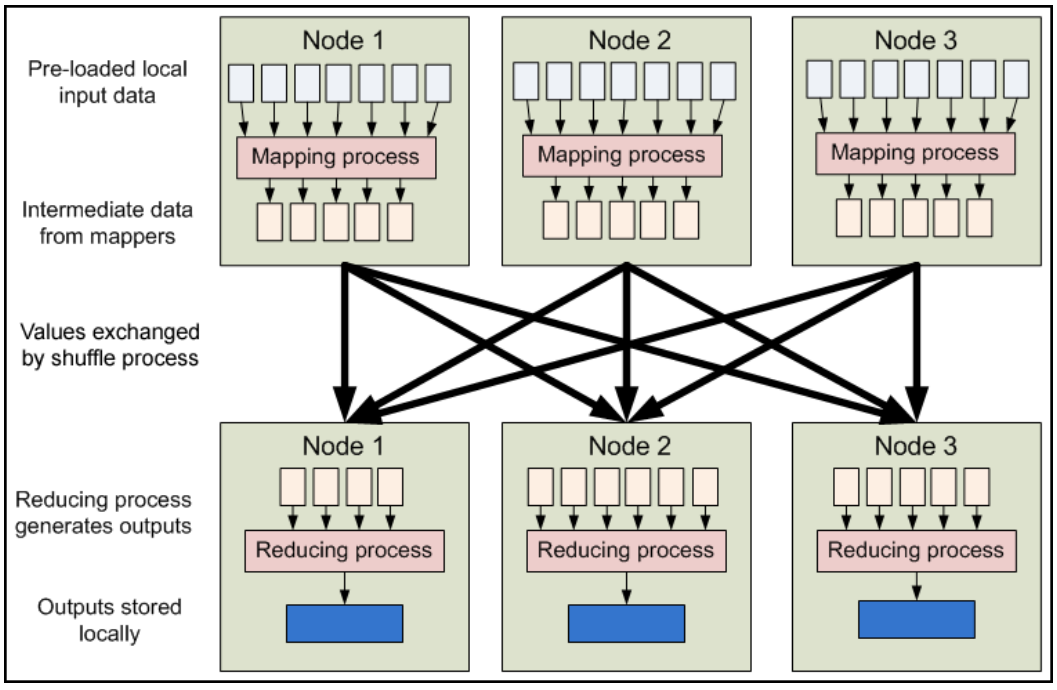


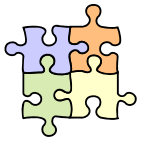
Esempio – MapReduce

- MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model.
- Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.
- Our implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. Programmers find the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day.



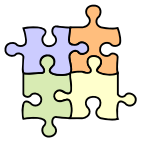
Esempio – MapReduce





- Esempio – Amazon Elastic Beanstalk

- Java EE è una tecnologia per applicazioni enterprise che, originariamente, era necessario eseguire su un proprio cluster di computer
  - ma posso ottenere le stesse qualità per applicazioni da eseguire su una piattaforma di cloud computing?
- La risposta corretta è (oggi) “no, ma quasi”
  - in generale, infatti, per ottenere alcune qualità è spesso necessario rinunciare ad altre qualità
  - ad esempio, le piattaforma applicative per il cloud computing spesso forniscono un supporto specifico per “applicazioni semplici” – sostenendo alcune qualità (ad esempio, scalabilità, semplicità di gestione) a scapito di altre qualità (ad esempio, relative alla consistenza, nel contesto della gestione della persistenza e delle transazioni)



- Esempio – Amazon Elastic Beanstalk

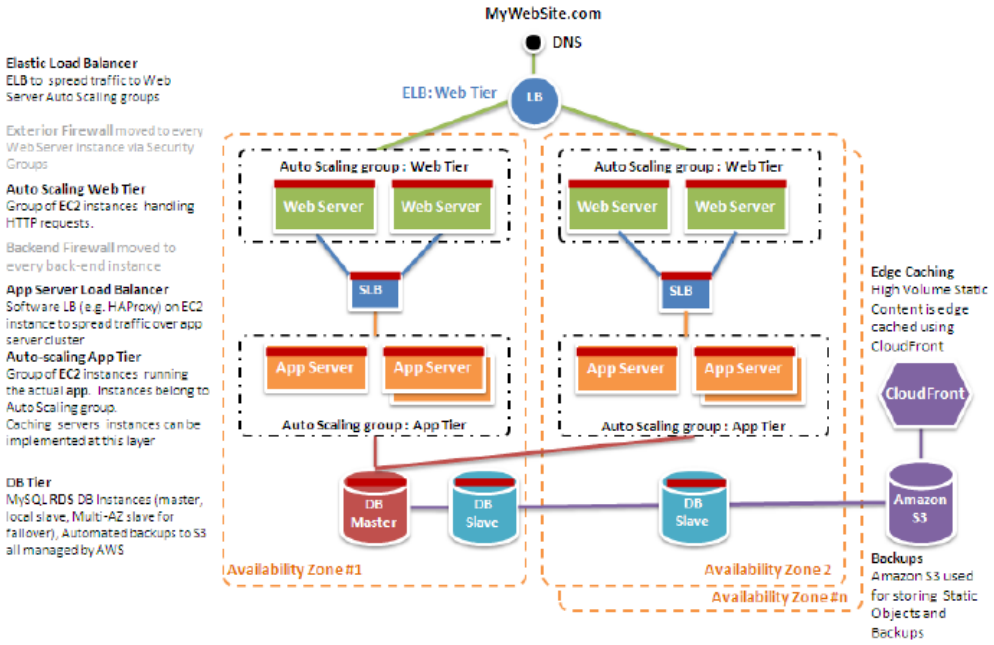
- AWS Elastic Beanstalk – http://aws.amazon.com/elasticbeanstalk/
  - AWS Elastic Beanstalk is an even easier way for you to quickly deploy and manage applications in the AWS cloud. **You simply upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring.** At the same time, with Elastic Beanstalk, you retain full control over the AWS resources powering your application and can access the underlying resources at any time. Elastic Beanstalk leverages AWS services such as Amazon Elastic Cloud Compute (Amazon EC2), Amazon Simple Storage Service (Amazon S3), (Amazon SNS), me ic

Come fa Amazon AWS a garantire queste qualità?

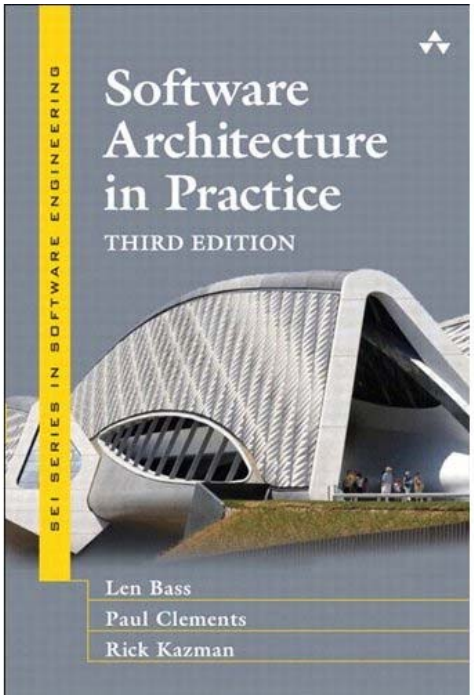
Ma a quali qualità stiamo rinunciando?

Esempio – Amazon Elastic Beanstalk

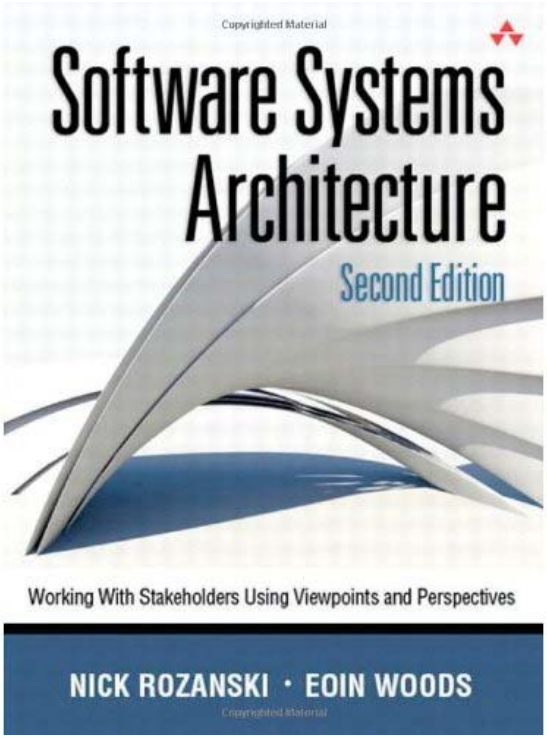
- AWS Elastic Beanstalk – http://aws.amazon.com/elasticbeanstalk/



* Alcune letture consigliate
Software Architecture in Pratice [SAP]

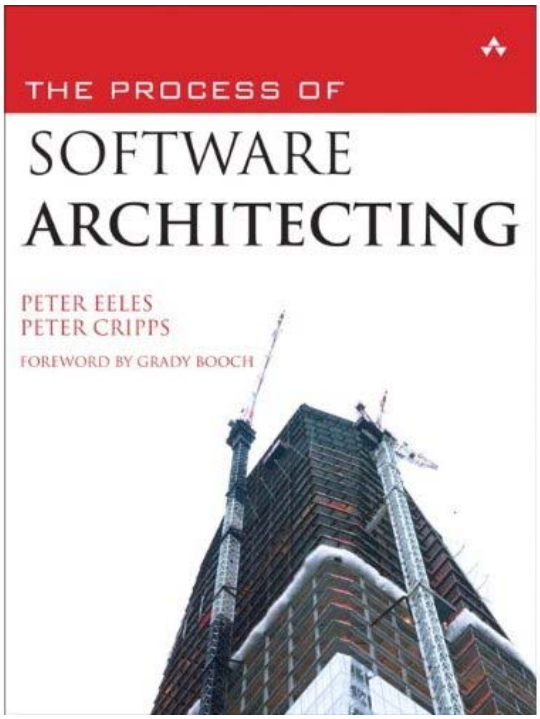


Alcune letture consigliate
Software Systems Architecture [SSA]

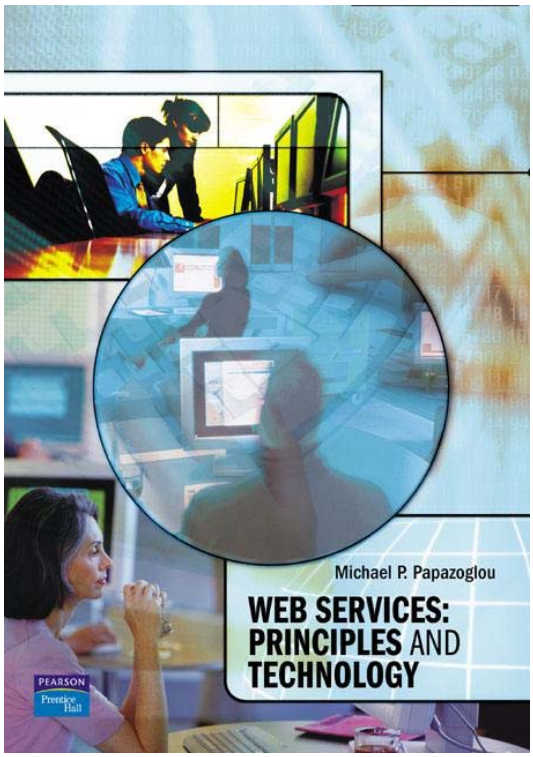


Alcune letture consigliate
Pattern-Oriented Sw Architecture [POSA4]

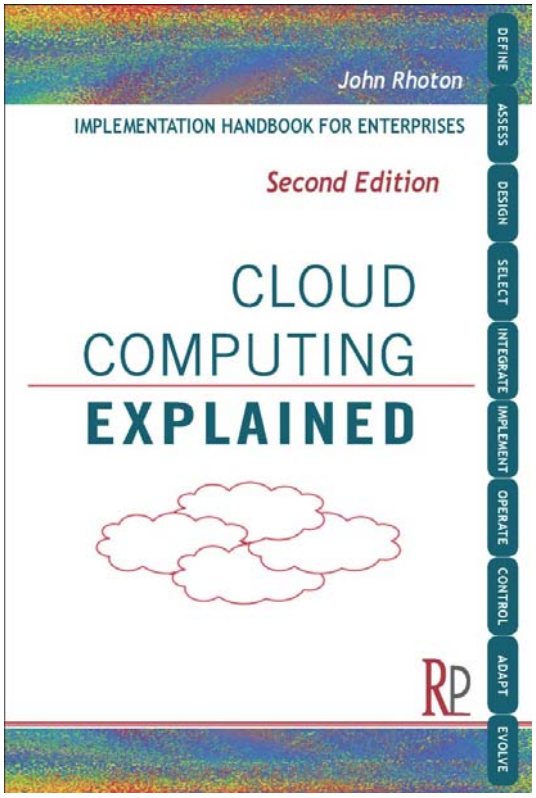
Alcune letture consigliate
The Process of Sw Architecting [PSA]



Alcune letture consigliate
Web Services [Papazoglou]



Alcune letture consigliate
Cloud Computing [Rhoton]



Alcune letture consigliate
Uno standard [ISO/EIC/IEEE 42010-2010]

INTERNATIONAL STANDARD

ISO/IEC/ IEEE 42010

First edition
2011-12-01

Systems and software engineering — Architecture description

Ingénierie des systèmes et des logiciels — Description de l'architecture