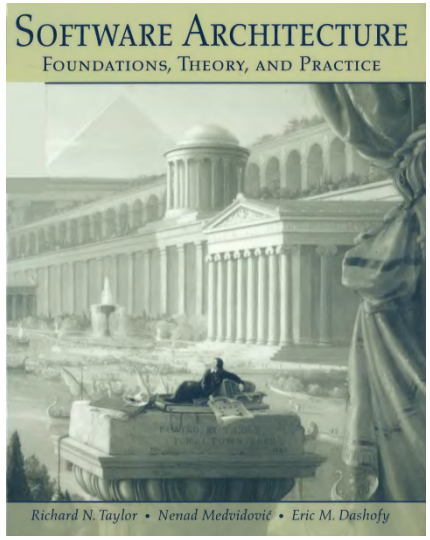
Architetture Software: Concetti Fondamentali

Rif. Cap. 3 –Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy



Sommario

- Definizioni di Architettura Software
- Elementi fondamentali di una Architettura
- Stili Architetturali
- Pattern Architetturali
- Design Pattern
- Modelli, Notazioni e Prospettive Architetturali
- Qualità di una Architettura Software

Architettura Software : definizioni

- Il sito del SEI (Software Engineering Institute@Carnegie Mellon) elenca più di 150 diverse definizioni del termine “architettura software”
  - http://www.sei.cmu.edu/architecture/start/glossary/community.cfm

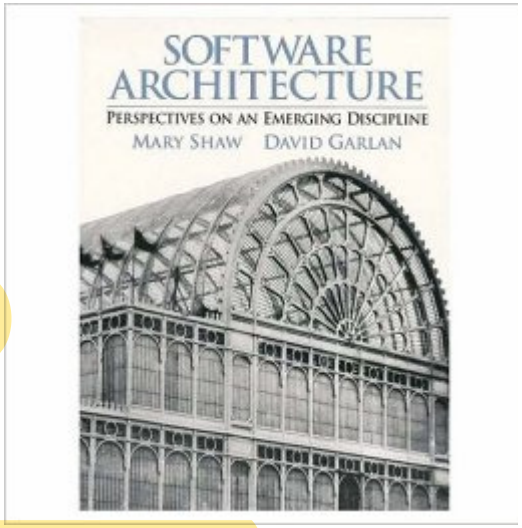
- ne presentiamo alcune tratte dalla letteratura

Una prima definizione (Perry&Wolf)

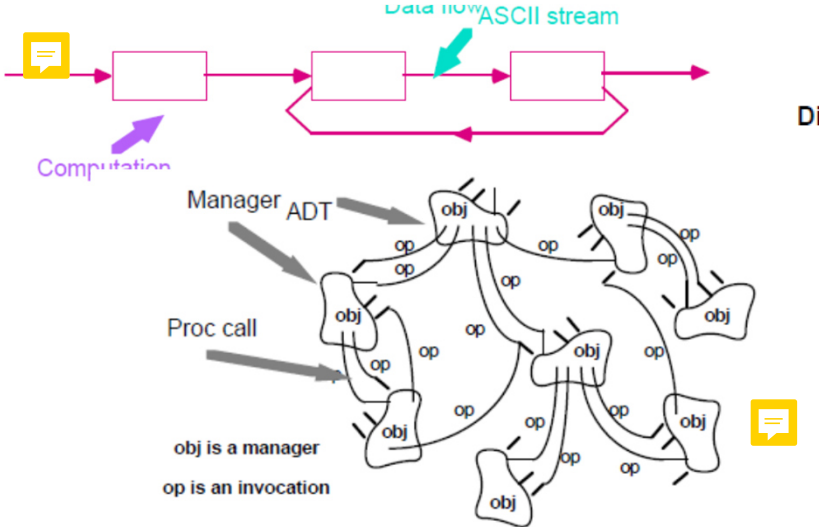
- Una delle prime caratterizzazioni (1992)
  - Software Architecture = { Elements, Form, Rationale }
    what *how* *why*
- un'architettura software è
  - un insieme di elementi architetturali
  - utilizzati secondo una particolare forma (nel senso di organizzazione, strutturazione)
  - insieme a una giustificazione logica che coglie la motivazione per la scelta degli elementi e della forma

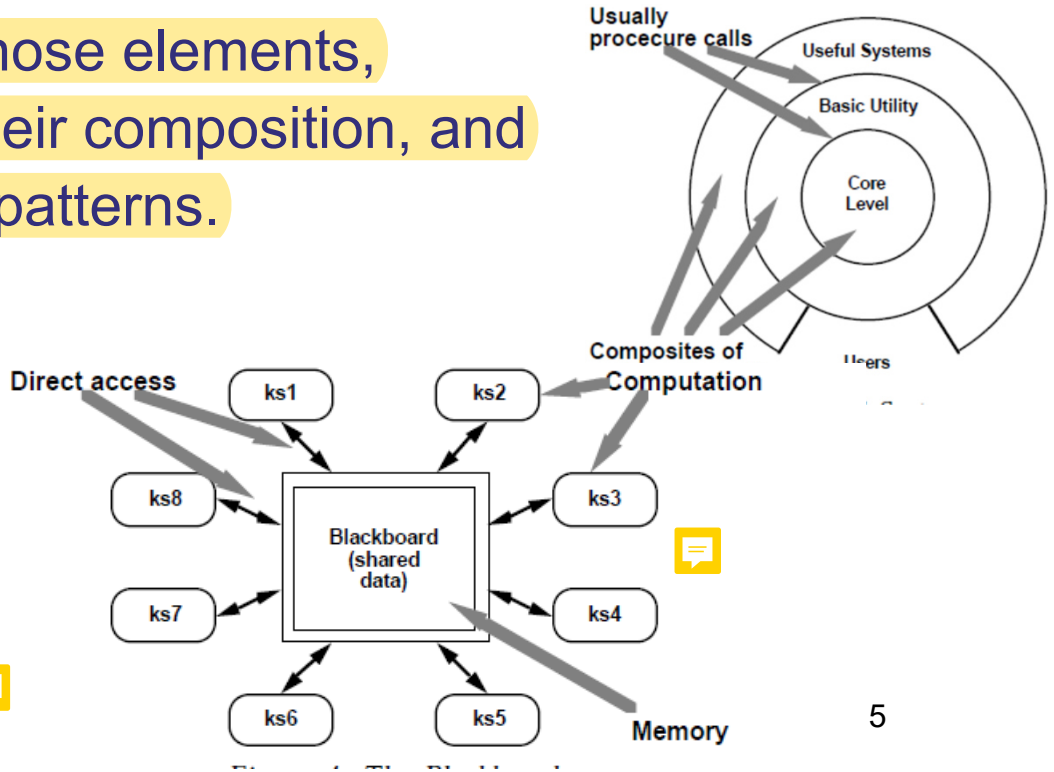
  - *ad esempio, che cosa sapete dell'architettura a strati?*

Garlan e Shaw (1993)



- Software architecture [is a level of design that] involves:
  - the description of elements from which systems are built,
  - interactions among those elements,
  - patterns that guide their composition, and
  - constraints on these patterns.





Booch, Rumbaugh, and Jacobson (1999)

- An architecture is the set of significant decisions about the organization of a software system,
- the selection of the structural elements and their interfaces by which the system is composed,
- together with their behavior as specified in the collaborations among those elements,
- the composition of these structural and behavioral elements into progressively larger subsystems,
- and the *architecture style* that guides this organization, these elements and their interfaces, their collaborations, and their composition

Definizioni dagli Standard IEEE

- **IEEE 1471- 2000**: «**Recommended Practice for Architectural Description for Software-Intensive Systems**”
  - The fundamental organization of a system embodied in its components, their relations to each other, and to the environment, and the principles guiding its design and evolution.

- **ISO/IEEE 42010 (2011) “Systems and software engineering — Architecture description”**
  - system fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution

L'approccio di Kruchten [1995]: 4+1 Views of Software Architectures

Software architecture =

{Elements, Forms, Rationale/Constraints}

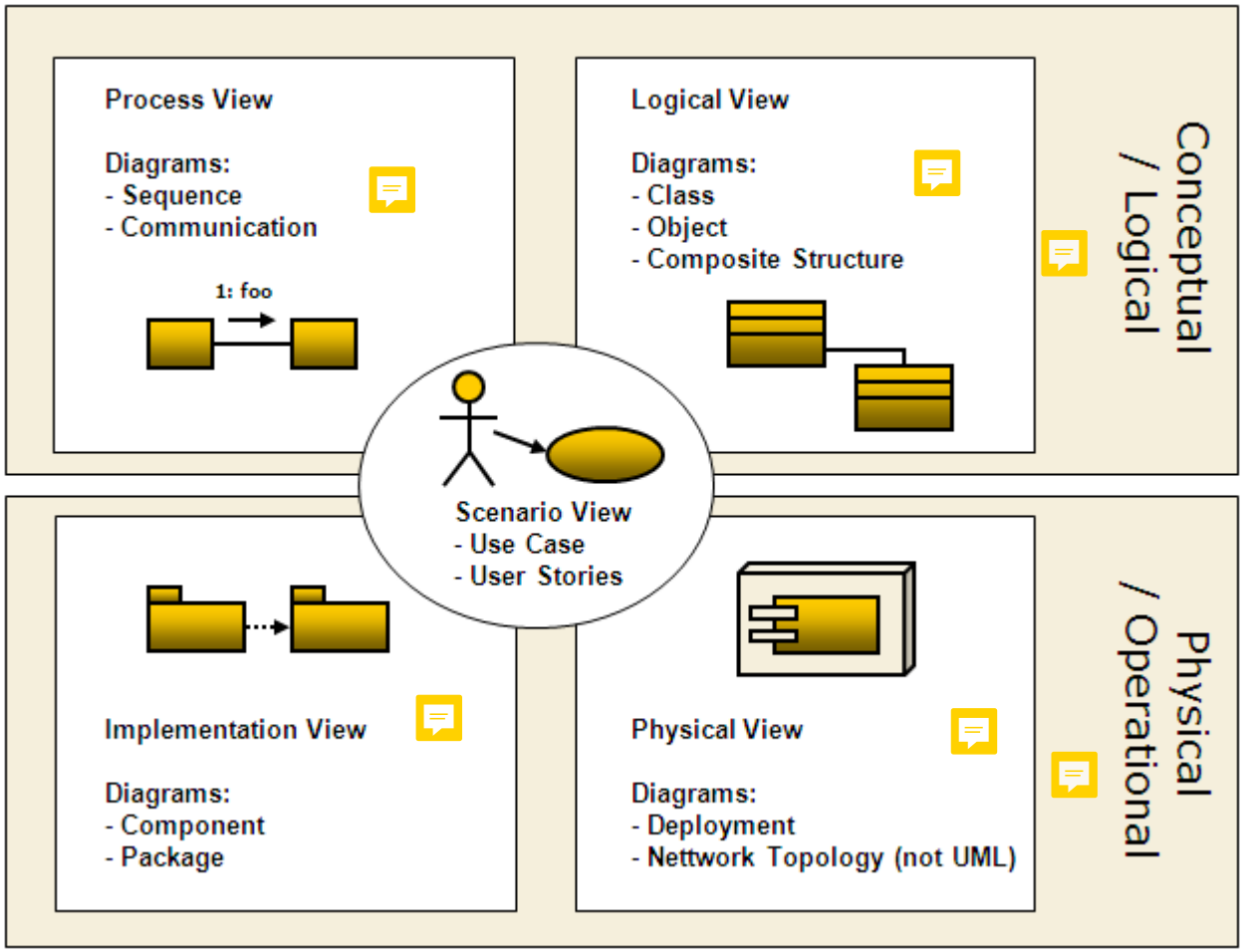
- Software architecture deals with abstraction, with decomposition and composition, with style and esthetics. To describe a software architecture, we use a model composed of multiple views or perspectives…

  - Logical View
  - Process View
  - Physical View
  - Development View
  - Scenarios View

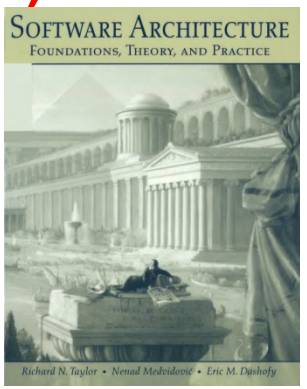
Architettura : Vista logica, Vista di processo, Vista Fisica, Vista Implementativa, Vista degli Scenari

4+1 Views of Software Architectures [Kruchten, 1995]

L'architettura viene specificata secondo 5 viste che coprono i diversi aspetti di una architettura.



Taylor, Medvidovic, and Dashofy (2009)



- L'architettura di un sistema software è l'insieme di *principali decisioni di progetto* relative al sistema

- Costituisce il modello (o blueprint) per la costruzione e successiva evoluzione di un sistema software

- Le *decisioni di progetto* riguardano ogni diverso aspetto del sistema che si sta sviluppando:
  - Struttura: …sarà a livelli
  - Comportamento: …ordine di esecuzione…
  - Interazione: …basata su eventi
  - Proprietà non-funzionali …
  - Implementazione: ..

Perchè “Principali decisioni di progetto”?

- Ovviamente non tutte le decisioni di progetto riguardano l’architettura…
  - Ad esempio le decisioni sulle strutture dati e sugli algoritmi non rientrano nell’architettura
    - Sono piuttosto decisioni di low level design

- Ogni architetto del Software stabilirà quali *principali decisioni* includere nell’architettura software.

Aspetto temporale

- Le decisioni di progetto saranno prese e riprese più volte durante la vita di un sistema

  → l'architettura ha una durata definita nel tempo

- In ogni istante il sistema ha una sola architettura

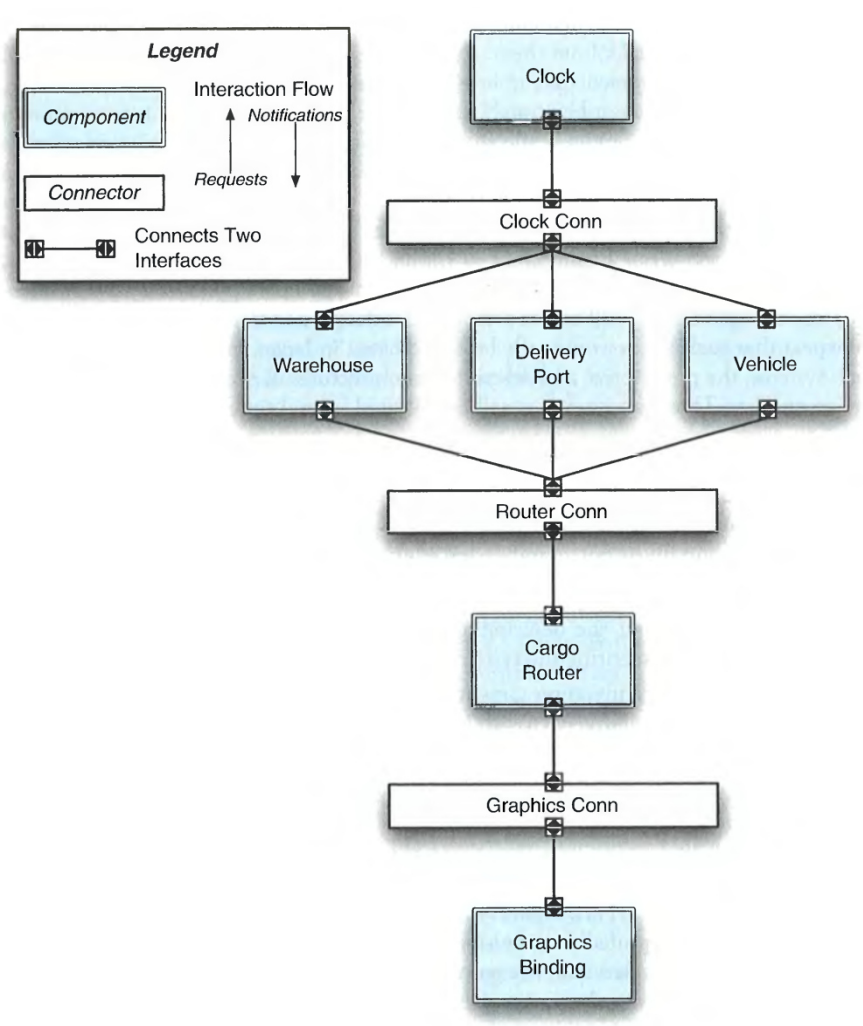
- Ma l'architettura cambierà nel tempo …

Architetture Prescrittive e Descrittive

- *Un'architettura prescrittiva* cattura le decisioni di progetto fatte *prima* di costruire il sistema
  - Rappresenta come essa è stata concepita o intesa (as-designed)

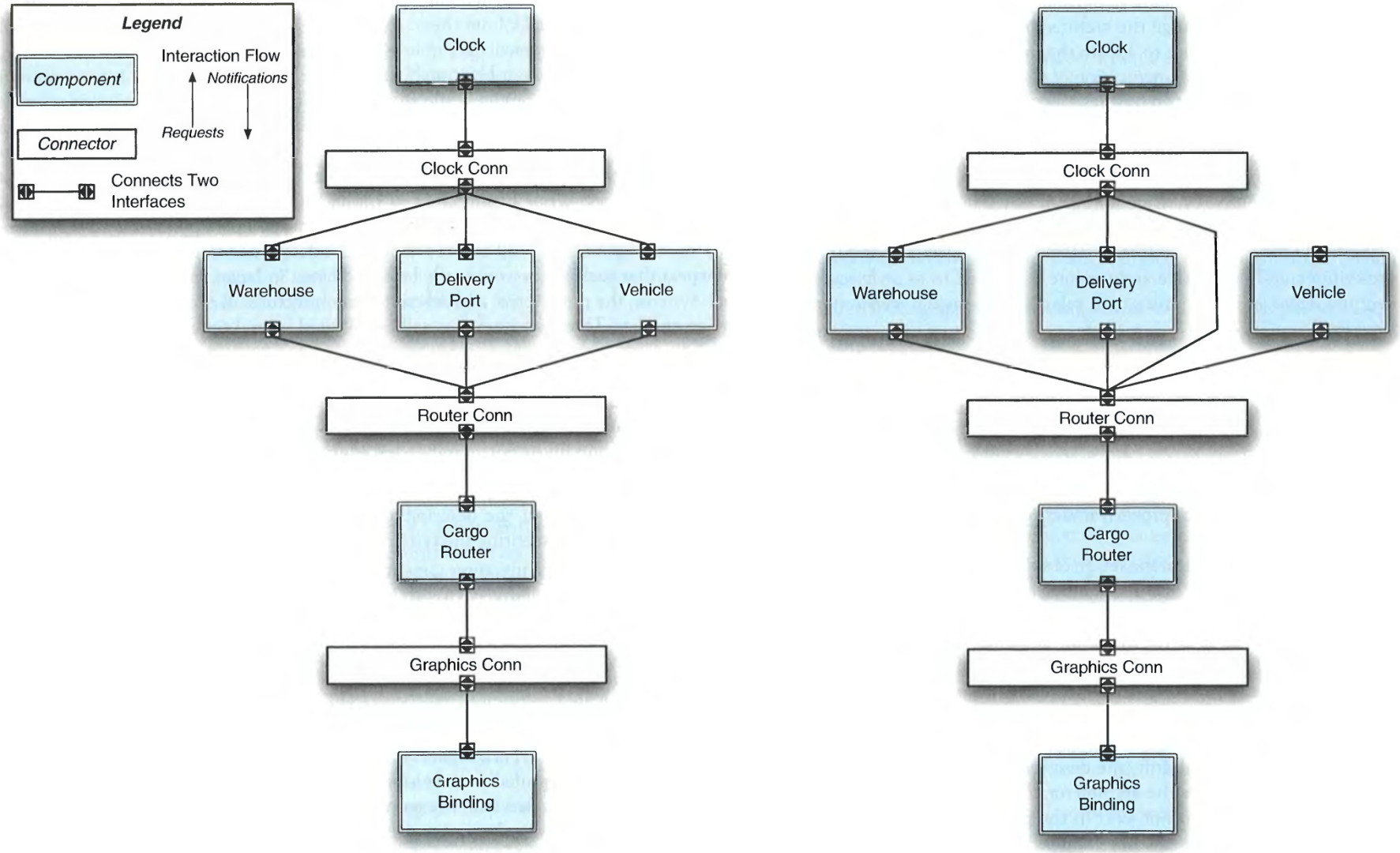
- *Un'architettura descrittiva* descrive come il sistema *è stato* realmente costruito
  - È l'architettura come è stata implementata o realizzata (as-implemented)

As-Designed vs. As-Implemented Architecture

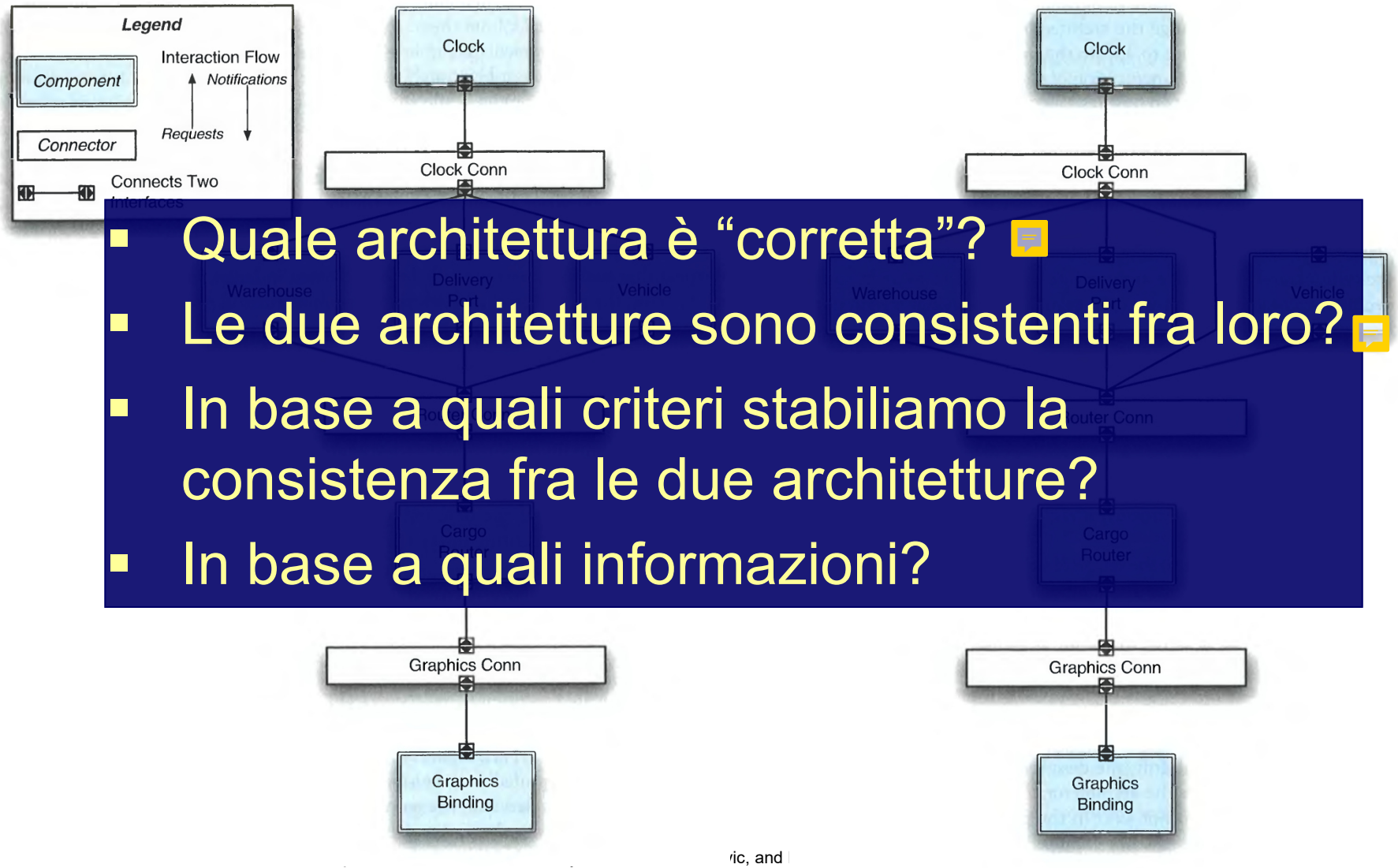


Esempio: Una architettura
A Componenti e Connettori

As-Designed vs. As-Implemented Architecture



As-Designed vs. As-Implemented Architecture



- Quale architettura è “corretta”?
- Le due architetture sono consistenti fra loro?
- In base a quali criteri stabiliamo la consistenza fra le due architetture?
- In base a quali informazioni?

Evoluzione Architetturale

- Quando un sistema evolve, teoricamente la sua architettura prescrittiva dovrebbe essere prima modificata
- In pratica, il sistema - e quindi la sua architettura descrittiva - è spesso modificato direttamente
- Le possibili cause:
  - Trascuratezza dello sviluppatore
  - Scadenze brevi che impediscono di pensare e di documentare
  - Mancanza di una architettura prescrittiva documentata
  - Necessità di ottimizzare il codice
  - Tecniche o strumenti inadeguati

Degrado Architetturale

- Due concetti correlati:
  - Deriva architetturale (Architectural drift)
  - Erosione Architetturale

- *Deriva Architetturale:* è l'introduzione di decisioni di progetto in una architettura descrittiva che
  - Non sono incluse, racchiuse o implicate dall'architettura prescrittiva
  - Ma che *non violano* nessuna delle decisioni dell'architettura prescrittiva
- *Erosione Architetturale* è l'introduzione di decisioni di progetto in una architettura descrittiva che *violano* l'architettura prescrittiva

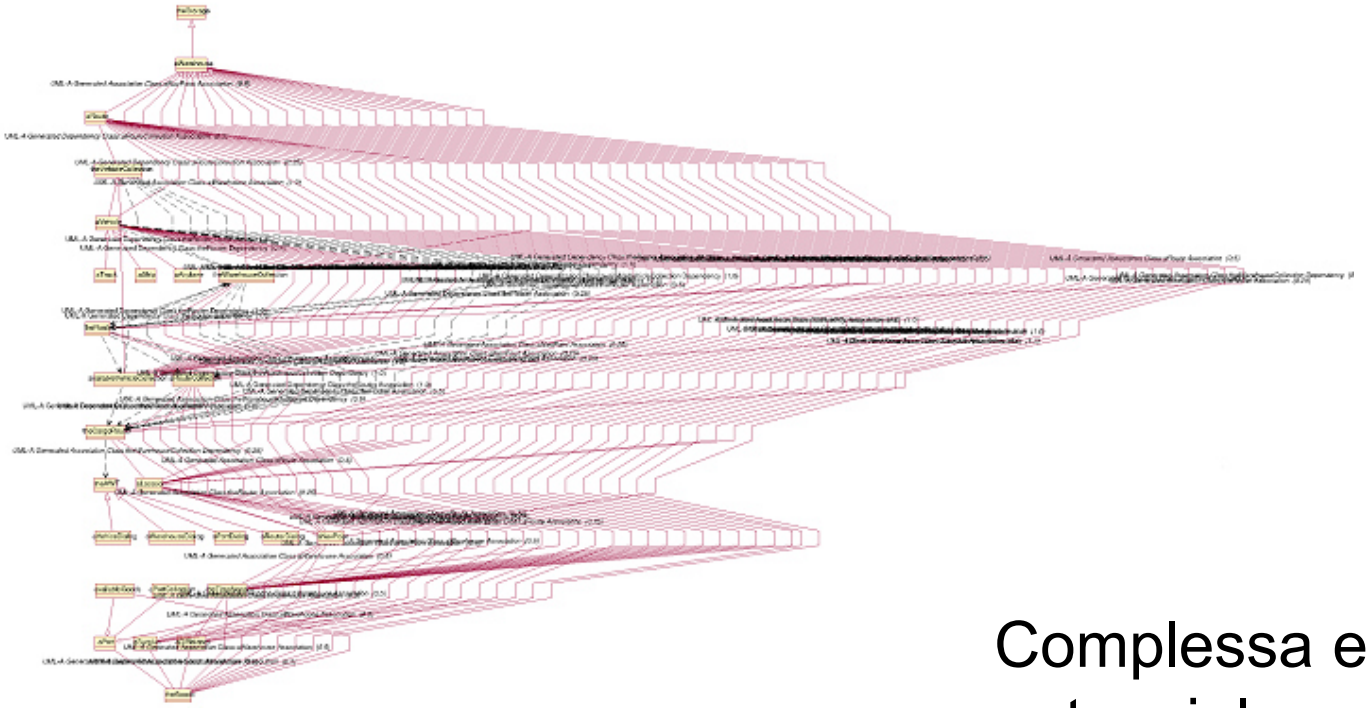
Architectural Recovery

- Se il degrado architetturale è permesso, prima o poi sarà necessario recuperare l'architettura del sistema

- *Architectural recovery* è il processo di recupero dell'architettura a partire dagli artifatti a livello implementativo

- Implementation-level artifacts possono essere:
  - Source code
  - Executable files
  - Java .class files

Implementation-Level View of an Application



Complessa e potenzialmente incomprensibile!

Prospettive Architetturali

- La prospettiva è un modo per evidenziare solo alcuni aspetti di una architettura, tralasciandone altri.

- Una prospettiva è un insieme non vuoto di decisioni di progetto architetturali.

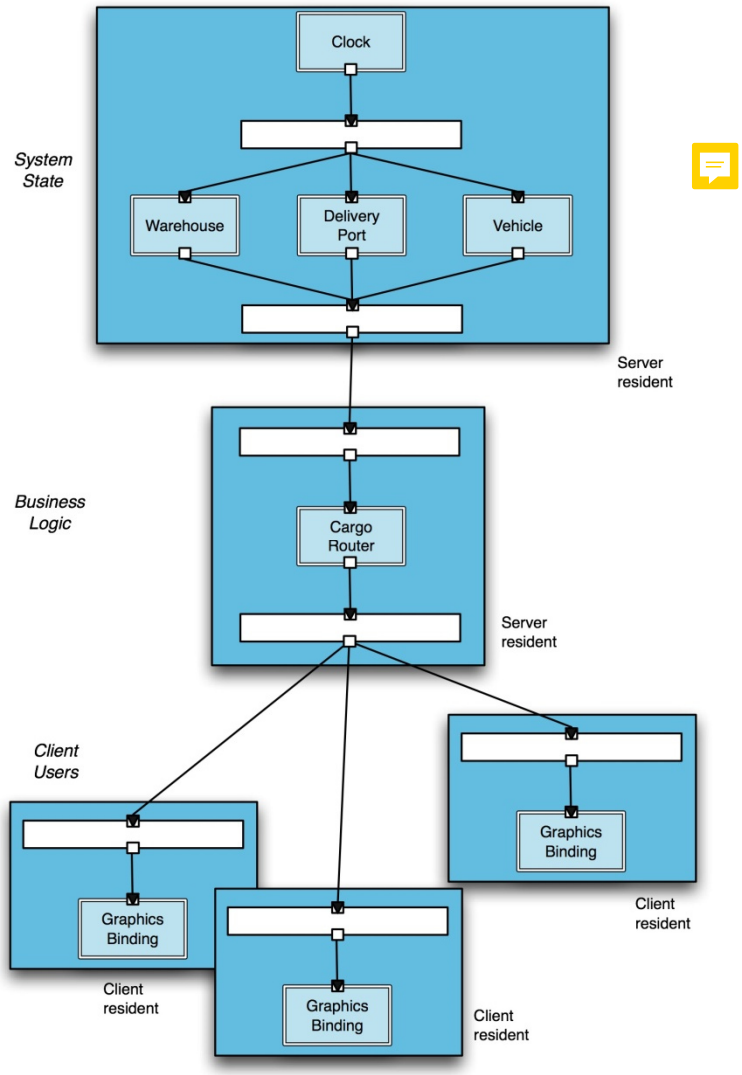
- Esempio: la prospettiva strutturale, di interazione, di comportamento, etc. …

- Cosa contiene la prospettiva di deployment ???

Deployment

- Un sistema software non può soddisfare lo scopo per cui è stato costruito fino a quando esso non viene dispiegato (*deployed)*
  - I moduli eseguibili vengono collocati sui dispositivi hadware su cui si prevede che funzioneranno
- La vista di deploy di una architettura può essere fondamentale anche per verificare se il sistema sarà in grado di soddisfare I suoi requisiti.
- Possibili dimensioni per la valutazione:
  - Memoria disponibile/ richiesta sui vari nodi
  - Consumo di potenza
  - Banda di rete richiesta

La Prospettiva Architetturale di Deployment



Elementi di una Architettura

Gli elementi fondamentali di una Architettura Software

- L'architettura di un sistema software non è monolitica
- Essa è un insieme di componenti interagenti di diverso tipo
- Gli elementi possono occuparsi di tre diversi tipi di aspetti:
  - **Processing,** ossia funzionalità o comportamento
  - **Dati**, ossia Informazioni o Stato
  - **Interazione,** ossia interconnessioni, comunicazione, coordinazione, etc...

Componenti

- I *Componenti Software* sono gli elementi che incapsulano **processing o dati** nell'ambito dell'architettura del sistema

- **Definizione**
  - A *software component* is an architectural entity that
    - encapsulates a subset of the system's functionality and/or data
    - restricts access to that subset via an explicitly defined interface
    - has explicitly defined dependencies on its required execution context
- Components typically provide *application-specific* services

Proprietà dei Componenti

- I Componenti software, esponendo solo l'interfaccia pubblica, sono scatole nere che realizzano i principi di ingegneria del software di Incapsulazione, Astrazione, e Modularità.
  - Ne consegue la loro componibilità, riusabilità ed evolvibilità.
- I Componenti trattano esplicitamente il contesto di esecuzione che essi assumono e da cui dipendono:
  - Interfaccia *Requires*; Risorse specifiche richieste (come file, directories,..);
  - Software di sistema richiesto (ambiente di runtime, middleware, sistema operativo, …)
  - Configurazione hardware richiesta

Connettori

- Nei sistemi complessi *l'interazione* può essere ancora più importante e impegnativa della funzionalità dei singoli componenti
- **Definizione**
  - Un connettore software è un elemento architetturale con il compito di effettuare e regolamentare le interazioni far i componenti.
- In molti sistemi software I connettori sono in genere semplici *procedure calls* o *accessi a dati condivisi*
  - Ma esistono connettori molto più complessi e sofisticati!
- I Connettori forniscono tipicamente servizi di interazione che sono *application-independent*

Esempi di Connettori

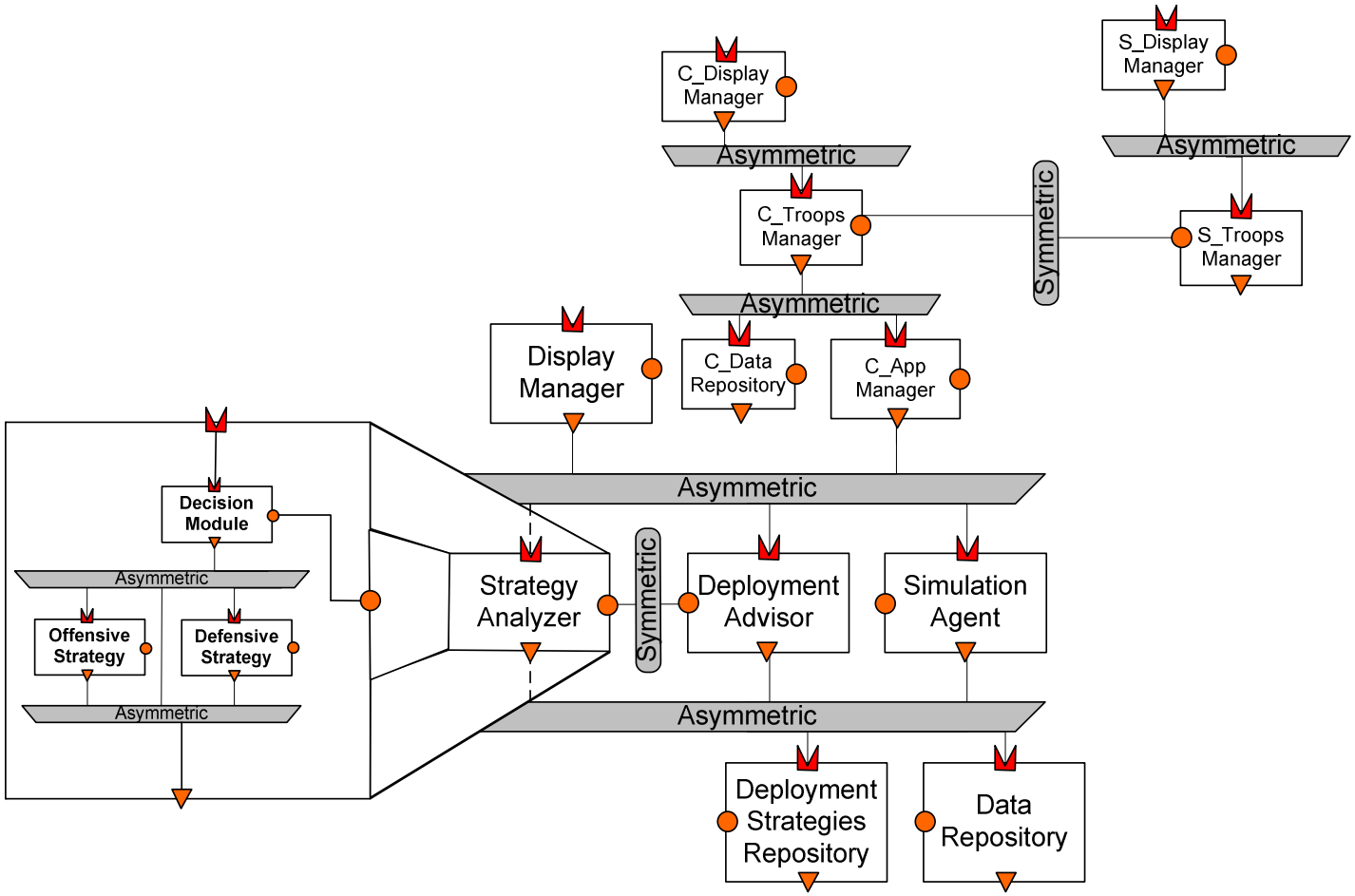
- Procedure call connectors
- Shared memory connectors
- Message passing connectors
- Streaming connectors
- Distribution connectors
  - Per la comunicazione in ambienti distribuiti (es. RPC)
- Wrapper/adaptor connectors
  - Nascono per integrare componenti preesistenti ed eterogenei (come **wrappers** e **glue code**)

Configurazioni

- Componenti e Connettori vengono composti in modo specifico in una data architettura di sistema, al fine di raggiungere l'obiettivo del sistema

- **Definizione**
  - Una *configurazione architetturale*, o topologia, è un insieme di associazioni specifiche fra componenti e connettori di una architettura software
  - Può essere rappresentata mediante un grafo

Una Configurazione di Esempio



Stili Architetturali 

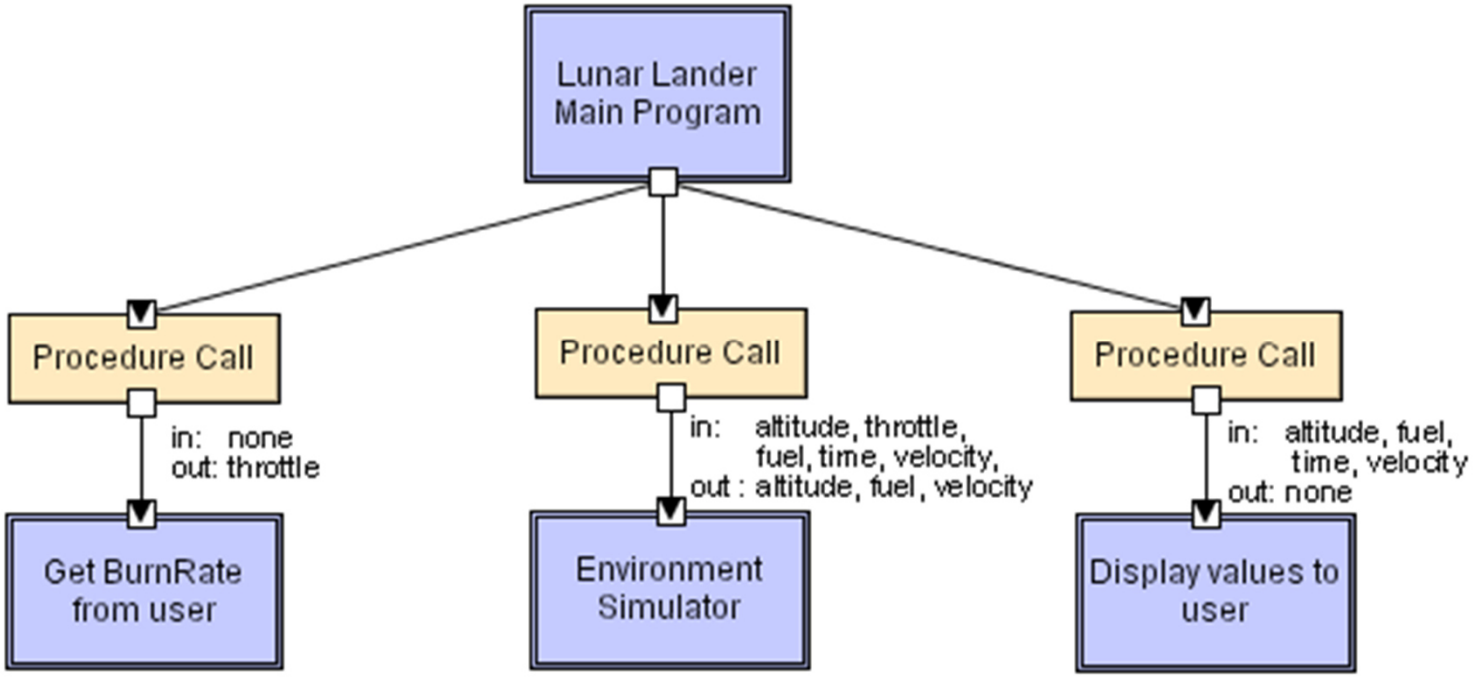
Stili Architetturali

- **Alcune scelte di progetto producono regolarmente soluzioni di qualità superiore**
  - Tali soluzioni sono più eleganti, efficaci, efficienti, affidabili, evolvibili, scalabili, etc.. di altre
- **Definizione**
  - Uno stile architetturale è una collezione di decisioni di progetto architetturali con un nome ben definito che:
    - Sono applicabili in un dato **contesto di sviluppo**
    - **Vincolano le decisioni** di progetto applicabili ad un particolare sistema di quel contesto
    - Portano **benefiche qualità** in ciascun sistema risultante

Alcuni Stili Architetturali…

- **Traditional, language-influenced styles**
  - Main program and subroutines
  - Object-oriented
- **Layered**
  - Virtual machines
  - Client-server
- **Data-flow styles**
  - Batch sequential
  - Pipe and filter
- **Shared memory**
  - Blackboard
  - Rule based
- **Interpreter**
  - Interpreter
  - Mobile code
- **Implicit invocation**
  - Event-based
  - Publish-subscribe
- **Peer-to-peer**

Esempio: lo stile Main Program e Subroutines



Pattern Architetturali

Pattern Architetturali

- **Definizione**
  - Un pattern architetturale è una collezione di decisioni di progetto architetturali con un nome ben definito che:
    - Sono applicabili ad un **problema di design** ricorrente, e sono parametrici per renderli applicabili in differenti contesti in cui il problema appare
  - **Esempio: Pattern** Three-Tiered usato nei moderni sistemi distribuiti
    - Usato in vari contesti: Science, Banking, E-commerce, Reservation systems

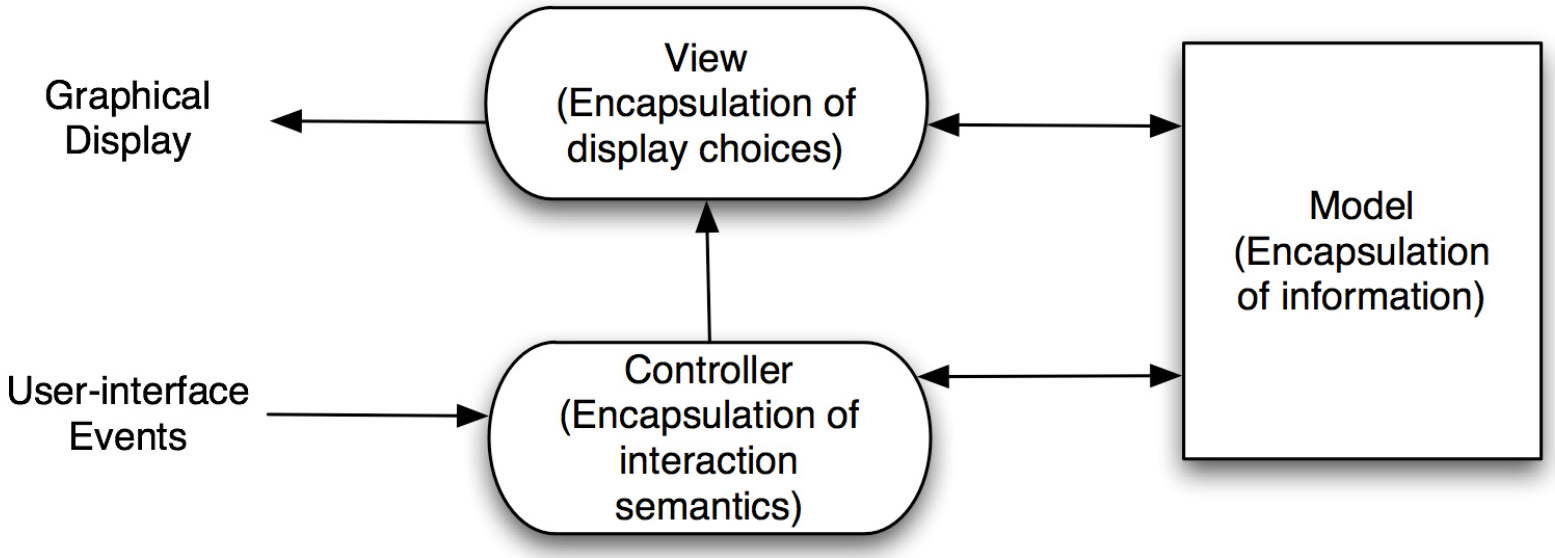
Three-Tiered Pattern

- Front Tier
  - Contains the user interface functionality to access the system's services
- Middle Tier
  - Contains the application's major functionality
- Back Tier
  - Contains the application's data access and storage functionality

Pattern Model-View-Controller (MVC)

- Obiettivo: Separare informazione, presentazione e interazione utente.
- Quando il valore di un oggetto del modello cambia, viene inviata una notifica sia alla view che al controller.
  - In tal modo la view si può aggiornare da sola
  - Il controller può modificare la view, se la sua logica lo richiede.
- Quando gestisce gli input utente, il sistema della UI manda l'evento utente al controller; se è richiesto un cambiamento dei dati, il controller aggiorna il model.

Model-View-Controller

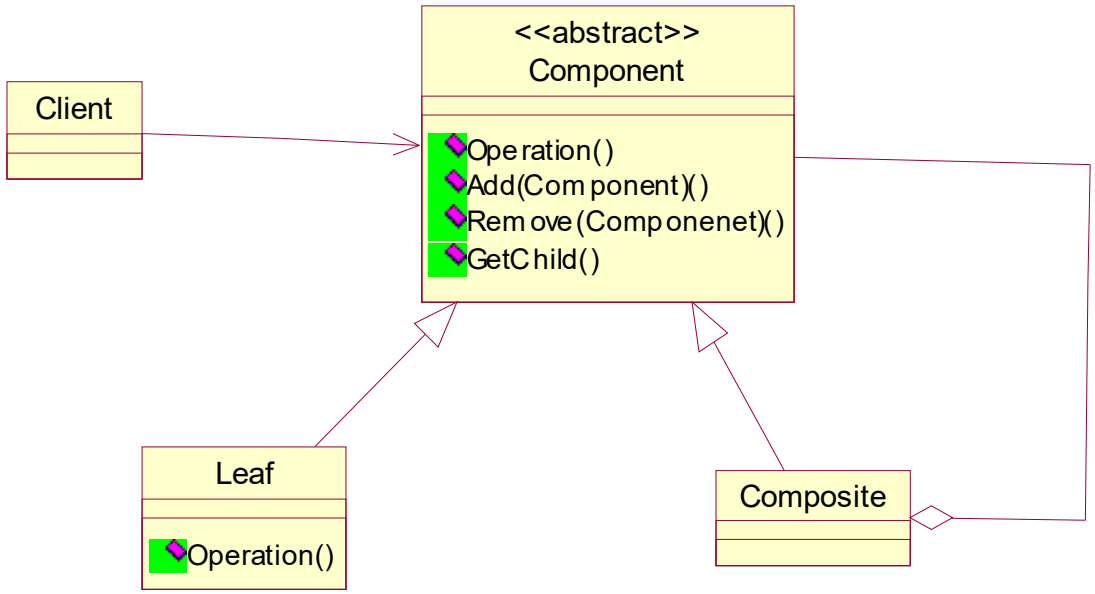


Design Patterns

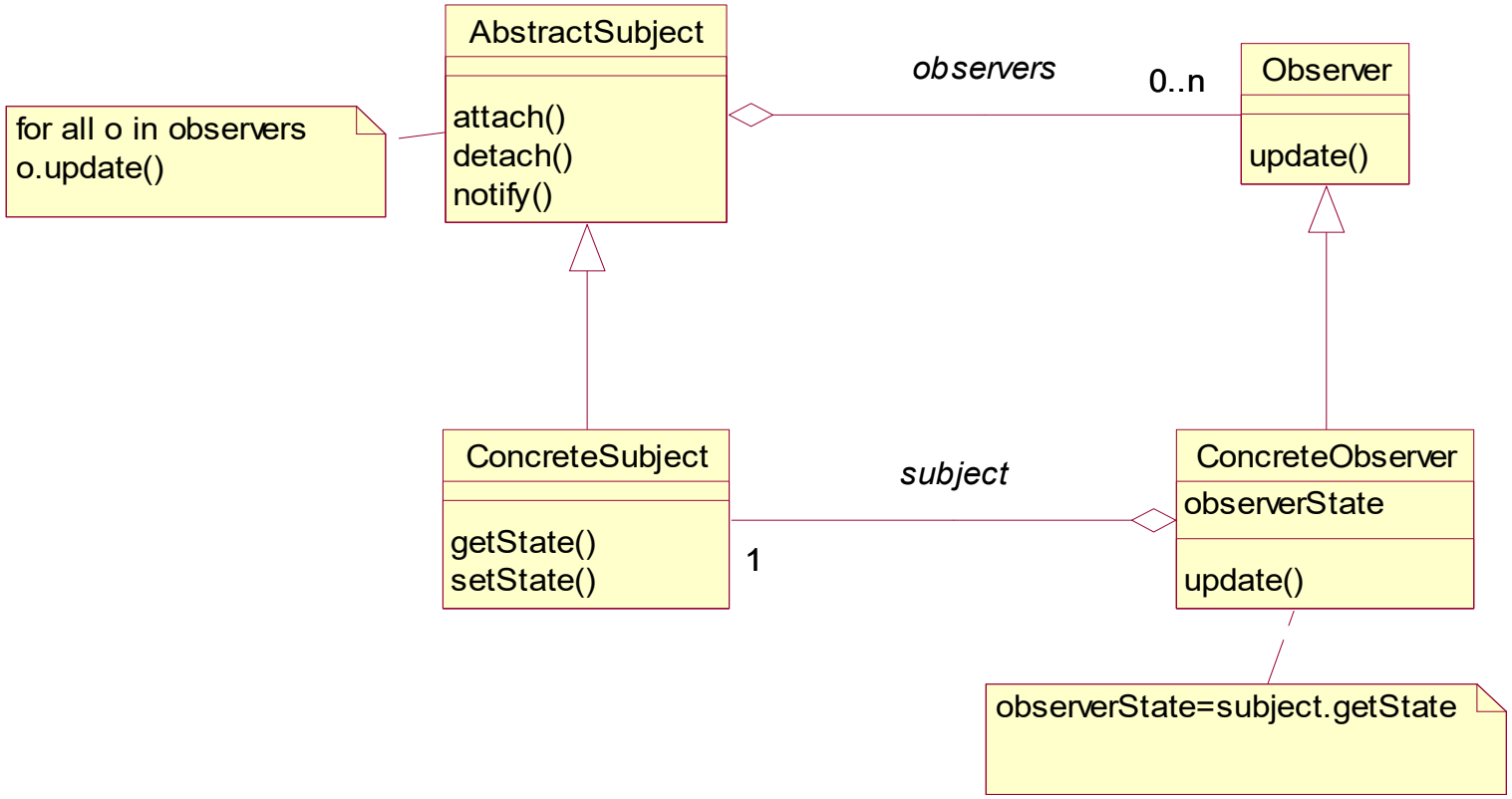
Design Patterns (Micro-architectural Styles)

- A pattern is "**a common solution to a common problem in a given context**."
- Mentre gli stili architetturali descrivono l'organizzazione di high-level del software (la macro-architecture), altri design patterns possono essere usati per descrivere dettagli ad un livello più locale.
  - **Pattern Creazionali** (es. builder, factory, prototype, and singleton)
  - **Pattern Strutturali** (adapter, bridge, composite, decorator, façade, flyweight, and proxy)
  - **P. di Comportamento** (command, interpreter, iterator, mediator, memento, observer, state, strategy, …)

Esempio: DP Composite



Esempio: DP Observer



Famiglie di programmi e Frameworks

- Un possibile approccio per permettere il riuso di progetti e componenti software consiste nel progettare famiglie di software, ossia "**software product lines**"
  - Si cercano le parti comuni fra membri di tali famiglie e si usano componenti riusabili ed estendibili per trattare le specificità interne alla famiglia.
    - Le linee di prodotto delle TV Sony / Boeing 747/…
- Nella programmazione OO si parla di **framework**: un sistema software parziale, che può essere opportunamente esteso
  - Es. Il *Framework Android* per lo sviluppo di app mobili

Modelli, Viste e Notazioni Architetturali

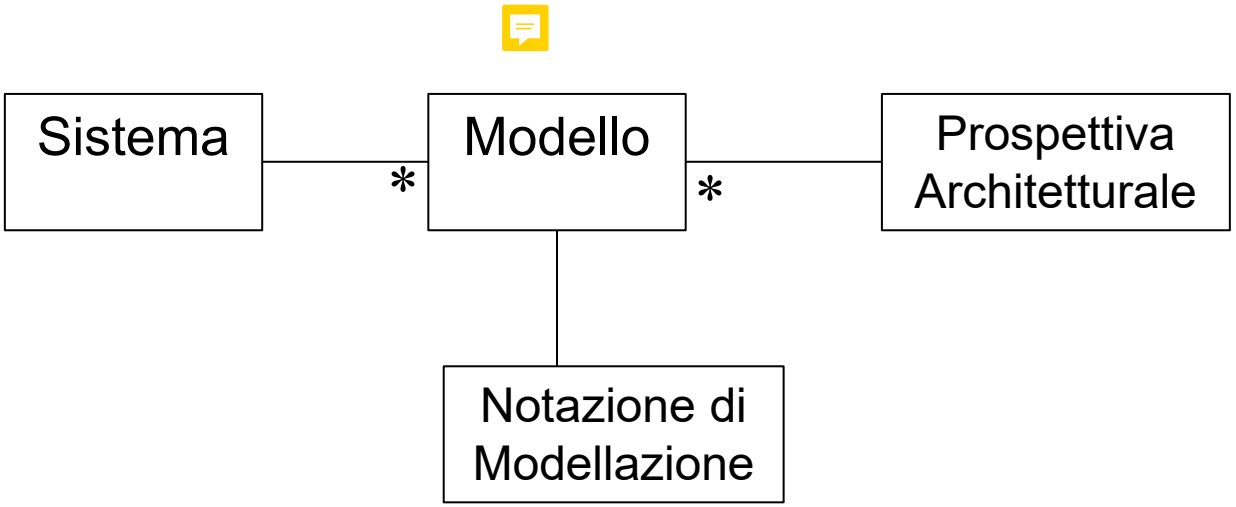
Modelli, Notazioni e Viste Architetturali

- Modello Architetturale
  - Un artifatto che documenta alcune o tutte le decisioni di progetto architetturale relative ad un sistema

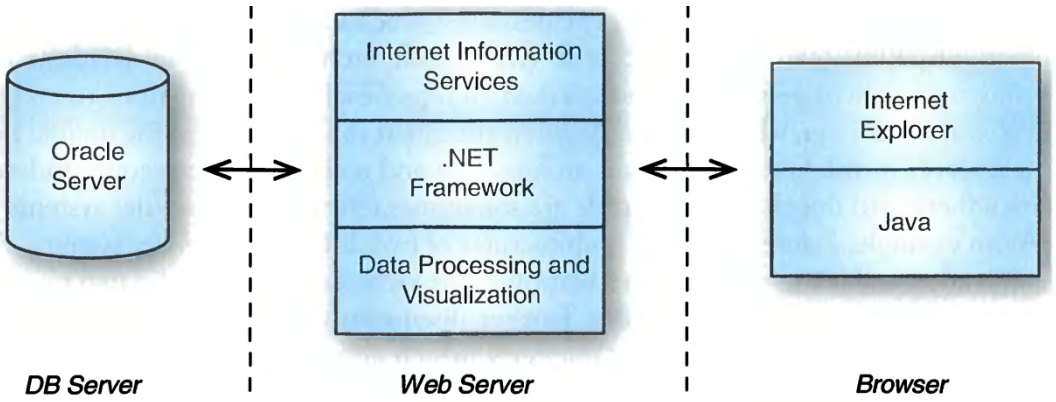
- Visualizzazione Architetturale (o Notazione)
  - Un modo per descrivere alcune o tutte le decisioni architetturali relative ad un sistema ad un determinato stakeholder

- Vista (o Prospettiva) Architetturale
  - Un sottoinsieme di decisioni di progetto architetturali correlate tra loro

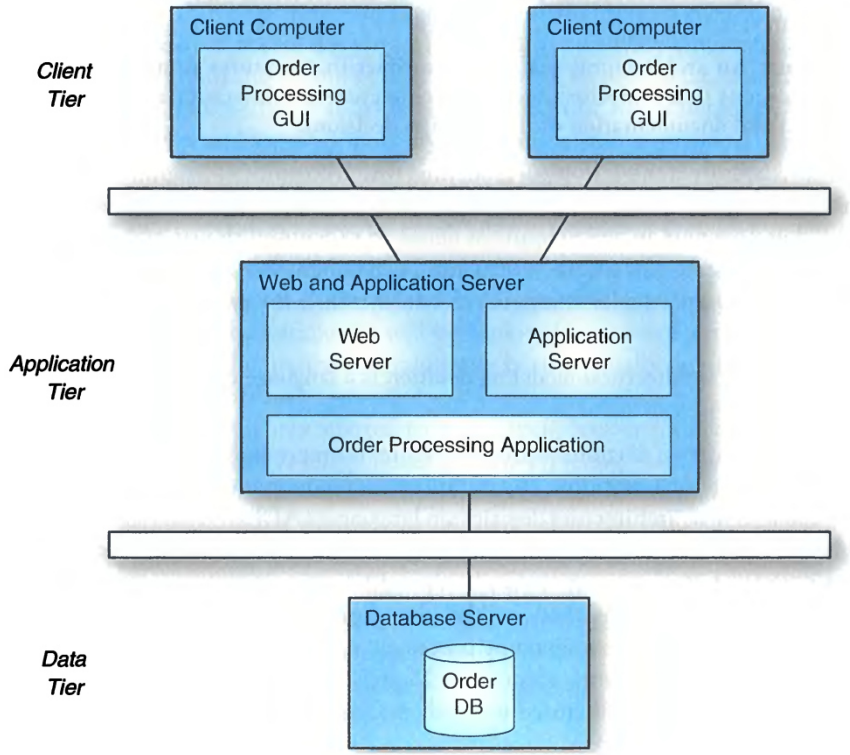
Relazioni fra Modelli, Prospettive, Notazioni



Esempio: due modelli architetturali con differenti notazioni



Modello (a)



Modello (b)

Notazioni per il Software Design

- La descrizione dei risultati di una attività di progettazione avviene usando una Notazione.

- Esistono molte notazioni per rappresentare gli artefatti del software design
  - Alcune modellano la struttura, altre il comportamento dell'architettura software
  - Alcune si usano in tutte le fasi di design, altre no
  - Alcune sono generiche, mentre altre sono specifiche di alcune metodologie di progettazione
  - …

Esempi: Notazioni per la Prospettiva Strutturale

- **Architecture description languages** (ADLs): textual, often formal, languages used to describe a software architecture in terms of **components and connectors**
- **Class and object** diagrams
- **Component diagrams**: rappresentano un insieme di componenti del sistema (“parti fisiche e sostituibili di un sistema che usano un dato insieme di interfacce e forniscono la realizzazione di un insieme di interfacce)
- **Deployment diagrams**: rappresentano la vista fisica del design, fatta di nodi fisici e loro relazioni
- **Entity-relationship diagrams** (ERDs): per rappresentare I modelli concettuali dei dati salvati nei sistemi informativi
- **Interface description languages** (IDLs)
- **Structure Charts**: describe the calling structure of programs

Esempi: Notazioni per la Prospettiva Comportamentale

- Activity diagrams
- Collaboration diagrams
- Data-Flow-Diagrams (DFD)
- Tabelle di decisione
- Flow-Charts
- Sequence diagrams
- State Transition diagram e statechart diagrams
- Linguaggi di specifica formale (spesso basati su pre e post-condizioni)
- PDL

Qualità delle Architetture Software

Qualità delle Architetture Software

- **Attributi di qualità di un buon design**:
  - maintainability, portability, testability, traceability
  - Correctness, robustness, “fitness of purpose”

- Una possibile distinzione è tra:
  - Attributi di qualità **osservabili a run-time** (performance, security, availability, functionality, usability),
  - Attributi **non osservabili a run-time** (modifiability, portability, reusability, integrability, and testability),
  - E quelli relativi a **qualità intrinseche** dell’architettura (conceptual integrity, correctness, and completeness, buildability)

Tecniche di analisi e valutazione della qualità del design

- **Software design reviews**:
  - Informali o semi-formali, spesso fatte in gruppo, tecniche per verificare ed assicurare la qualità di un artefatto software
- **Analisi Statica**
  - Analisi statica formale o semiformale (es. Analisi statica in compilazione, data-flow analysis, analisi simbolica…)
- **Simulation and prototyping**:
  - Tecniche di analisi dinamica per valutare altri aspetti di un design (ad esempio, simulazione delle prestazioni o prototipo per la fattibilità).

Misure sul design

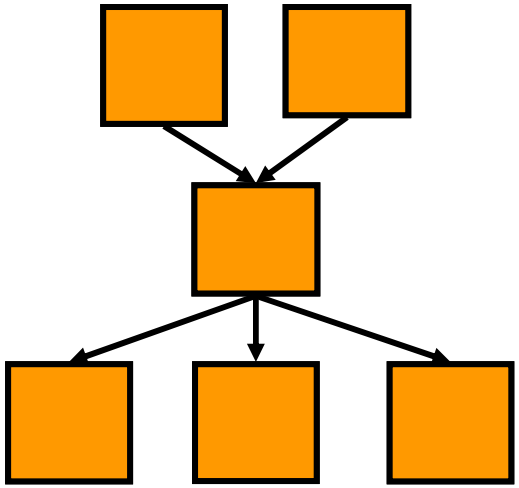
- Le Misure possono essere usate *per valutare* oppure *stimare* quantitativamente vari aspetti di un design, quali la dimensione, la struttura o la qualità.

- Molte misure dipendono dall'approccio di progettazione e sviluppo adottato

- **Es.: Misure per il design Function-oriented**
  - Si eseguono su un modello rappresentante la decomposizione gerarchica dei moduli
  - Fan-in, Fan-out, Numero di Moduli, Coesione ed Accoppiamento…

Fan-in e Fan-out

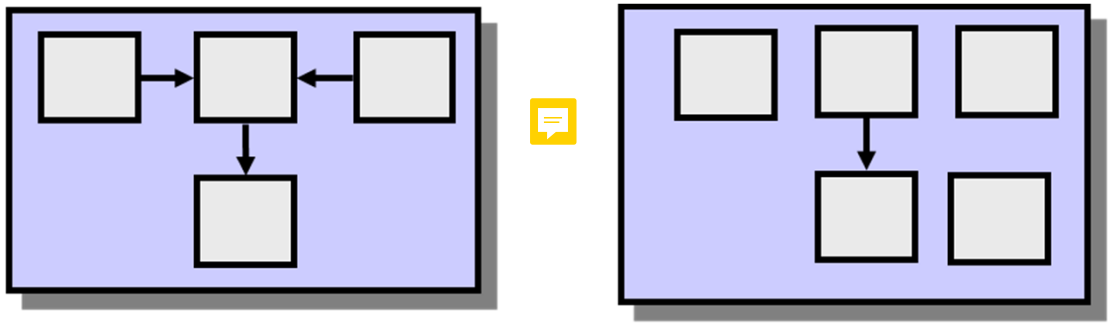
Fan-in = number of ingoing dependencies
Fan-out = number of outgoing dependencies



Heuristic: a high fan-in/fan-out indicates a high complexity

Coesione

Cohesion is concerned with the interactions within a module



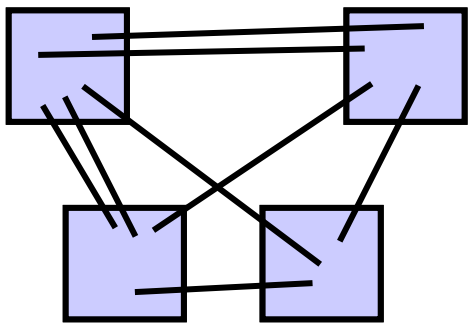
high cohesion

low cohesion

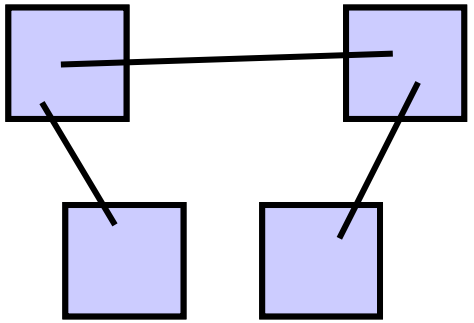
Heuristic: Keep things together that belong together.
High cohesion within a module is good

Accoppiamento

Coupling is the degree of interdependence between modules



high coupling



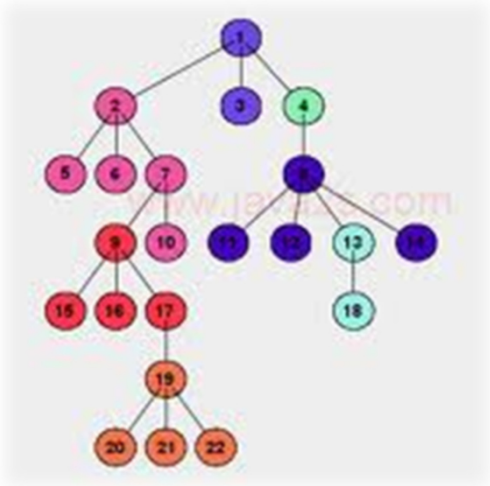
low coupling

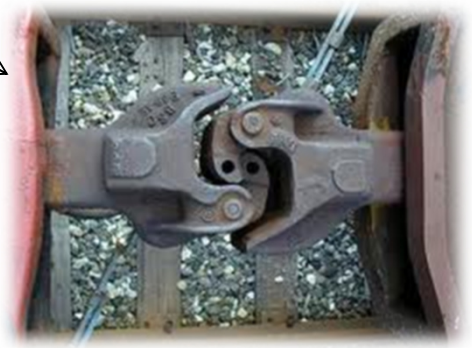
Heuristic: minimize coupling between modules

Misure per il Design OO

- Metriche di **Chidamber** & **Kemerer**

- WMC: Weighted Methods per Class
- DIT: Depth of Inheritance Tree
- NOC: Number Of Children
- CBO: Coupling Between Object Classes
- RFC: Response For a Class
- LCOM: Lack of COhesion of a Method





Strategie e Metodi per la Progettazione

Strategie e Metodi per la Progettazione Software

- Strategie e Metodi ci guidano durante la progettazione.
- Le **Strategie di Progettazione** forniscono un approccio generale alla progettazione del software, come ad esempio:

- Divide-and-conquer and stepwise refinement
- top-down vs. bottom-up strategies
- data abstraction and information hiding
- use of patterns and pattern languages
- use of an iterative and incremental approach.

Metodi di Progettazione

- Sono più specifici delle strategie:
  - Forniscono una **notazione** da usare
  - Una descrizione del **processo** da seguire
  - Un insieme di **lineeguida** per applicare il metodo
- Esempi:
  - Function-Oriented (Structured) Design (basato sui DFD)
  - Object-Oriented Design
  - Component-Based Design
  - Service-Oriented-Design
  - Model-Based-Design …

Processi Architetturali

- *L'architettura software è centrale in diversi processi del CVS:*
- Architectural design
- Architecture modeling and visualization
- Architecture-driven system analysis
- Architecture-driven system implementation
- Architecture-driven system deployment, runtime redeployment, and mobility
- Architecture-based design for non-functional properties, including security and trust
- Architectural adaptation

Stakeholders in a System's Architecture

- Architects
- Developers
- Testers
- Managers
- Customers
- Users
- Vendors