Progettazione di un sistema

- La progettazione di un sistema (**Software Design**) è l'insieme dei task svolti dall'ingegnere del software per trasformare il modello di analisi nel modello di design del sistema
- Obiettivo ultimo di System e Object Design è definire un nuovo documento, scritto in linguaggio tecnico/formale, da dare ai programmatori affinchè questi siano in grado di implementare il software descritto nel Documento dei Requisiti Software

Scopo del System Design

1. Definire gli obiettivi di design del progetto
2. Definire l'**Architettura del sistema**, decomponendolo in sottosistemi più piccoli
   - Possono essere realizzati in parallelo da team individuali
3. Selezionare le strategie per costruire il sistema, quali:
   - Strategie hardware/software
   - Strategie relative alla gestione dei dati persistenti
   - Il flusso di controllo globale
   - Le politiche di controllo degli accessi
   - ...

Scopo dell’Object Design

- Data l’Architettura del Sistema, specificare il dettaglio realizzativo dei sottosistemi più piccoli
- Selezionare le strategie ottimali per l’implementazione
  - Design Pattern

Il System Design

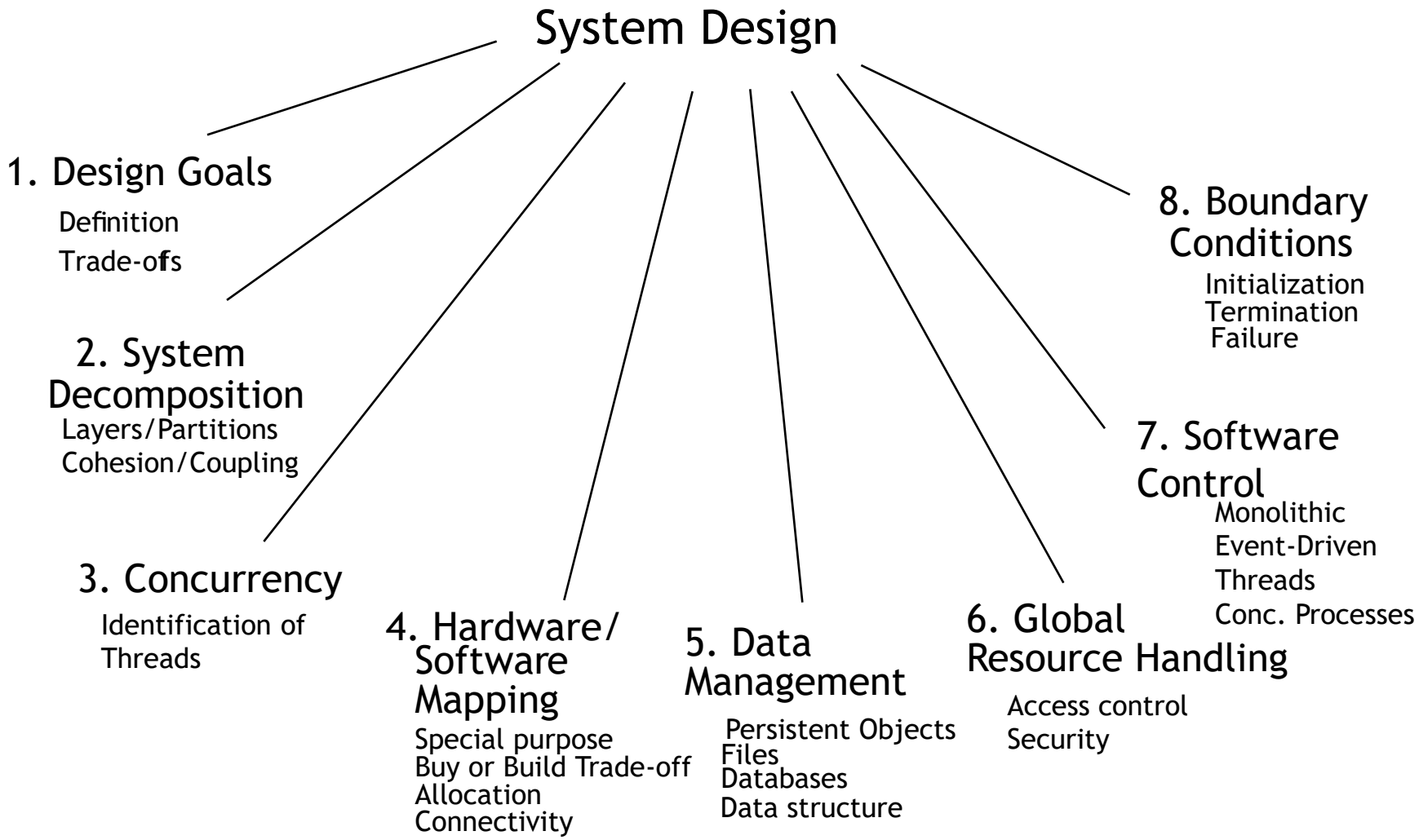
System Design

- Cambia radicalmente punto di vista:
  - Finora abbiamo lavorato per interagire col cliente
    - Linguaggio per profani, poco formale
  - Nel system design i nostri interlocutori sono le squadre di programmatori che dovranno implementare il sistema
    - Chi leggerà i nostri documenti sarà un esperto di informatica
    - Linguaggio per esperti tecnici

Attività di system design

- Il system design è costituito da tre macro-attività:
  1. Identificare gli obiettivi di design:
     - gli sviluppatori identificano quali caratteristiche di qualità dovrebbero essere ottimizzate e definiscono le priorità di tali caratteristiche
  2. Progettazione della decomposizione del sistema in sottosistemi:
     - basandosi sugli use case ed i modelli di analisi, gli sviluppatori decompongono il sistema in parti più piccole. Utilizzano stili architetturali standard.
  3. Raffinare la decomposizione in sottosistemi per rispettare gli obiettivi di design:
     - La decomposizione iniziale di solito non soddisfa gli obiettivi di design. Gli sviluppatori la raffinano finché gli obiettivi non sono soddisfatti.

Passi del System Design



Identificare gli obiettivi qualitativi del sistema

- E’ il primo passo del system design
- Identifica le **qualità** su cui deve essere focalizzato il sistema
- Molti design goal possono essere ricavati dai requisiti non funzionali o dal dominio di applicazione, altri sono forniti dal cliente
- E’ importante formalizzarli esplicitamente poiché ogni importante decisione di design deve essere fatta seguendo lo stesso insieme di criteri

Criteri di design

- Possiamo selezionare gli obiettivi di design da una lunga lista di qualità desiderabili.
- I criteri sono organizzati in cinque gruppi:
  - Performance
    - Tempo di risposta, Throughput, Requisiti di Memoria, ...
  - Affidabilità
    - Robustezza, Disponibilità, Tolleranza ai fault, Sicurezza, ...
  - Costi
    - Sviluppo, Manutenzione, Gestione, ...
  - Mantenimento
    - Estendibilità, Modificabilità, Adattabilità, Portabilità, ...
  - Usabilità

Design Trade-offs

- Quando definiamo gli obiettivi di design, spesso solo un piccolo sottoinsieme di questi criteri può essere tenuto in considerazione.
  - Es: non è realistico sviluppare software che sia simultaneamente sicuro e costi poco.
- Gli sviluppatori devono dare delle priorità agli obiettivi di design, tenendo anche conto di aspetti manageriali, quali il rispetto dello schedule e del budget.

Design Trade-offs (cont.)

- Esempi:
  - Spazio vs. velocità. Se il software non rispetta i requisiti di tempo di risposta e di throughput, è necessario utilizzare più memoria per velocizzare il sistema (es. Caching, più ridondanza). Se il software non rispetta i requisiti di memoria, può essere compresso a discapito della velocità.
  - Tempo di rilascio vs. funzionalità. Se i tempi di rilascio sono stringenti, possono essere rilasciate meno funzionalità di quelle richieste, ma nei tempi giusti.
  - Tempo di rilascio vs. qualità. Se i tempi di rilascio sono stretti, il project manager può rilasciare il software nei tempi prefissati con dei bug e, in tempi successivi, correggerli, o rilasciare il software in ritardo, ma con meno bug.
  - Tempo di rilascio vs. staffing. Può essere necessario aggiungere delle risorse al progetto per accrescere la produttività.

Esempio iniziale

Abbozzare gli otto passi del system design per un'applicazione reale

Concetti di base di buon design

Qualità del codice

Concetti di System Design

- Molti criteri di Design sono strettamente dipendenti dal problema trattato
  - La scelta del miglior bilanciamento tra i vincoli posti è un compito dell'analista, che opera in base alla propria esperienza/sensibilità
- Esistono però dei Criteri di Design generici, che valgono per qualunque software O-O
  - Questi criteri, se ben applicati, migliorano notevolmente la qualità interna del codice, senza introdurre particolari svantaggi

Pensare in avanti

- Quando progettiamo una classe dovremmo sforzarci di pensare a quali cambiamenti potranno essere richiesti in futuro:

  “**Progettare per ANTICIPARE IL CAMBIAMENTO**”

- Le nostre scelte iniziali devono facilitare l’evoluzione futura
  - E’ una delle richieste fondamentali, nonché fonte di valutazione per il progetto!

La Decomposizione

- Per ridurre la complessità della soluzione, decomponiamo il sistema in parti più piccole, chiamate sottosistemi.
  - Un sottosistema tipicamente corrisponde alla parte di lavoro che può essere svolta autonomamente da un singolo sviluppatore o da un team di sviluppatori.
- Decomponendo il sistema in sottosistemi relativamente indipendenti, i team di progetto possono lavorare sui sottosistemi individuali con un minimo overhead di comunicazione.
- Nel caso di sottosistemi complessi, applichiamo ulteriormente questo principio e li decomponiamo in sottosistemi più semplici.

La Decomposizione

- P= problema,
- C=complessità di P,
- E=sforzo per la risoluzione di P
- Dati 2 problemi P1 e P2, se C(P1) > C(P2), allora E(P1)> E(P2).
- Ma è stato dimostrato che C(P1+P2)> C(P1) + C(P2), e quindi si ha che E(P1+P2)> E(P1) + E(P2)

Modellazione di sottosistemi

- Subsystem (UML: Package)
  - Collezioni di classi, associazioni, operazioni e vincoli che sono correlati

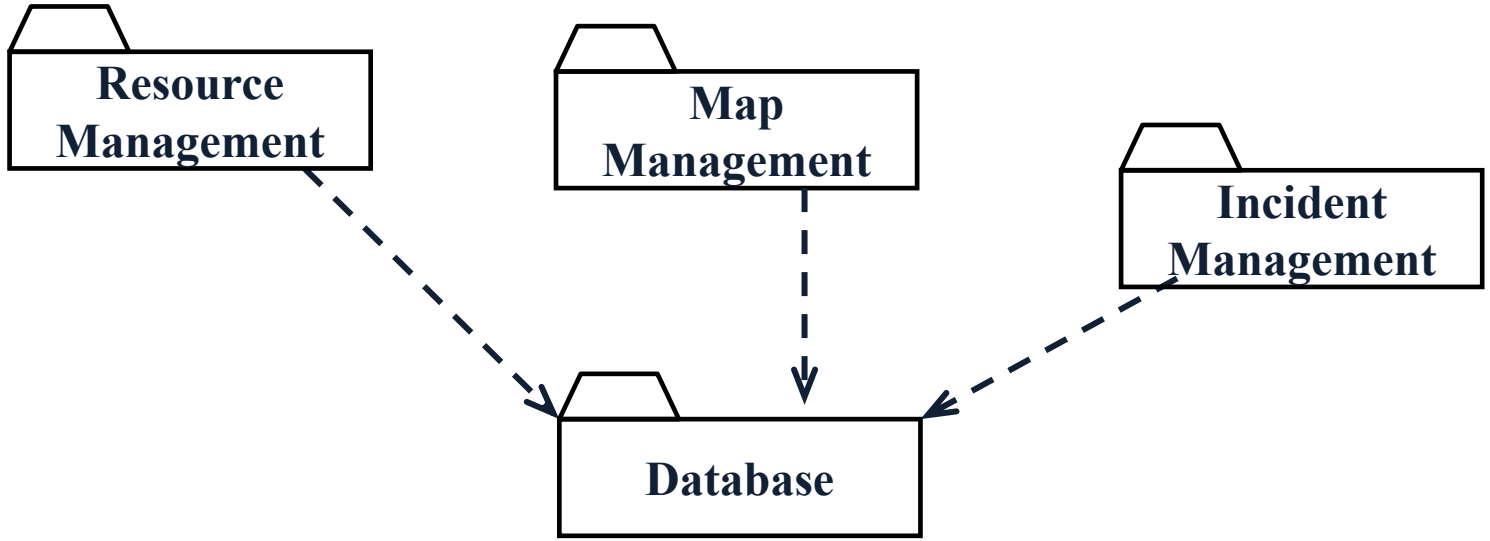
- JAVA fornisce i package che sono costrutti per modellare i sottosistemi
  - C++ / C# hanno il costrutto di namespace per indicare lo stesso concetto

Servizi e interfacce di sottosistemi

- Un sottosistema è caratterizzato dai servizi che fornisce agli altri sottosistemi
- L'insieme di operazioni di un sottosistema che sono disponibili agli altri sottosistemi forma l'interfaccia del sottosistema
  - Include il nome delle operazioni, i loro parametri, il loro tipo ed i loro valori di ritorno.
- Il system design si focalizza sulla definizione dei servizi forniti da ogni sottosistema in termini di:
  - Operazioni
  - Loro parametri
  - Loro comportamento ad alto livello

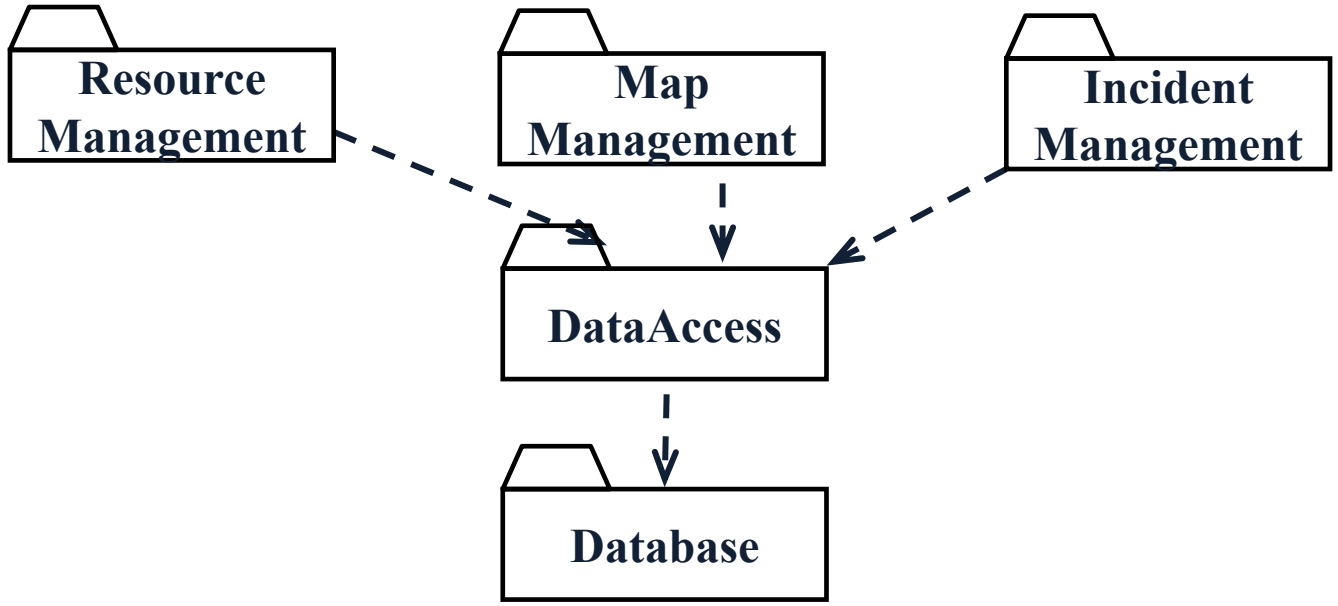
Esempio di scelta di design: accesso diretto al database

- Non riesco a specificare chiaramente le responsabilità di ogni modulo (Gestisce, Salva?)
- Tutti i sottosistemi accedono al database direttamente, rendendoli vulnerabili ai cambiamenti dell'interfaccia del sottosistema Database
  - Se cambia il DBMS, devo modificare il codice in tutte le classi che accedono al DB!



Esempio: accesso al database attraverso un sottosistema di Storage

- Chiariamo le responsabilità:
- Introduco il sottosistema DataAccess che gestisce l'I/O da db
  - Se cambia il DBMS, devo modificare solo il sottosistema Storage!



Osservazioni

- Nell’esempio proposto abbiamo ridotto le relazioni fra i quattro sottosistemi, ma …
- Abbiamo aumentato la complessità!
- Con l’obiettivo di ridurre le relazioni si rischia di aggiungere dei livelli di astrazione che consumano tempo di sviluppo e tempo di elaborazione.

Indipendenza Funzionale

- La suddivisione in sottosistemi può essere guidata da due fattori qualitativi fondamentali:
  - Coesione (cohesion)
  - Accoppiamento (coupling)

- L’obiettivo è l’indipendenza funzionale dei sottosistemi, che si massimizza con Alta Coesione e Basso Accoppiamento