Ingegneria del Software
12. Progettazione

Dipartimento di Informatica
Università di Pisa
A.A. 2014/15

progettare prima di produrre

- Tipico della produzione industriale
  - “Sul tavolo da disegno si usa la gomma, sul cantiere la mazza” [Frank Lloyd Wright]
  - La frase di Wright si applica al software, nonostante la apparente omogeneità dei materiali tra progetto e realizzazione: il codice è più “duro” dei modelli!
- Alcune motivazioni
  - Complessità del prodotto
  - Organizzazione e riduzione delle responsabilità
  - Controllo preventivo della qualità

obiettivi della progettazione: produttore

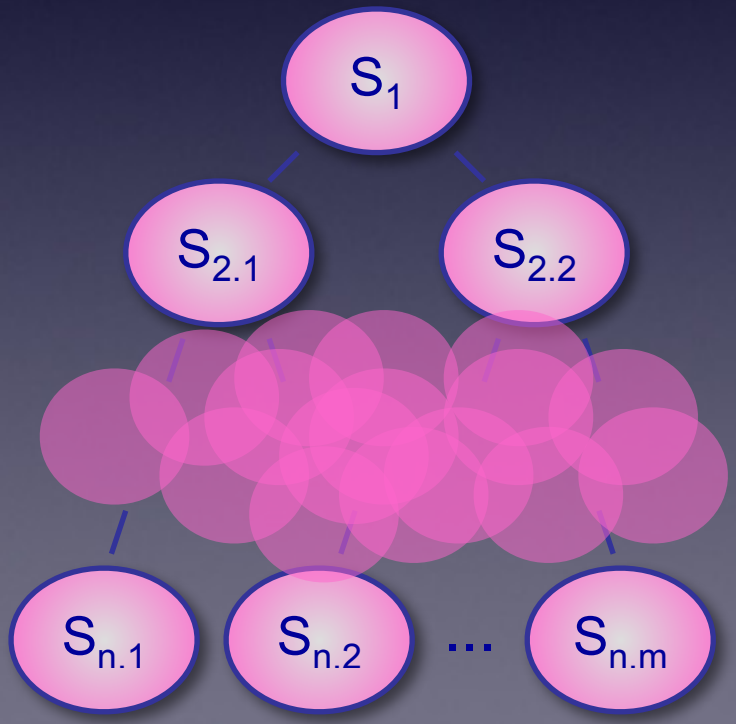
- Trasformare la complessità globale del sistema
  - Ridurre le difficoltà di comprensione e realizzazione
  - Organizzare il sistema in sottosistemi di minore complessità
  - Identificare i sottosistemi riusabili
- Soddisfare i requisiti di qualità del produttore
  - Il modello progettuale deve essere una guida leggibile e comprensibile per chi genera il codice, per chi lo verifica e per chi dovrà mantenerlo e fornire supporto agli utenti, una volta che il software sarà operativo
  - Il modello progettuale deve fornire un'immagine completa del software, che copra il dominio dei dati, delle funzioni e del comportamento da un punto di vista realizzativo

altri obiettivi

- Produrre la documentazione necessaria
  - Per far procedere la codifica senza ulteriori informazioni
  - Per tracciare i requisiti nelle unità
  - Per definire le configurazioni del sistema
- Associare componenti fisici alle unità
  - Per organizzare il lavoro di codifica
- Definire gli strumenti per le prove delle unità
  - Casi di prova e componenti fittizi per le prove e l'integrazione

metodo

- Progettaz. di alto livello
- Progettaz. di dettaglio



- Top-down
  - decomposizione di problemi
- Bottom-up
  - composizione di soluzioni
- Sandwich
  - approccio “naturale”

progettazione di alto livello (architetturale)

- L’architettura di un sistema software (in breve architettura software) è la struttura del sistema, costituita dalle parti del sistema, dalle relazioni tra le parti e dalle loro proprietà visibili

[check dispensa]

progettazione di dettaglio

- Definizione di unità di realizzazione (sottosistema terminale)
  - Un sistema che non deve essere ulteriormente trasformato
  - Quando un'altra trasformazione avrebbe un costo ingiustificato o porterebbe a un'inutile esposizione di dettagli
  - Un carico di lavoro assegnabile a una sola persona
  - Un sottosistema definito (ad esempio, un componente)
  - Un insieme di funzionalità affini (ad esempio, un package)
- Descrizione delle unità di realizzazione
  - Da zero o per specializzazione di sistemi esistenti
  - Definizione delle caratteristiche "interessanti"
    - stato dei/interazione tra sistemi

principi di progettazione

- Astrazione
  - dati
  - controllo
    - corsi precedenti
- Information Hiding
  - controllo delle interfacce
- Raffinamento
  - elaborazione dei dettagli di ogni astrazione
- Modularità
  - compartimentalizzazione dati e funzioni
    - Coesione e accoppiamento

moduli come astrazione sul controllo (librerie)

- Nei linguaggi di programmazione tradizionali per modulo si intende una prima forma di ***astrazione*** effettuata sul ***flusso di controllo*** e il concetto di modulo è identificato con il concetto di ***subroutine*** o ***procedura***

- Una procedura può effettivamente nascondere una scelta di progetto riguardante l'algoritmo utilizzato. ***Ad esempio, per l'ordinamento di un vettore di elementi interi si hanno più algoritmi diversi per tale compito: bubblesort, shellsort, quicksort, etc.***

- Le procedure in quanto astrazioni sul controllo sono utilizzate come parti di alcune classi di moduli, che prendono il nome di ***librerie*** (ad esempio, librerie di funzioni matematiche e grafiche)

moduli come astrazione di dato

- Un'astrazione di dato (o Struttura Dati Astratta, ADS) è un modo di *incapsulare* un *dato* in una rappresentazione tale da regolamentarne l'accesso e la modifica
- *Interfaccia* dell'oggetto (operazioni) *stabile* anche in presenza di modifiche alla struttura dati
- Risultati dell'attivazione di un'operazione su un oggetto dipendenti anche dal valore del dato (*stato*)
- Non puramente funzionali (come le *librerie*)

information hiding

- Componenti come *scatole nere*
  - Fornitori e clienti di funzionalità (relazione d'uso)
  - È nota solo l'interfaccia (dichiarazione dei servizi)
- Sono mantenuti nascosti
  - Algoritmi usati
  - Strutture dati interne
- Vantaggi di un'alto IH
  - Nessuna assunzione sul funzionamento della componente
  - Manutenibilità, riusabilità

modularità

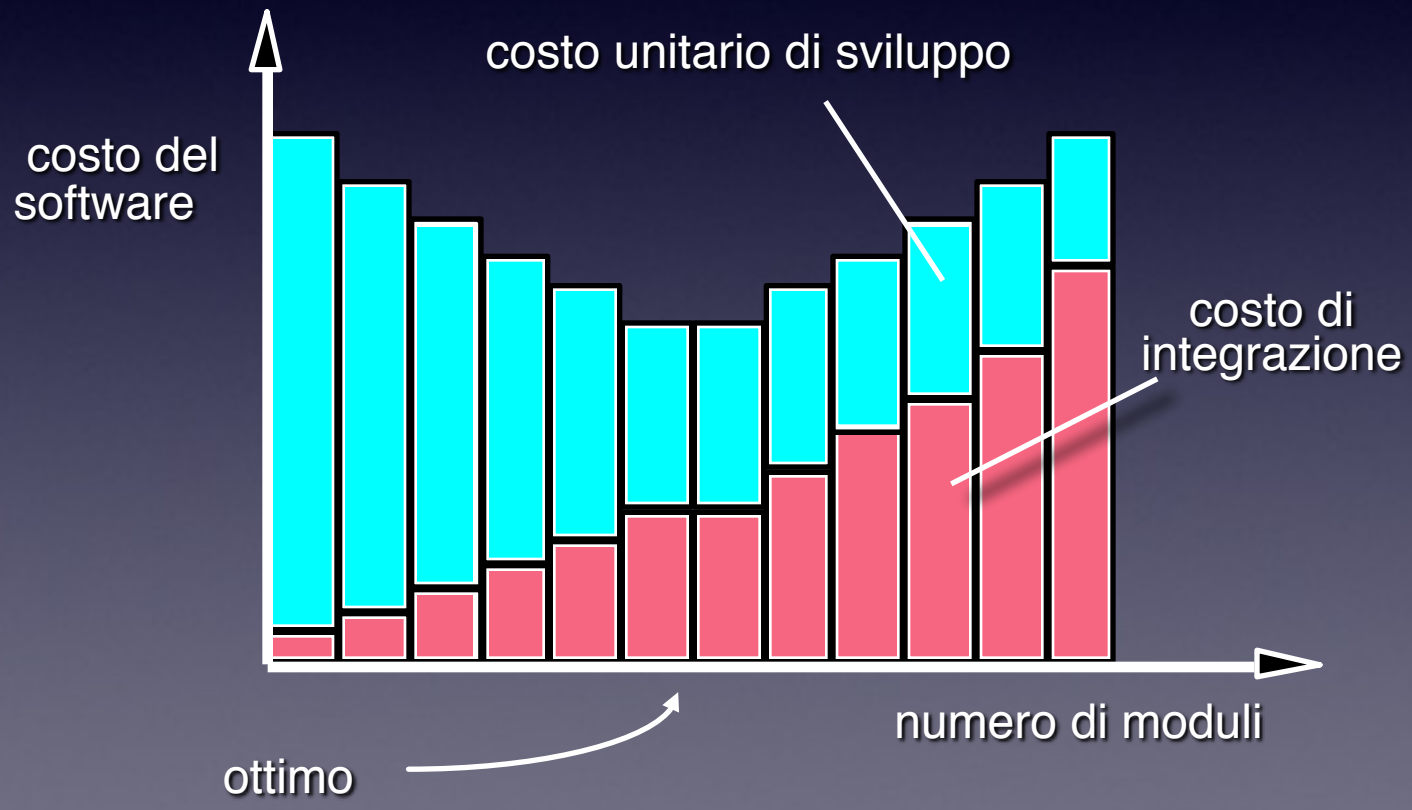
- Coesione: proprietà di un (sotto)sistema
  - grado in cui un sistema realizza “uno e un solo concetto”
  - funzionalità “vicine” devono stare nello stesso sistema
  - vicinanza per tipo, algoritmi, dati in ingresso e in uscita
- Accoppiamento: proprietà di un insieme di (sotto)sistemi
  - grado in cui un sistema è “legato” ad altri sistemi

modularità

- Si hanno vantaggi con (sotto)sistemi che esibiscono
  - un alto grado di coesione
  - un basso accoppiamento
- Maggior riuso e migliore mantenibilità
- Ridotta interazione fra (sotto)sistemi
- Miglior comprensione del sistema

modularità: compromessi

- Quale è il giusto numero di moduli?



dall'uso alla (ri)progettazione

- Reingegnerizzazione
  - In seguito a successivi e massicci interventi di manutenzione
  - In seguito a un cambio di piattaforma o di tecnologia
- Riprogettazione
  - Quando lo sviluppo parte dalla progettazione
  - Quando i requisiti sono definiti e sostanzialmente stabili
  - In caso di interventi, anche radicali, sia sulla struttura del sistema sia sul codice
  - Sempre, per reing. e riuso

refactoring (ristrutturazione)

- Refactoring è “il processo di cambiare un sistema software senza alterarne il comportamento esterno ma migliorandone la struttura interna” [Fowler]

- Si cercano
  - ridondanze
  - elementi inutilizzati
  - algoritmi inefficienti
  - strutture dati inappropriate o “deboli”