Progettazione Architettura

- L'architettura software di un sistema è un artefatto, frutto della attività di progettazione
- Una architettura software è descritta dai suoi componenti (e sottosistemi) e dalle relazioni tra essi
  - ***Componenti***, costituiscono i 'building block' di un sistema (es: moduli, classi, funzioni)
  - ***Relazioni***, denotano una connessione tra componenti: aggregazione, eredità, interazione
- Un sottosistema è un sistema le cui operazioni sono indipendenti dai servizi di altri sottosistemi
- Un componente fornisce servizi ad altri componenti e non è considerato come un sistema a sé stante

Viste architetturali

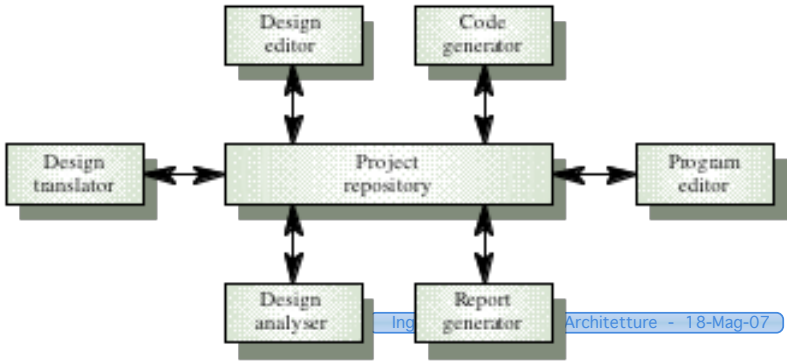
- Vari punti di vista possono produrre varie architetture che rappresentano prospettive diverse del sistema
  - Mostrare i componenti del sistema
  - Definire le interfacce tra sottosistemi

Stili architetturali

- L'architettura può essere conforme ad un modello o *stile*
  - Termini intercambiabili: architetture di riferimento, architectural pattern e stili architetturali
- *La conoscenza degli stili può semplificare la definizione dell'architettura per un sistema*
- I sistemi più grandi sono eterogenei e non seguono un singolo stile
- Caratteristiche importanti che una architettura dovrebbe esibire
  - Modularità
  - Indipendenza funzionale
  - Information hiding

Modello Repository (Blackboard) [1980]

- I sottosistemi necessitano di scambiare dati tra loro
- I dati sono tenuti in un database centrale o repository accessibile da tutti i sottosistemi
- Adatto per sistemi che condividono grandi quantità di dati

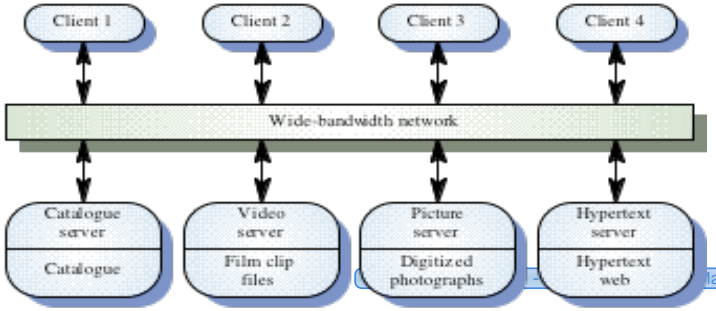


Modello Repository

- Vantaggi
  - Modo efficiente di condividere grandi quantità di dati, senza doverli duplicare in ciascun sottosistema
  - I sottosistemi non necessitano di sapere come i dati sono prodotti
  - Backup, controllo degli accessi, recovery sono centralizzati
  - La struttura del repository è pubblicata, nuovi sottosistemi possono essere inseriti senza problemi
- Svantaggi
  - I sottosistemi devono scegliere una struttura del repository (è un compromesso tra necessità diverse dei sottosistemi)
  - Cambiamenti sul formato dei dati possono essere difficili (coinvolgono tutti i sottosistemi)
  - Difficile distribuire i dati efficientemente
  - Possibili problemi di inconsistenza dei dati

Modello Client-Server

- Ambiente distribuito
- Set di server che forniscono servizi specifici ad altri sottosistemi
  - Stampa, gestione dati, etc.
- Set di client che chiamano i server
- La rete permette ai client di accedere ai server

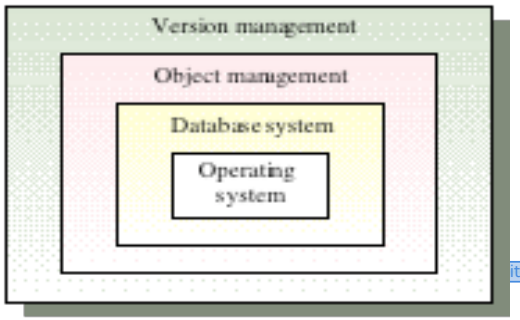


Client-server

- Vantaggi
  - Usa efficacemente sistemi distribuiti, ovvero tanti elaboratori, ciascuno dedicato ad un servizio
    - Può richiedere hardware meno costoso rispetto ad un sistema centralizzato
  - Aggiungere nuovi server o aggiornare un server è facile, non coinvolge altri sottosistemi
- Svantaggi
  - Ogni client gestisce i propri dati e li trasferisce ai server per le elaborazioni, tale scambio di dati può essere inefficiente
  - Management dati (per backup, sicurezza) ridondante su ciascun server
  - Bisogna far conoscere eventuali nuovi server
  - Non esiste un registro di nomi e servizi
    - Può essere difficile trovare i servizi disponibili

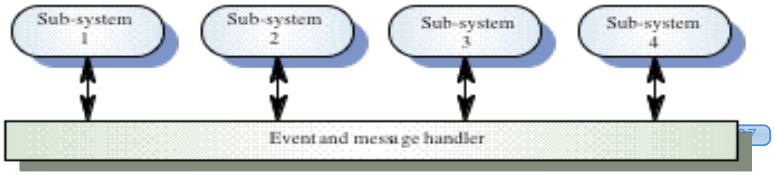
Modello a macchina astratta (o a strati)

- Organizza un sistema in un set di strati (o macchine astratte) ognuno dei quali fornisce un set di servizi
- Esempio modello OSI [1980]
- Supporta lo sviluppo incrementale dei diversi livelli. Quando l'interfaccia di un livello cambia solo il livello adiacente è influenzato
- Le prestazioni possono degradare



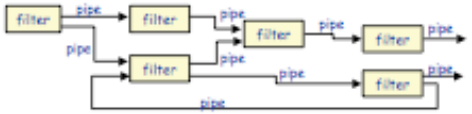
Event-driven

- Eventi esterni pilotano i sottosistemi di elaborazione
- Broadcast
  - Un evento è inviato a tutti i sottosistemi
  - Ogni sottosistema che è in grado di gestire l'evento lo gestisce
  - I sottosistemi decidono se l'evento è di interesse
- Interrupt-driven
  - Usati in sistemi realtime
- Esempi: fogli elettronici, sistemi di produzione
- Pro: Nuovi sottosistemi possono facilmente essere aggiunti
- Contro: un sottosistema non sa chi gestisce gli eventi che genera



Pipe and filter

- Ogni filtro trasforma i dati in input e passa i risultati in output
- I filtri non conoscono a chi sono connessi
  - Conoscono solo che riceveranno dati in ingresso dal canale di comunicazione
- I filtri possono essere implementati in parallelo e riusati
- Il comportamento del sistema è la composizione del comportamento dei filtri
- Esempi: shell comandi Unix, Compilatori



Riflessione Computazionale [1981]

- Un sistema riflessivo è un sistema a livelli in cui
  - Il livello sottostante (livello base) non conosce che esistono i livelli superiori (livelli meta)
  - Un livello superiore è in grado di ridefinire alcune operazioni del livello sottostante e di alterare le attività ed il flusso di esecuzione del livello inferiore, ai fini di
    - Cambiare la natura delle operazioni eseguite e per inserire nuove funzionalità
- A runtime, il livello superiore ***intercetta*** le operazioni del livello inferiore ed ha la capacità di ***ispezionare*** [introspect] il livello inferiore
  - Ispezionare serve a conoscere la struttura del livello inferiore (es. classi, attributi, metodi) ed il valore di variabili a runtime