DIETI UNINA UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

Corso di Laurea in Ingegneria Informatica

Corso di Ingegneria del Software

Prof. Roberto Pietrantuono

Progettazione

Obiettivi della progettazione

Attività ponte tra i requisiti e l'implementazione: trasforma i requisiti nell'architettura di un prodotto finito

- Nella fase di progettazione si decidono le modalità di passaggio da "**che cosa**" deve essere realizzato (specifica dei requisiti) a "**come**" la realizzazione deve avere luogo
- Progettare significa eseguire un processo di definizione della soluzione al problema, il cui obiettivo è trovare e descrivere un modo:

- Per implementare i requisiti *funzionali*...
- … rispettando i vincoli imposti dai requisiti *non funzionali*
    - inclusi i vincoli sul budget
- … in conformità con una serie di principi di *qualità*

Dominare la complessità

- Approccio generale: **scomposizione** del sistema in parti e **assegnazione di responsabilità** a ciascuna di esse al fine di raggiungere gli **obiettivi globali**.
- Scomposizione del sistema in parti:
  - ***complessità*** delle singole parti ***minore*** della complessità totale
  - le parti possono essere realizzate ***indipendentemente***
- Il primo risultato della progettazione è la **definizione dell'architettura software** che mette in luce le parti principali del sistema e come esse si combinano e cooperano
- Il risultato finale è il **progetto software**, che definisce la scomposizione in moduli, le funzioni assegnate ad essi, e le loro inter-relazioni

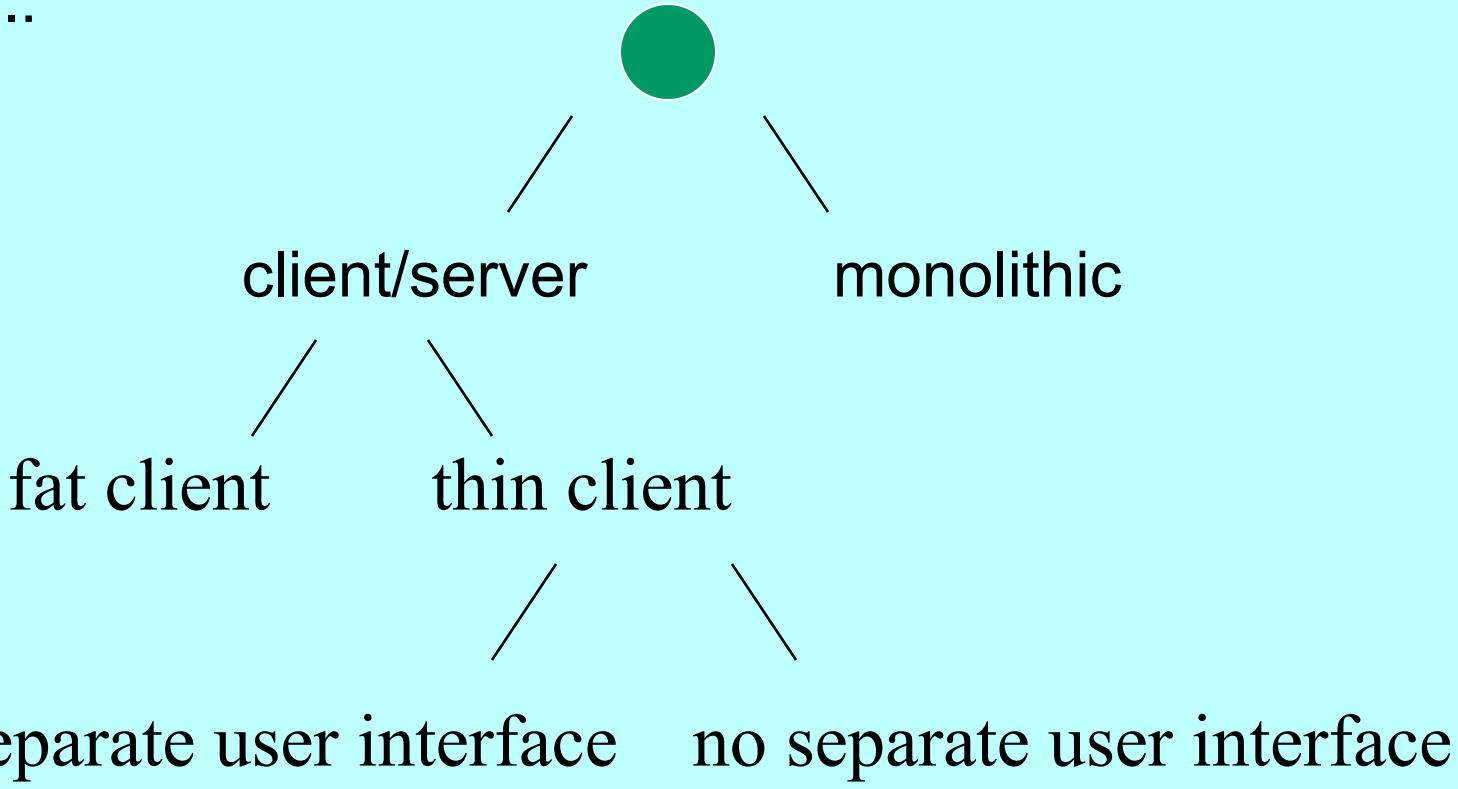
Design decisions

- **Un progettista affronta una serie di problemi di *design***
  - sotto-problemi del problema complessivo della progettazione
  - Ciascun problema ha normalmente più soluzioni alternative:
    - ***design options***.
  - Il progettista effettua una serie di ***design decisions*** (**scelte di progetto**, vincolanti per le attività successive) per risolvere ciascun problema
    - Sceglie la migliore fra le alternative possibili

- Ad una stessa specifica possono corrispondere più progetti, ossia più soluzioni

Design space

- Lo spazio dei possibili progetti scegliendo tra le alternative è chiamato ***design space***
- Es.:



Principi utili in fase di progettazione

- Oltre alla ***progettazione in vista del cambiamento,*** il principio della **modularità** e le tecniche di modularizzazione sono estremamente importanti per la progettazione
  - Corretta identificazione dei moduli e delle loro relazioni (ad es. fattorizzazione delle parti comuni, identificazione moduli riusabili)
  - **Alta coesione, basso accoppiamento**
- Altri principi importanti sono: ***separazione degli interessi, rigore e formalità, information hiding, riusabilità***

Attività di progettazione

- Progettazione dell'architettura software
  - sottosistemi/componenti
- Progettazione delle interfacce
- Progettazione dei componenti
- Progettazione delle interfacce utente
- Progettazione dei dati persistenti
- Progettazione degli algoritmi
- Progettazione dei protocolli

Progettazione Orientata agli Oggetti (OOD)

- La **progettazione orientata agli oggetti**, molto diffusa grazie anche alla diffusione dei linguaggi ad oggetti (e.g., C++, Java), utilizza **tipi di dato astratti**
- L'approccio tipico è **bottom-up**
- Nella modellazione ad oggetti si individuano gli ***oggetti*** e le ***classi*** e le loro relazioni strutturali (statiche) e comportamentali (dinamiche)

Dall' analisi al progetto

- In **fase di analisi**, si mira a modellare le entità del **dominio del problema**, le loro proprietà e responsabilità, le loro relazioni e le loro interazioni, a prescindere da come esse saranno realizzate nel software
- In **fase di progetto**, vengono introdotte oggetti e classi legati al **dominio della soluzione**, identificando le strutture necessarie a risolvere il problema descritto. Porta alla ristrutturazione/arricchimento di quanto prodotto nella fase di analisi
- Si arriva ad un definizione di progetto dettagliata, che può anche dar luogo alla generazione automatica del codice.

OOD

- In fase di progettazione, i modelli prodotti nell'analisi vengono rivisitati e viene definita l'architettura software
- Si deve trovare soluzione ai problemi di ***interfacciamento, persistenza, concorrenza, prestazioni, allocazione fisica, collaborazione distribuita, collezionamento di dati/oggetti, …***
  - Ciò comporta l'aggiunta di ulteriori elementi e il raffinamento degli esistenti.
- **Interfacciamento attore-sistema:** non esistono regole per determinare tali entità. Una soluzione è introdurre **oggetti coordinatore** per la conduzione di uno o più casi d'uso e di un **set di oggetti di interfaccia** *(ad es. finestre per l'inserimento di informazioni e la stampa a video)*

OOD

- **Collezionamento degli oggetti**: si può far ricorso a *classi di libreria* (come vettori, collezioni, tabelle hash) o *classi specifiche*, ad es. liste collegate
- **Persistenza degli oggetti**: uso di *DBMS OO*, oppure uso di soluzioni *ad hoc* (come "ognuno salvi se stesso" o uso di *classi per la memorizzazione*)
- **Collaborazione distribuita**: uso di *middleware* o soluzioni *ad hoc* (come *pattern di progettazione, proxy*)
- A valle di questa fase, le classi saranno raggruppate in ***package*** (il sistema può essere organizzato in sottosistemi, se complesso)

Diagrammi UML nel ciclo di vita

... Requirements ... Design ... Implementation

Use Case

Class diagram

Sequence diagram

Activity diagrams and Statecharts

Component diagram

Deployment diagram

Package diagram

Diagrammi UML in progettazione

- Diagramma dei componenti (per sottosistemi/componenti e loro interfacce)
- Diagramma delle classi (di progettazione)
- Diagrammi di sequenza (per le interazioni tra più oggetti tramite scambio di messaggi)
- Diagrammi di stato (per il protocollo di interazione con un oggetto e/o per il comportamento interno di un oggetto)
- Diagrammi di attività (per i dettagli di un'operazione o di un algoritmo, o per un processo)
- Diagramma dei package (per la struttura del codice)
- Diagramma di allocazione (in forma descrittore) per l'allocazione del software sull'hardware

Diagramma delle classi: da analisi a progettazione

- **Diagramma delle classi** (*class diagram* – CD) è usato per creare un *modello concettuale* del dominio, ovvero esso fornisce una rappresentazione delle entità del mondo reale presenti nel dominio di interesse (***domain objects***) e cattura le relazioni tra di esse.
- In analisi, non occorre completare il CD con quei dettagli (e *ornamenti UML*) che non sono rilevanti ai fini di comprensione del dominio, come il tipo delle proprietà delle classi o la cardinalità delle associazioni.

Diagramma delle classi: da analisi a progettazione

- Il CD di prima analisi ha dunque lo scopo di rappresentare i concetti significativi del dominio del problema, la terminologia del problema e il contenuto informativo del dominio.
- Esso rappresenta altresì un'ontologia del dominio del problema

Diagramma delle classi: da analisi a progettazione

- Il modello di analisi si può raffinare in più iterazioni, aggiungendo dettagli, ornamenti UML, e rendendolo più completo, nella descrizione del dominio del problema e nell'interazione del sistema con gli attori.
- Ad esempio, in un raffinamento del **diagramma delle classi** aggiungiamo alle associazioni il loro **tipo, la loro cardinalità, la navigabilità e i ruoli**

Diagramma delle classi: da analisi a progettazione

- Il **diagramma delle classi di progetto** è ottenuto raffinando il diagramma di analisi, aggiungendo i dettagli tralasciati nel precedentemente, ed offrendo una soluzione al problema.
- Il dettaglio offerto dai design-CD deve essere tale da specificare l'implementazione. In particolare:
  - Le classi devono specificare l'insieme completo dei loro attributi (dettagliando nome, tipo, visibilità, etc.) e delle operazioni (dettagliando nome, nome e tipi dei parametri, tipo del valore restituito, visibilità, etc.);
  - Tutte le associazioni vanno raffinate e ben definite (tipo, cardinalità, direzione e ruolo).

Diagramma delle classi: da analisi a progettazione

- **Al termine, la progettazione offre:**
  - **classi *complete*** (che offrono *almeno* ciò che i loro utilizzatori attendono), ***sufficienti*** (che offrono *solo* ciò che i loro utilizzatori attendono), ed ***essenziali*** (i servizi offerti devono essere primitivi, semplici, atomici e distinti);
  - **Moduli** (classi, componenti, package, etc.) con ***massima coesione* e *minimo accoppiamento*** (poche *relazioni di dipendenza* tra gli elementi);
  - **Ulteriori entità** (classi, oggetti, componenti, etc.) che sono introdotte per rispondere ad esigenze e scelte progettuali (es.: un componente che offre servizi di caching per migliorare le performance).

Diagramma di sequenza: da analisi a progettazione

- Il **diagramma di sequenza** (*sequence diagram* – SD) mostra la dinamica delle iterazioni tra gli attori e le entità del dominio per realizzare una funzionalità del sistema (uno o più scenari di un caso d'uso)

Diagramma di sequenza: da analisi a progettazione

- Raffinare un diagramma di sequenza è utile per verificare se la modellazione proposta delle classi di analisi è adeguata per catturare i requisiti sui dati del sistema
- Ovvero se le associazioni e gli attributi identificati sono quelli **sufficienti** e **necessari** per permettere lo svolgimento dei casi d'uso (se non sono sufficienti, allora c'è qualcosa da aggiungere alla modellazione; se non sono necessari, c'è qualcosa da rimuovere).