Progettazione di dettaglio
Diagrammi di struttura composita

Laura Semini
Ingegneria del Software
Dipartimento di Informatica
Università di Pisa

Progetto 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 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

Diagramma di struttura composita

- Diagramma che mostra la struttura di dettaglio (o struttura interna) di:
  - un classificatore strutturato (che vedremo) o
    - di solito di un componente, ma anche di una classe
  - di una collaborazione (che non vedremo)

Classificatore strutturato

- È un classificatore di cui si vede la struttura interna, data in termini di parti, porti e connettori
- Un classificatore strutturato definisce l'implementazione di un classificatore
  - così come le interfacce del classificatore definiscono cosa deve fare
  - la sua struttura interna definisce come viene fatto il lavoro
- I tipi che definiscono le parti contenute in un classificatore strutturato, possono a loro volta essere strutturati, ricorsivamente

Parti

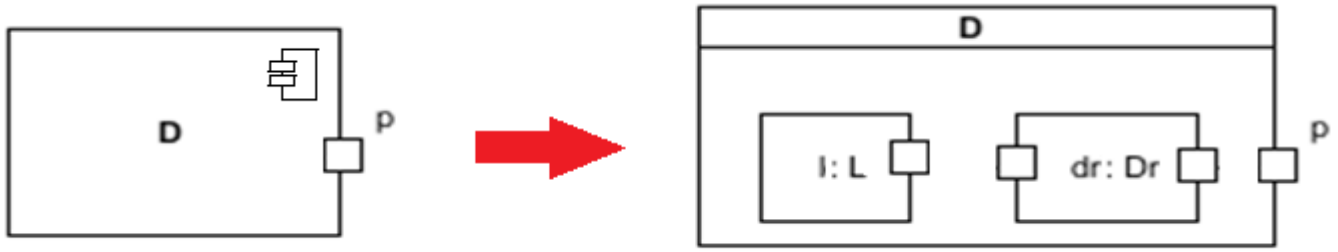
- Una parte ha un nome, un tipo e una molteplicità (anche se sono tutti facoltativi)

nomeParte : Tipo [molt]

- Una parte p:T descrive il ruolo che una istanza di T gioca all'interno dell'istanza del classificatore la cui struttura contiene p
- La molteplicità indica quante istanze possono esserci in quel ruolo
- Un'istanza di p è un'istanza di T (e quindi di un qualunque sottotipo)

Porti

- Un porto è un insieme di interfacce omogenee, come nei diagrammi di componenti.
- Anche in questi diagrammi è rappresentato con un quadratino.
- La struttura composita che mostra la struttura di dettaglio del componente D ha
  - tutti i porti di D sul proprio bordo,
  - i porti associati alle parti che definiscono la struttura interna di D, che permettono le interazioni tra loro e con l'esterno.

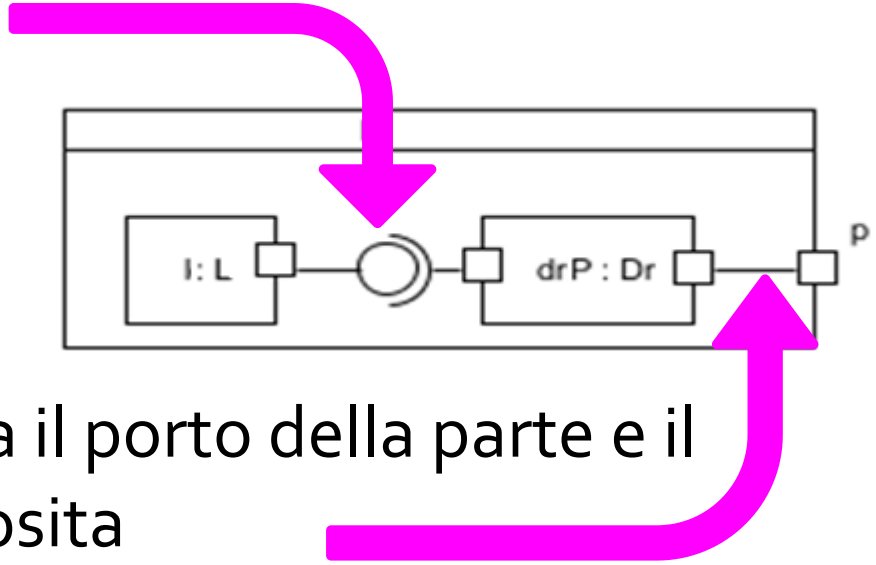


Connettore

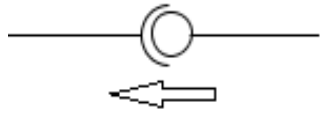
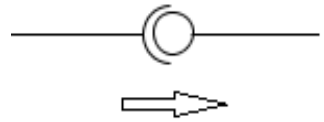
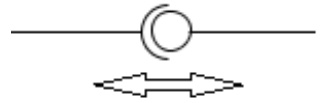
- Un connettore può essere di due tipi
  - di assemblaggio (assembly connector)
    - tra due parti
    - esprime un legame che permette una comunicazione tra due istanze dei ruoli specificati dalla struttura
  - di delega (delegation connector)
    - tra una parte e un porto della componente
    - identifica l'istanza che realizza le comunicazioni attribuite a un porto

Notazione connettori

- Assemblaggio
  - collega i porti delle due parti le cui istanze devono comunicare
  - si usa la notazione lollipop
- Delega
  - si usa una semplice linea tra il porto della parte e il porto della struttura composita



Osservazione: verso del lollipop rispetto al verso dell'informazione

- Il verso del lollipop non ha alcun legame con il verso in cui viaggiano i dati: ha solo a che vedere con chi ha il controllo e con chi, interrogato, risponde
  - un'interfaccia con solo operazioni di read 
  - un'interfaccia con solo operazione di write 
  - un'interfaccia con operazioni di read e write 

Dalla specifica all'implementazione...

- ...passando per le strutture composite

Fig. 1

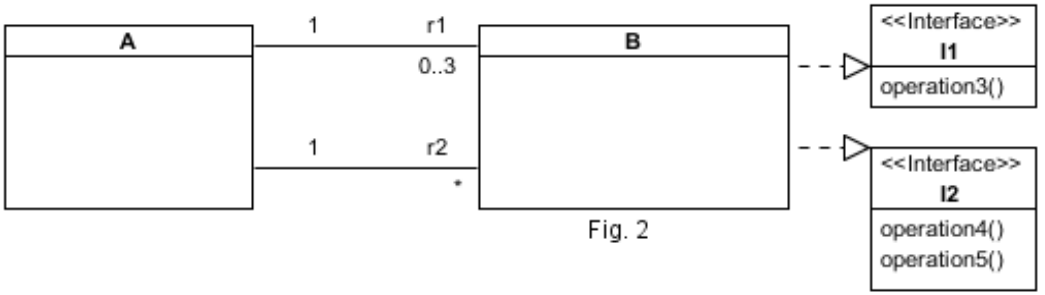


Fig. 2

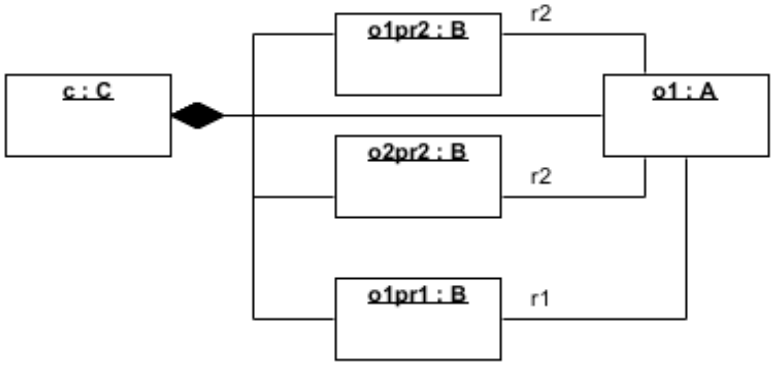


Fig. 3

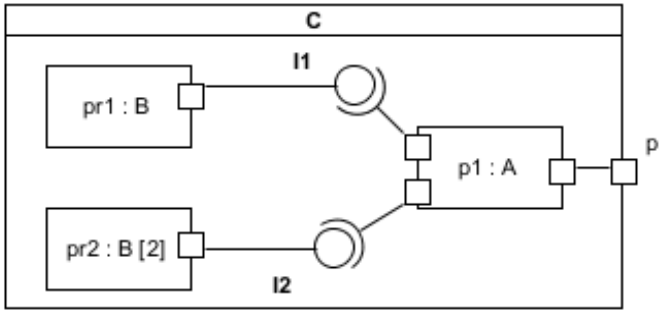


Fig. 4

Legenda

- In Fig. 1 si mostra la componente C
- In Fig. 2 si specificano le classi A e B, con associazioni, ruoli e interfacce. Si assume che un'istanza di A usi l'interfaccia I1 (I2) di B quando connessa con un'istanza di B in ruolo r1 (r2 risp.)
- In Fig. 3 si mostra un diagramma degli oggetti conforme al diagramma in Fig. 2
- In Fig. 4 si mostra la struttura di C, che fornisce una astrazione del diagramma in Fig. 3. Introduce vincoli rispetto al diagramma delle classi: dice che uso istanze di A e di B, collegate in un certo modo, con dato ruolo nel contesto di C, e date molteplicità

Come strutturare una componente

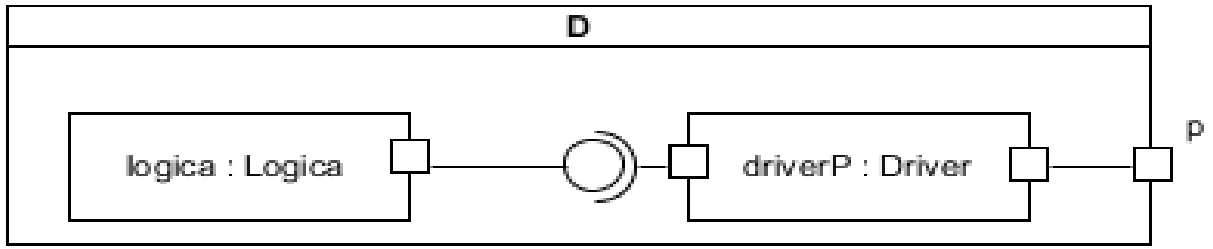
- Un modo conveniente di strutturare una componente prevede di separare gli aspetti di comunicazione da quelli di realizzazione delle funzionalità richieste
  - Favorisce modificabilità e comprensibilità della componente
  - Risponde ai principi generali di progettazione
    - Coupling
    - Information hiding

Un pattern generale di strutturazione

- Nei prossimi lucidi presentiamo un pattern molto generale, cioè applicabile a quasi tutte le componenti, per iniziare a definire la loro struttura interna.
- Questo pattern costituisce un modo per partire, la struttura definitiva sarà una specializzazione di quella iniziale, e dipenderà dalle specificità di ogni componente.

Pattern di strutturazione, 1

- Supponiamo di avere una componente D con un porto p
- La struttura di D dovrà avere almeno due parti
  - driverP, che realizza la parte di comunicazione richiesta per implementare il porto
  - logica, che realizza la funzionalità richiesta alla componente
- e due connettori
  - di delega tra driverP e P
  - di assemblaggio tra driverP e logica (il verso del lollipop non è fissato a priori)



Pattern di strutturazione, 2

- In generale, data una componente con n porti, la sua struttura minima dovrà prevedere
  - n parti col ruolo di driver
    - collegate ognuna a un porto, con un connettore di delega
  - una parte che realizza la logica della componente
    - collegata a tutti i driver con connettori di assemblaggio
- I nomi "driver" e "logica" per le parti di una componente sono usati in questo pattern, per aiutare a distinguere le loro responsabilità

Dettagliare maggiormente la struttura

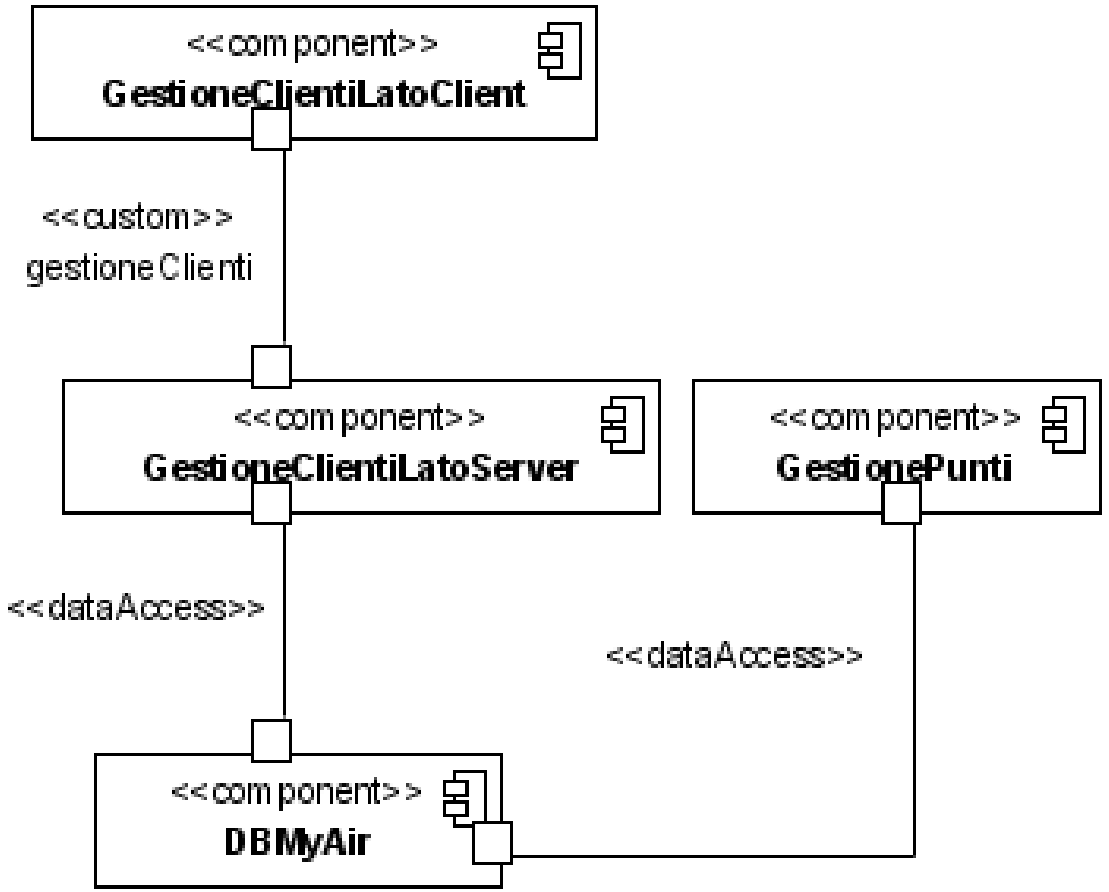
- È ovviamente possibile, e normalmente necessario, dettagliare maggiormente la struttura di una componente
  - raffinando la "logica" in un insieme di parti interconnesse
    - con connettori di assemblaggio
    - con dipendenze
  - introducendo dei "proxy", per realizzare comunicazioni con sistemi remoti o chiamate al sistema operativo
    - connessi alle parti che descrivono la logica, con connettori di assemblaggio
    - non sono collegati ai porti, perché non realizzano la comunicazione con altre componenti del sistema che si sta progettando
- l nome "proxy" è introdotto in questo pattern, per distinguere il ruolo da quello di un driver: non è corretto pensare sia simile al proxy di RMI

Riassumendo: Dare un diagramma di struttura composita della componente C

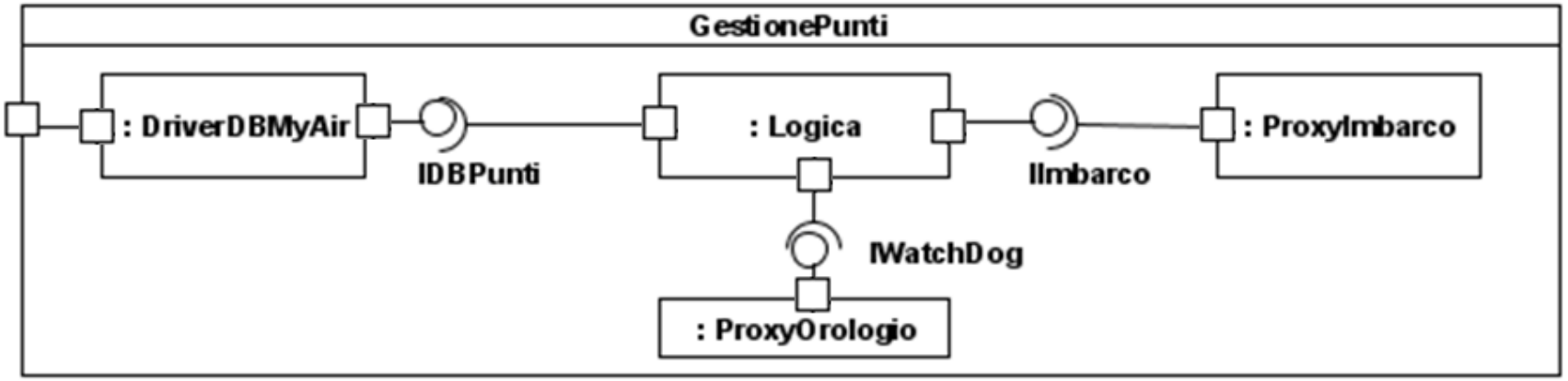
- Assumiamo C abbia 2 porti verso le componenti C1 e C1 e comunichi con un sistema esterno S
- La struttura di C dovrà avere almeno 4 parti
  - : DriverC1 e : DriverC2, drivers che realizzano la comunicazione con C1 e C2, rispettivamente
  - : Logica, che realizza la funzionalità richiesta alla componente
  - : ProxyS, proxy che realizza la comunicazione con S
- e un certo numero di connettori
  - 2 di delega tra i driver e i porti che realizzano
  - 3 di assemblaggio tra driver / proxy e logica (il verso del lollipop non è fissato a priori ma dipende dal sistema che si sta progettando)

Esercizio myAir

- Vista C&C: la componente GestionePunti realizza i casi d'uso AccumuloPunti e AggiornamentoAnnuale



Struttura interna di GestionePunti



Parte	Responsabilità
DriverDBMyAir	Realizza l'interfaccia IDBPunti che permette a Logica di accedere tramite il solo porto della componente al DBMyAir
ProxyImbarco	Realizza la connessione con il sistema Imbarco, passando alla Logica la lista degli imbarcati
ProxyOrologio	Sveglia la logica alla data e ora richiesta tramite IWatchDog
Logica	Realizza i casi d;uso, sfruttando le interfacce introdotte

Syllabus e osservazione

- Dispensa di architetture (Architetture software e Progettazione di dettaglio)
- Attenzione: l'articolo in calce alla dispensa non distingue tra proxy e driver.
- [Arlow 16.12.1 tranne per quanto riguarda la notazione del connettore]