Progettazione di dettaglio
Diagrammi di struttura composita

Roberta Gori, 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)
    - 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]

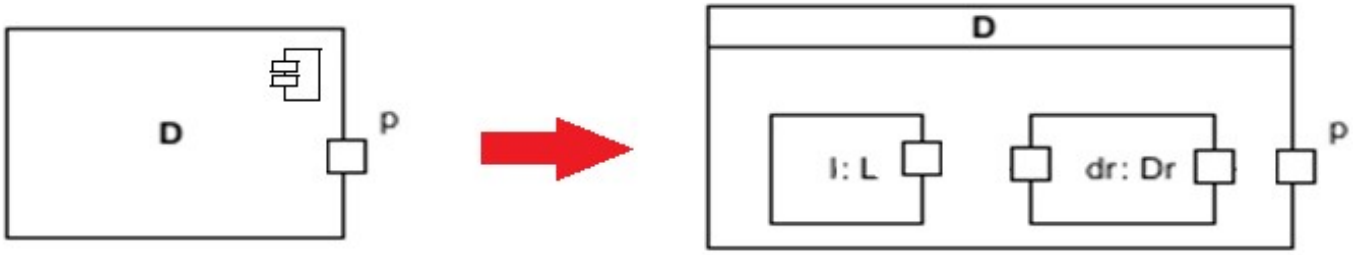
- Una parte p:T descrive il ruolo che una istanza di T gioca all'interno dell'istanza del classificatore la cui struttura contiene p
- La molteplicità indica quante istanze possono esserci in quel ruolo
- Un'istanza di p è un'istanza di T (e quindi di un qualunque sottotipo)

named role, multiplicity 1

named role, multiplicity many

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.

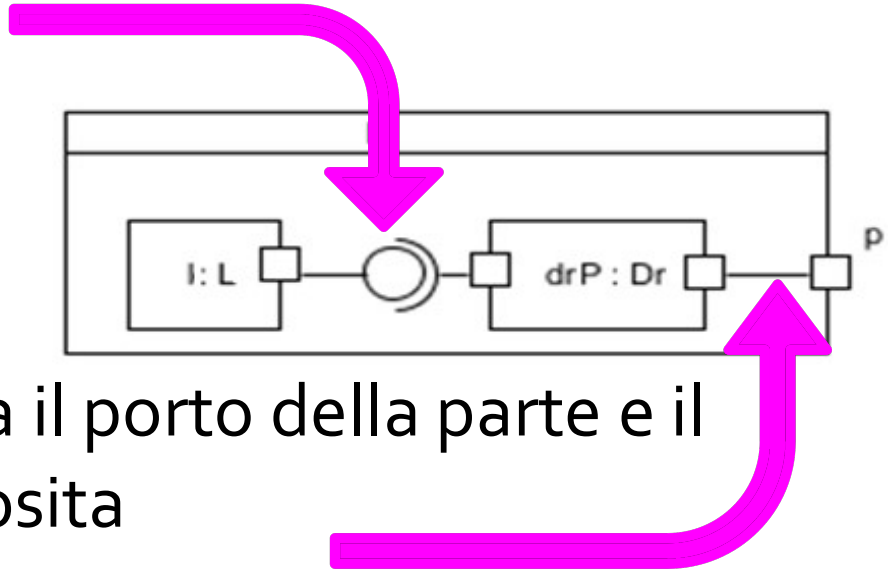


Connettore

- 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

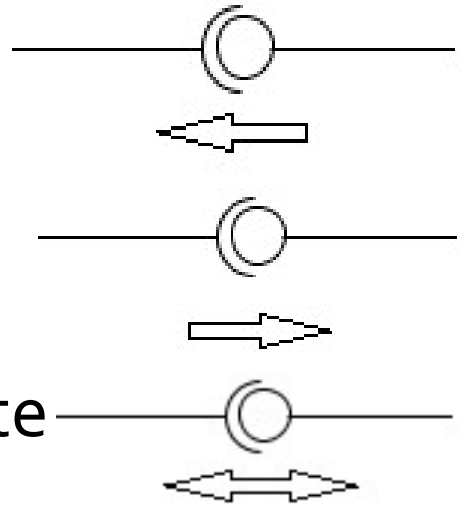
Notazione connettori

- 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 (frecce nei disegni): 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

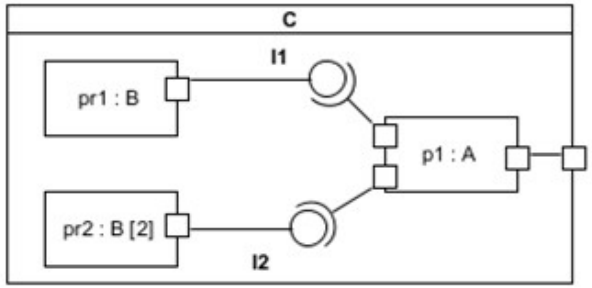


Fig. 2

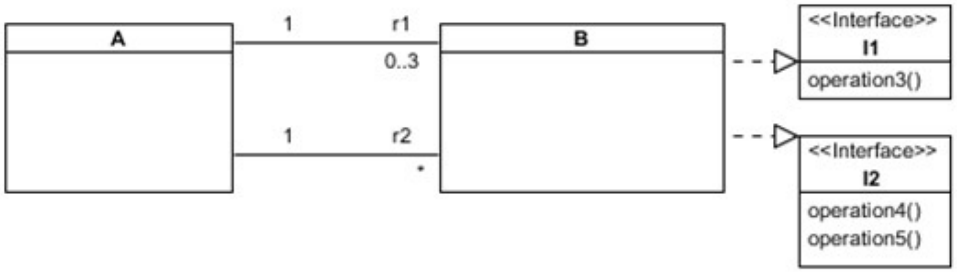


Fig. 3

Fig. 4

Legenda

- In Fig. 1 si mostra la componente C
- In Fig. 2 si mostra la struttura di C: dice che si usano istanze di A e di B, collegate in un certo modo, con dato ruolo nel contesto di C, e date molteplicità
- In Fig. 3 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. 4 si mostra un diagramma degli oggetti conforme ai diagrammi in Fig. 2 e in Fig. 3

Come strutturare una componente

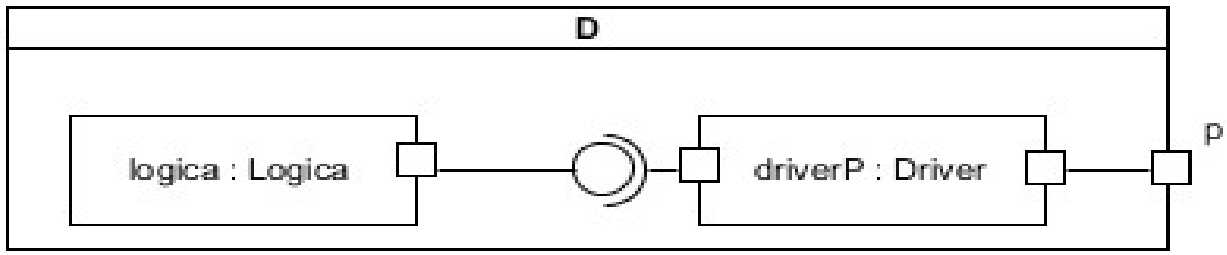
- Un modo conveniente di strutturare una componente prevede di separare gli aspetti di comunicazione da quelli di realizzazione delle funzionalità richieste
  - Favorisce modificabilità e comprensibilità della componente
  - Risponde ai principi generali di progettazione
    - Coupling
    - Information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

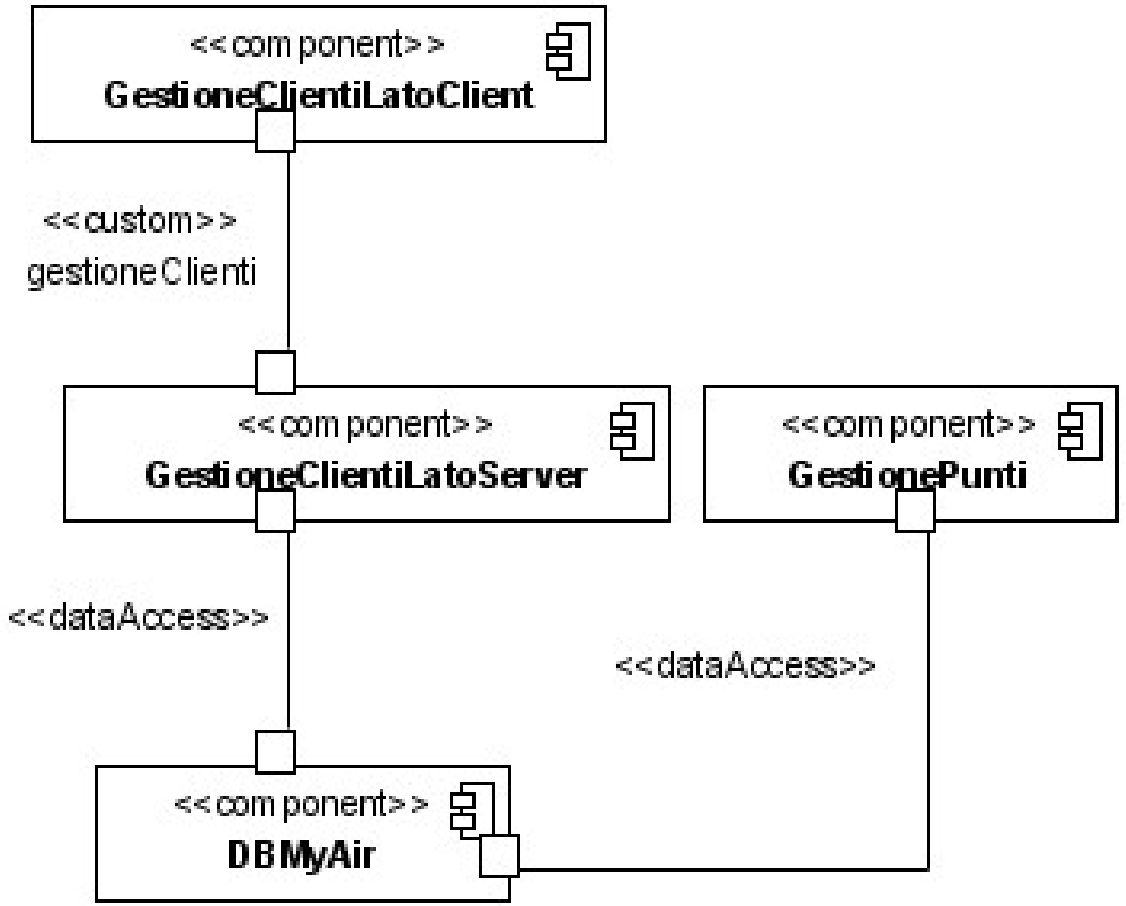
- È ovviamente possibile, e normalmente necessario, dettagliare maggiormente la struttura di una componente
  - raffinando la "logica" in un insieme di parti interconnesse
    - con connettori di assemblaggio
    - con dipendenze
  - introducendo dei "proxy", per realizzare comunicazioni con sistemi 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

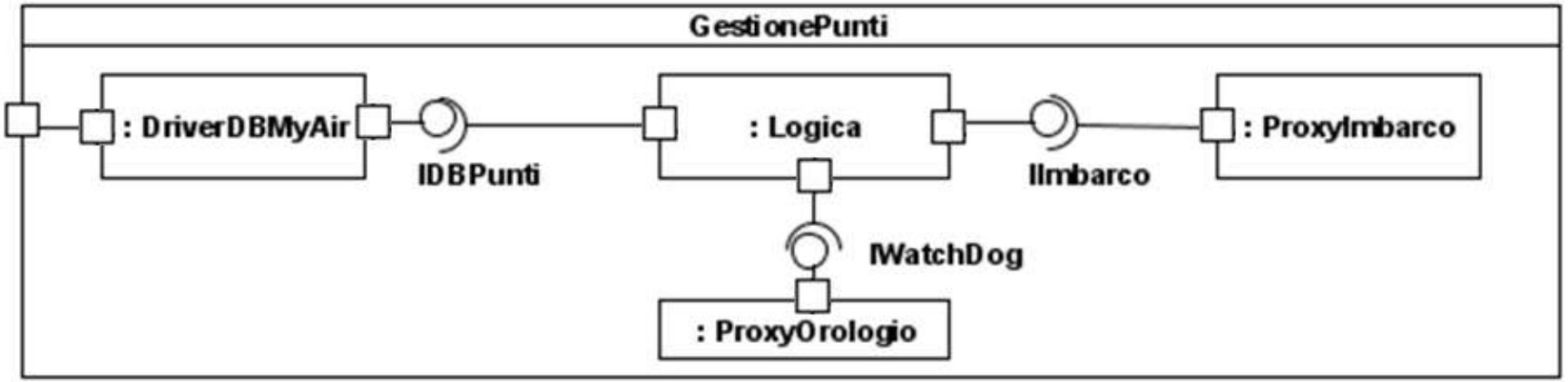
- Assumiamo C abbia 2 porti verso le componenti C1 e C2 e comunichi con un sistema esterno S
- La struttura di C dovrà avere almeno 4 parti
  - : DriverC1 e : DriverC2, drivers che realizzano la comunicazione con C1 e C2, rispettivamente
  - : Logica, che realizza la funzionalità richiesta alla componente
  - : 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 Aggiornamento Annuale



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

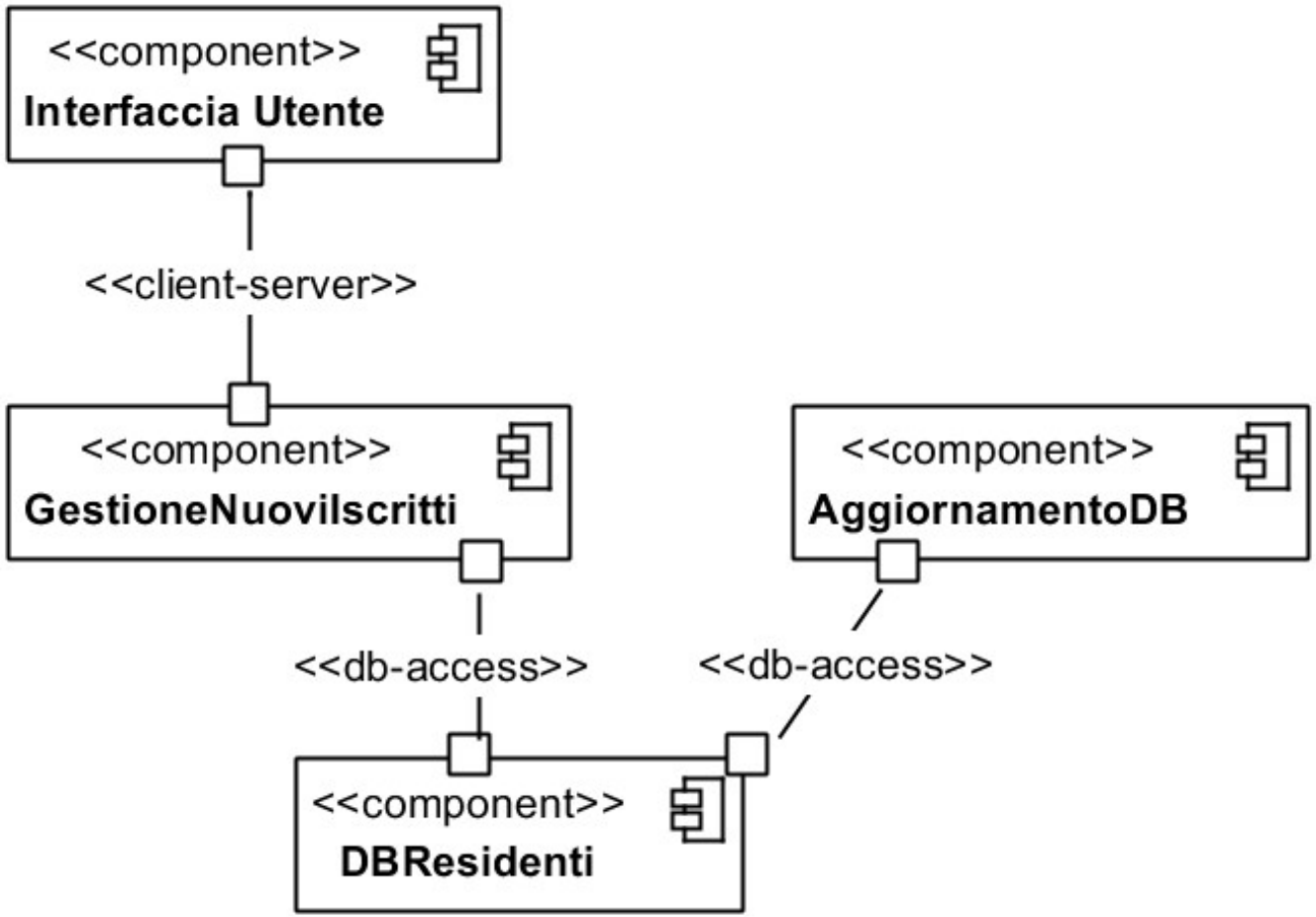
Esercizio: PisaMover

- *Si consideri la nuova tariffa scontata per i biglietti stazione ferroviaria-aeroporto per i residenti del comune di Pisa.*
- *L'utente interessato deve registrarsi presso la biglietteria aziendale o via web, fornendo generalità, comune di residenza e codice fiscale.*
- *Il sistema controlla la veridicità della dichiarazione usando i dati mantenuti in locale, e in caso di verifica positiva il suo codice fiscale viene inserito tra quelli accettati dalle biglietterie automatiche. In questo modo l'utente potrà acquistare biglietti a prezzo scontato presso le biglietterie automatiche, indicando il codice fiscale.*
- *Ogni tre mesi il sistema richiede la lista dei residenti del comune di Pisa, con una richiesta a un servizio offerto dall'Anagrafe, e aggiorna il proprio database: elenco dei residenti da consultare per future richieste di registrazione; lista dei codici fiscali accettati dalle biglietterie automatiche, (rimuovendo i non più residenti).*

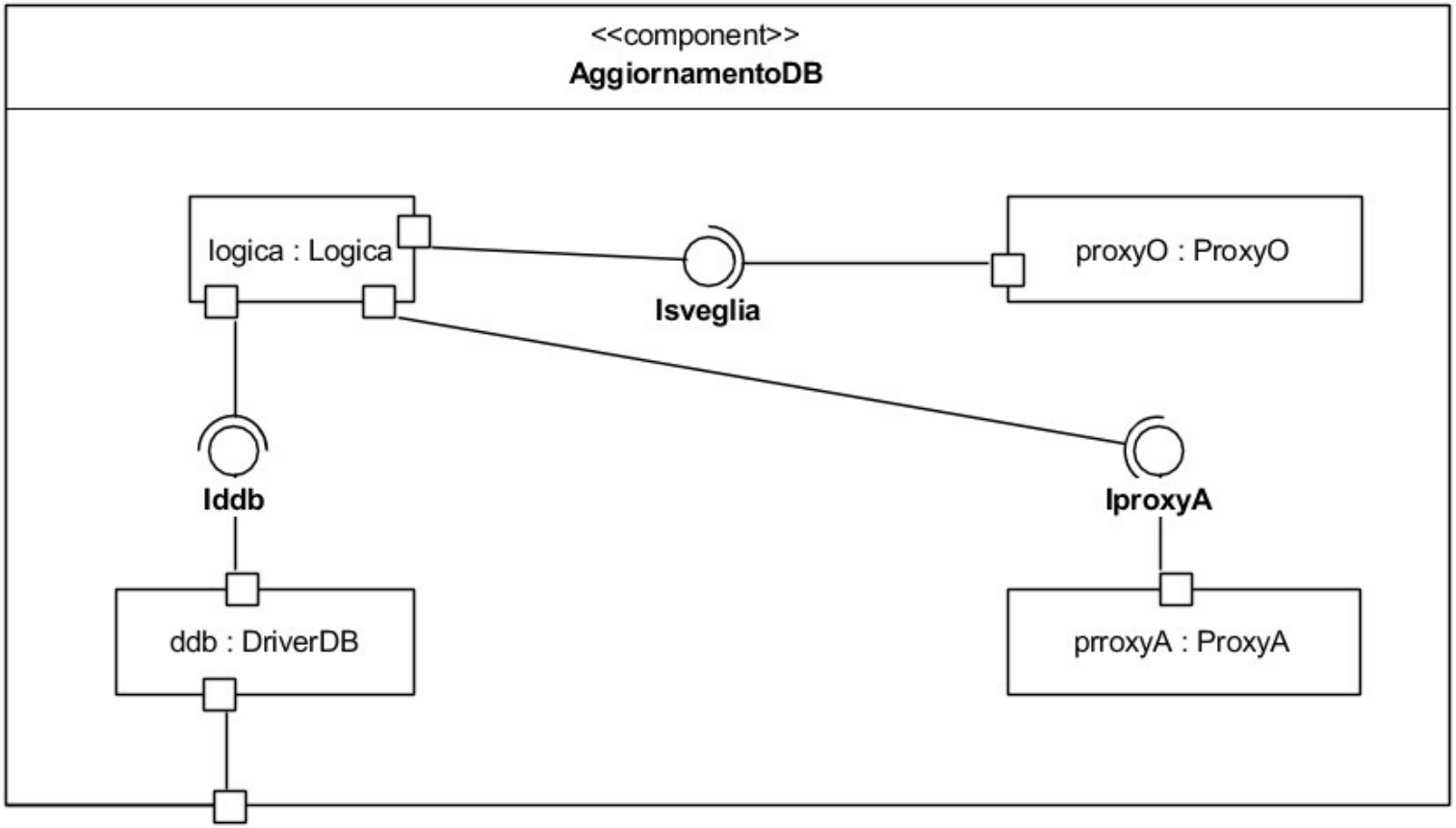
- Si disegni un diagramma C&C, in cui si evidenziano i principali componenti software necessari per realizzare il nuovo requisito relativo alla gestione dei biglietti scontati.

- Si dia un diagramma di struttura composita della componente che comunica con l'Anagrafe.

C&C


<<component>>
Interfaccia Utente
<<client-server>>
<<component>>
GestioneNuovilscritti
<<component>>
AggiornamentoDB
<<db-access>>
<<db-access>>
<<component>>
DBResidenti

Struttura composita



Nella soluzione data la logica offre un'interfaccia al proxy dell'orologio e non viceversa come in altre soluzioni. Il proxy ha un thread che «dorme» per tre mesi e poi invoca Isveglia per «svegliare» la logica

Rebu, estensione al problema

Con riferimento al caso di studio REBU si consideri un nuovo requisito. Per motivi di sicurezza, sono state introdotte delle limitazioni al numero di ore consecutive di lavoro per un autista.

Un turno di lavoro può durare al massimo 9 ore, di cui una dedicata a una pausa pranzo, da effettuarsi non prima di 3 e non dopo 5 ore di guida. Eventuali altre pause sospendono il turno ma non verranno conteggiate come pausa pranzo.

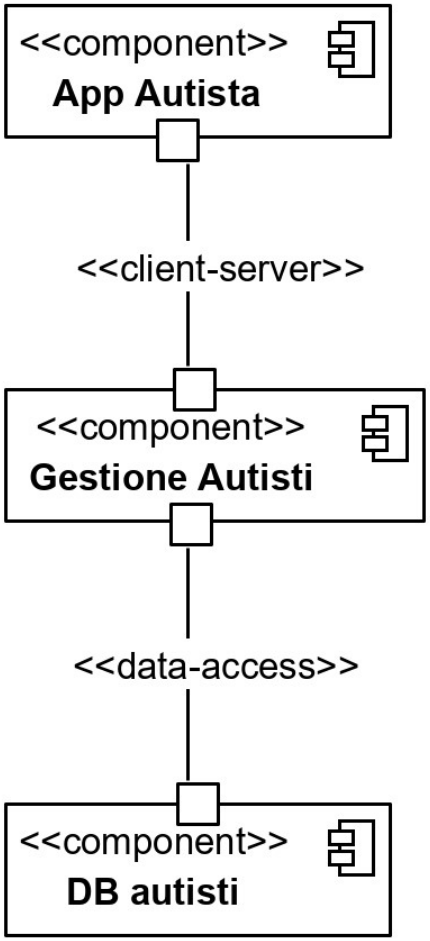
Quando un autista chiede di iniziare un nuovo turno, il sistema lo autorizza e quindi monitora l'autista durante il turno, imponendo uno stop ove necessario.

L'autista segnala inizio e fine turno e inizio e fine di ogni pausa,
Il sistema impone una pausa pranzo dopo 5 ore, se non già effettuata e lo stop dopo 9 se l'autista nin si è già fermato prima.

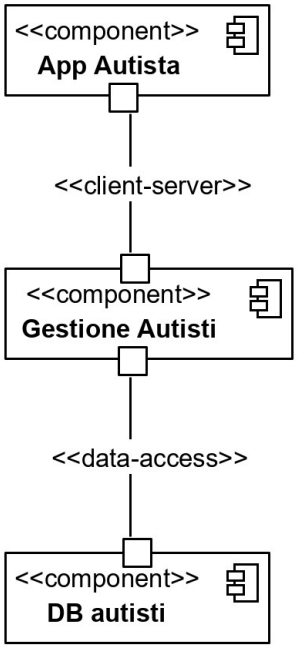
Architettura

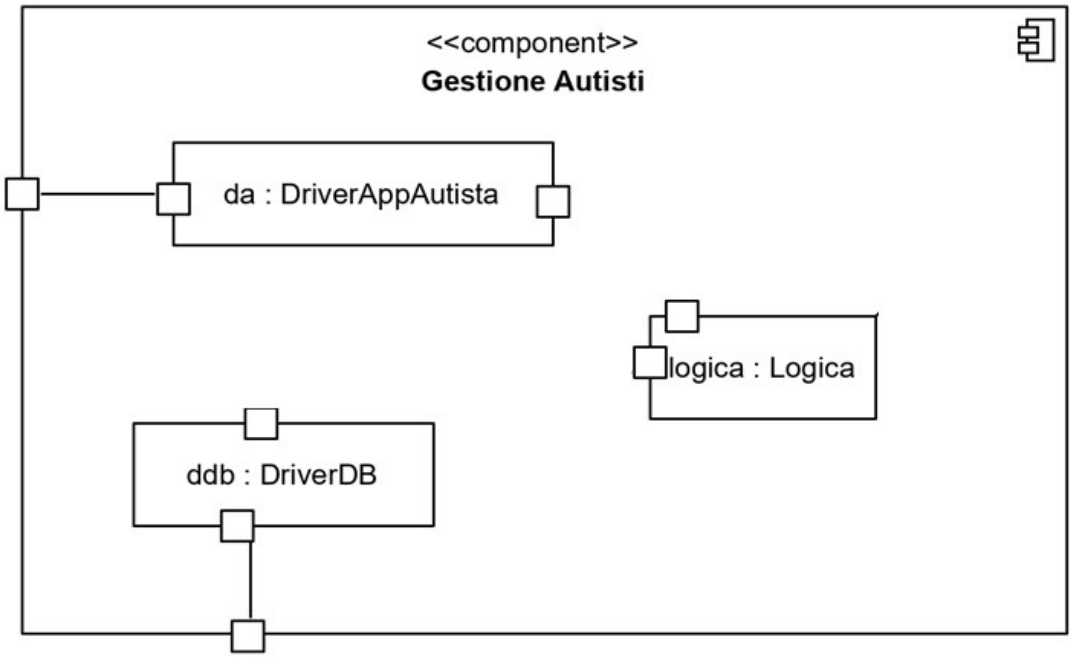
- La progettazione architetturale ha individuato le seguenti componenti:
  - DB autisti, che mantiene informazioni sugli autisti, comprensive del loro conto economico;
  - Gestione Autisti, che realizza la business logic.
  - App Autista, che realizza l'interfaccia dell'autista
- Dare
  - la vista C&C
  - Dare il diagramma di struttura composita di Gestione Autisti (Sapendo che l'implementazione si basa sulla creazione di oggetti di tipo Turno ogni volta che viene iniziato un nuovo turno)

C&C

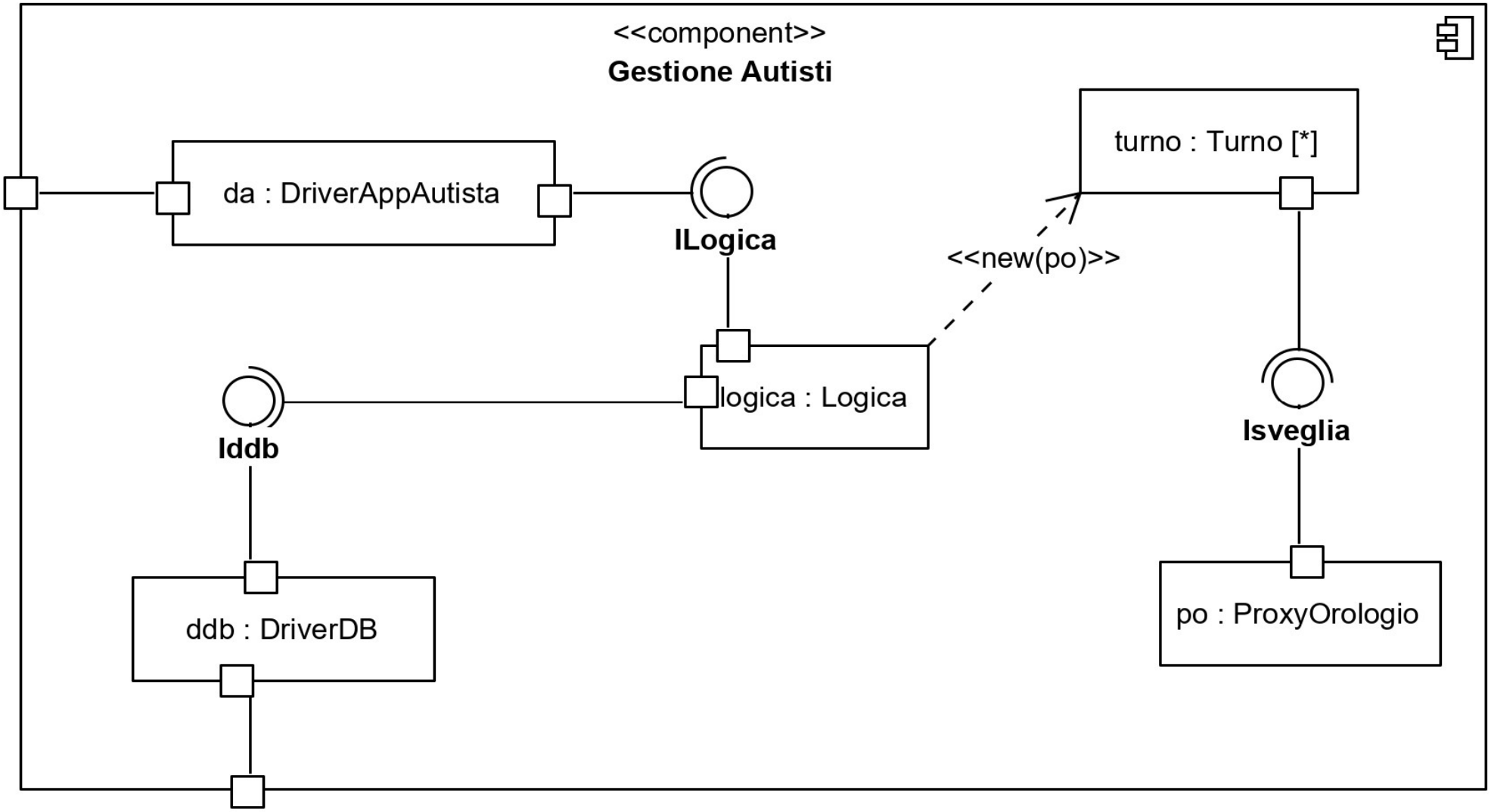

<<component>>
App Autista
<<client-server>>
<<component>>
Gestione Autisti
<<data-access>>
<<component>>
DB autisti

Dalla vista C&C al dettaglio di una componente: le parti di comunicazione e la logica della componente





Completare la struttura interna della componente



Syllabus e osservazione

- Dispensa di architetture (Architetture software e Progettazione di dettaglio)
- Attenzione: l'articolo in calce alla dispensa, lettura facoltativa per gli interessati, non distingue tra proxy e driver.