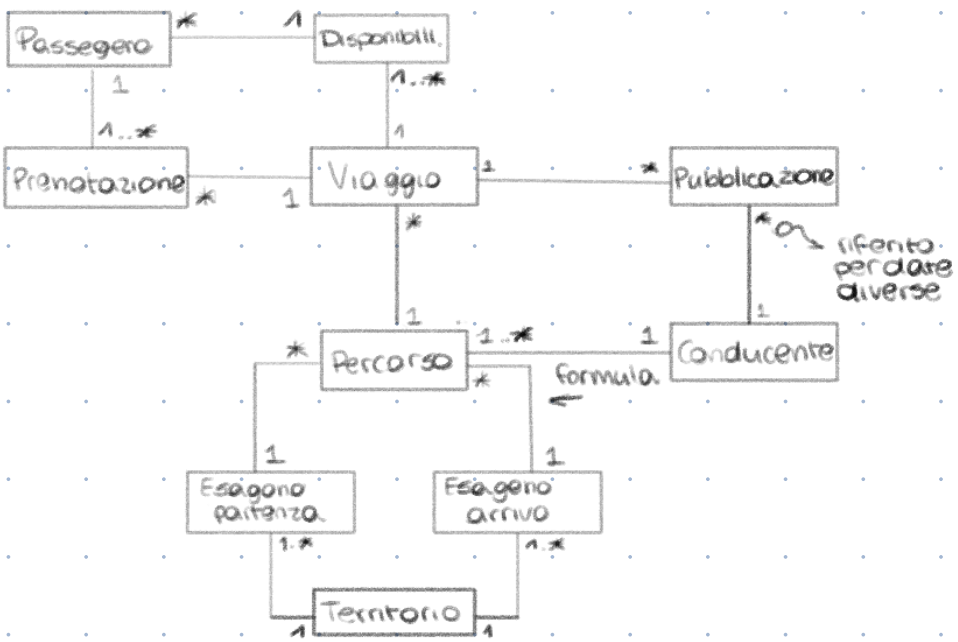
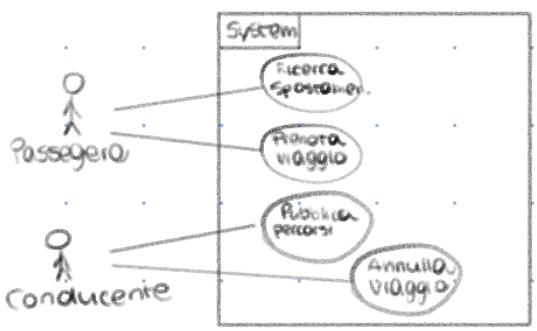
Domain model



Use case



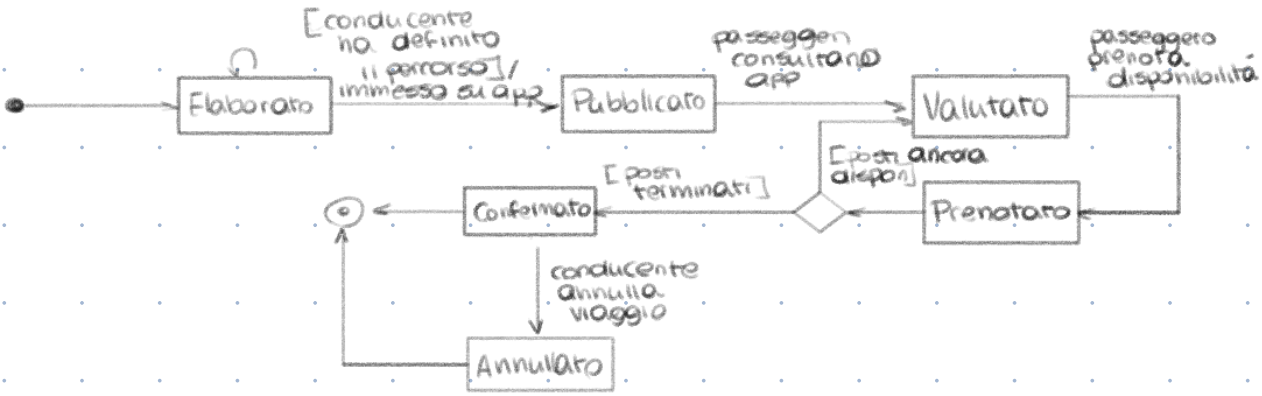
UC: Prenota viaggio
Attore: Passeggero
Pre: Passeggero autenticato dal sistema.
Passeggero abbia visualizzato disponibilità.

1. Passeggero seleziona prenotazione
2. Sistema richiede immissione dati personali
3. Passeggero inserisce dati
4. Sistema conferma prenotazione

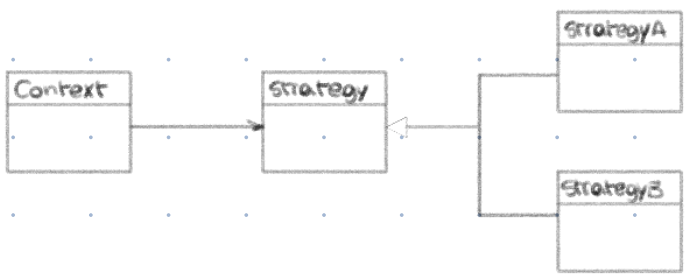
Post: Viaggio annullato dal conducente.

Stare diagram

1. Prenotato
2. Annullato
3. Confermato
4. Definito
5. Pubblicato
6. Visualizzato
7. Elaborato



② Strategy é un pattern comportamentale attuato in totale simbiosi rispetto alla richiesta del meccanismo di composizione, su cui caratterizza tutti i pattern del catalogo GOF, cioé che ogni classe software modelli un unico comportamento
L'intento del pattern consiste nel separare un oggetto da un proprio comportamento affinché sia intercambiabile durante run-time.
Proseguendo, la soluzione promuove l'utilizzo di un' interfaccia, sovrapposta tra classe di alto livello e business logic, affinché sia soddisfatto il principio dipendancy inversion, il quale ammette che entità software non dovrebbero mai dipendere da classi concrete e garantendo, soprattutto, estensione comportamentale da parte di vari e propri sviluppi, dove qualora si presenti la necessità di una nuova funzionalità essa si tradurra in una nuova classe che implementi il nuovo comportamento, favorendo in questo modo anche open-closed principle, in quanto il sistema adegua nuove funzionalità senza modificare metodi già esistenti
A livello implementativo la classe esterna inizializza un' istanza dell' interfaccia affinché possa imporre quando variare il proprio comportamento adeguando il metodo incapsulato al suo interno, poi realmente implementato dalle classi che estendono l'astrazione.
Si noti che in questo contesto é la high level class a stabilire qualora attuare intercambiabilità comportamentale.
Un esempio di quanto detto é



Similitudine rispetto a State, ma in esso é adottato un meccanismo automatizzato, basato sul riferimento dello stato corrente dell' oggetto, pertanto é proprio l'interfaccia ad imporre quando attuare intercambiabilità comportamentale:

③ Polymorphism é un software pattern del modello GRASP, attuato per gantire una migliore gestione delle dipendenze, principali artefici della disfatta di differenti sistemi software.
Semplicemente é un pattern che ovvia all'implementazione di statici e rigidi costrutti di selezione, poché renderebbe il codice difficile da comprendere, provocando quindi design smells come opacity, e complicato da manipolare almanzi a modifiche.
Tramite la soluzione é garantita la possibilità che classi derivate possano specializzare il contenuto di un metodo e di attuarlo qualora richiamato.
Per cui l'assegnamento del vincolo comportamentale consiste nella specializzazione di metodi ereditati, potenzialmente all'interno da una gerarchia di sottotipi, a cui sia posta anche la soddisfazione di compatabilità comportamentale, in modo tale che si eviti di sviluppare complessi costrutti di selezione, ma richiamando solamente il metodo dell'oggetto posseduto.
Un esempio é dato dalla gerarchia di figure geometriche, suddivise in shape, superclasse, rettangolo e quadrato, sottoclassi.
Tutte le entità possiedono il metodo getArea(), pertanto, evitando di adeguare costrutti di selezione, ad ogni entità derivata é assegnata la responsabilità di specializzarne il contenuto, in modo tale che se richiamata la funzionalità su uno dei due termini software finali, ottenendo il risultato richiesto mediante un flusso di controllo impartito dalla superclasse verso il destinato.