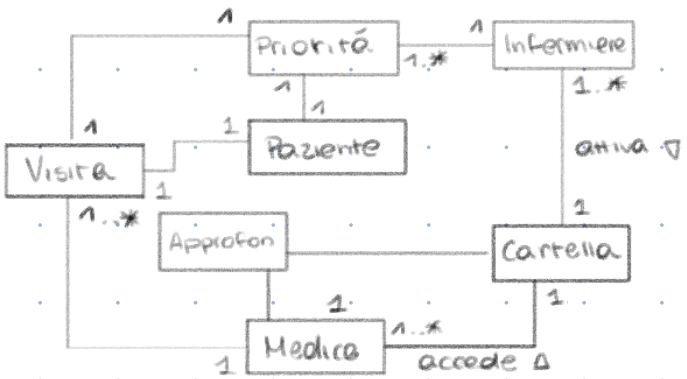
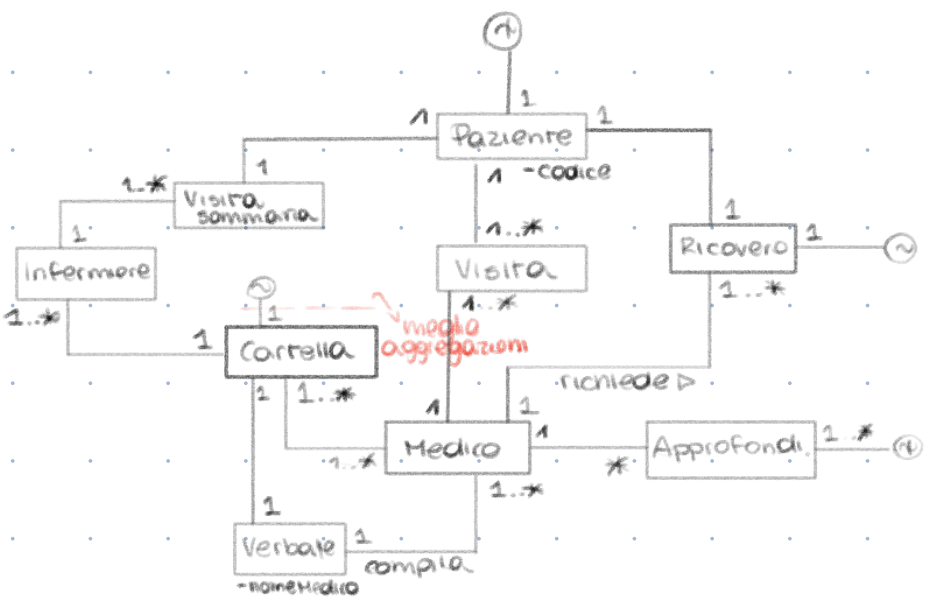
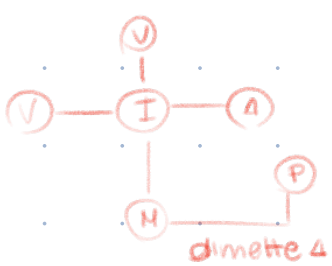
Domain model

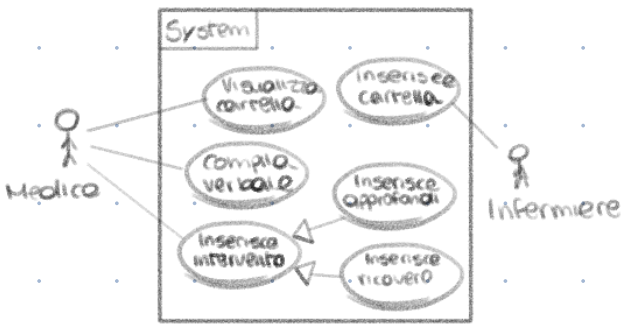


Final model





Use case



UC: Compila verbale
Pre: Medico autenticato al sistema software
Medico visita paziente

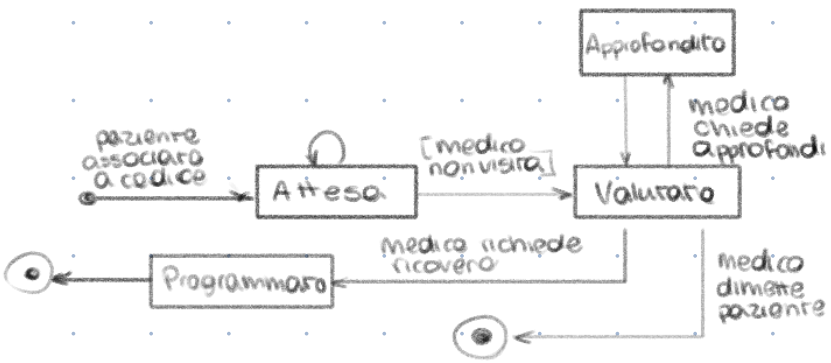
1. Medico seleziona verbale
2. Sistema richiede dati personali del dottore
3. Medico inserisce dati
4. Sistema visualizza nuova finestra
5. Medico inserisce dati del verbale
6. Sistema visualizza verbale inserito

Post: Medico dimette paziente

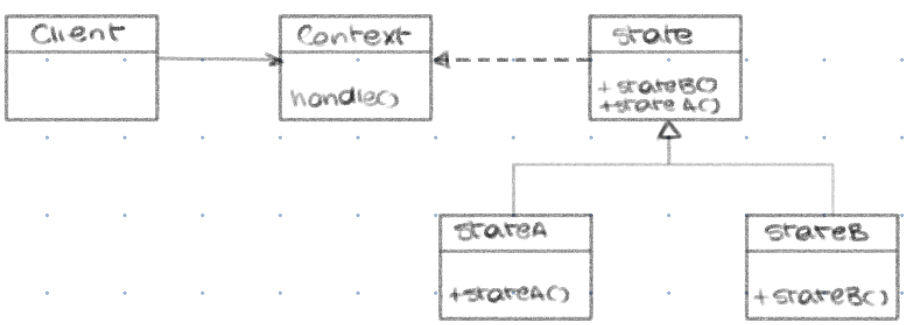
Sequence diagram

state machine diagram

1. Approfondito
2. Attesa
3. Attiva
4. Osservato
5. Compilato
6. Valutato



② State è un pattern comportamentale del catalogo GoF, in totale sintonia rispetto all'intuizione implementativa del meccanismo di composizione, ossia ogni singola classe deve sviluppare un unico asse di cambiamento al suo interno.
Il pattern promuove la possibilità di intercambiare il comportamento di un oggetto, durante run-time, osservando il proprio stato corrente.
Per certi aspetti state è simile al pattern strategy, poiché entrambi permettono intercambiabilità comportamentale, tuttavia vige una differenza sostanziale rispetto al secondo citato, in cui è la classe esterna a stabilire quando debba variare il comportamento, in questa soluzione è l'interfaccia a mantenere un riferimento allo stato corrente e pertanto ad imporre quando cambiare il comportamento.
Proseguendo, a livello implementativo è adottata, come già citato, un'interfaccia, in modo tale che possa essere soddisfatto il DIP e soprattutto automatizzando il processo di intercambiabilità ed i propri comportamenti sono sviluppati da una gerarchia di sottotipi, soddisfando in tale modo LSP, poiché mantenuta compatibilità comportamentale, e OCP, dato che l'estensione di funzionalità si traduce in una nuova entità della gerarchia.



-> handle(), metodo che permette di mantenere sempre attivo il riferimento allo stato corrente dell'oggetto della classe esterna

③ Le user stories rappresenta uno degli strumenti agile più utilizzati all'interno di team di sviluppo, poiché permettono di esprimere requisiti funzionali rispetto ad una differente prospettiva.
L'intento consiste nella descrizione dei vari step che compongono la soluzione finale, come se fossero obiettivi valorizzabili; infatti il proprio utilizzo si compone di tre sostanziali parti, in quanto viene illustrato il beneficio raggiungibile se adeguata e portata a termine una certa volontà.
La struttura è così suddivisa

As I <role>, I do that <goal> as that <benefit>

a prima impatto potrebbero risultare facili da sviluppare, ma la loro complessità cresce man mano alle particolarità tecniche che occorrano all'interno del sistema. Per questa ragione sono spesso adeguati principi di buon uso delle user stories, tra cui alcuni sono:

- indipendenti, ogni user story non deve dipendere da corrispose, pertanto gli obiettivi valorizzabili non possono mai caratterizzare unioni di intenti
- specifiche, ogni user story deve esprimere minime particolarità implementative
- testabili, ogni implementazione dettata da user story deve essere prima testata per poi essere associata alla soluzione finale
- stimabili, ogni user story deve essere stimabile secondo complessità, durata temporale e importanza

Spesso sono adattate anche in altri strumenti agile, come Scrum, racchiuse in un unico backlog, da cui verrà stabilita una stima per ognuna di esse, indirizzando lo sviluppo e soprattutto attribuendo tutto il necessario per favorire la stesura degli sprint backlog.