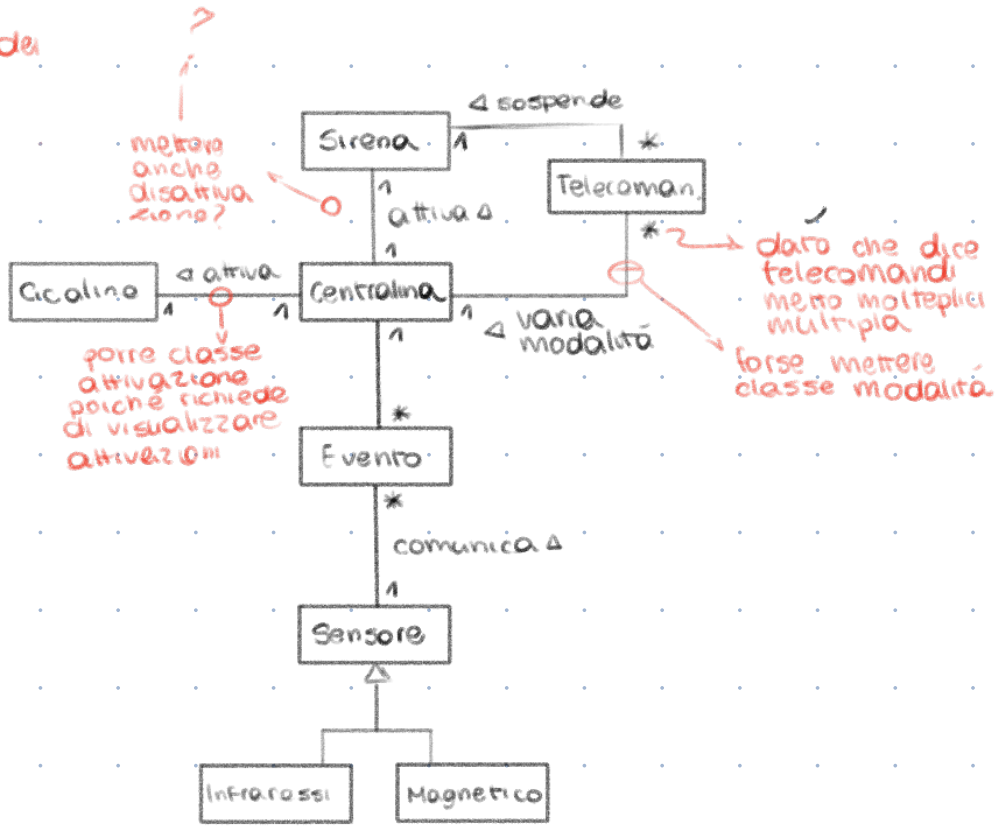
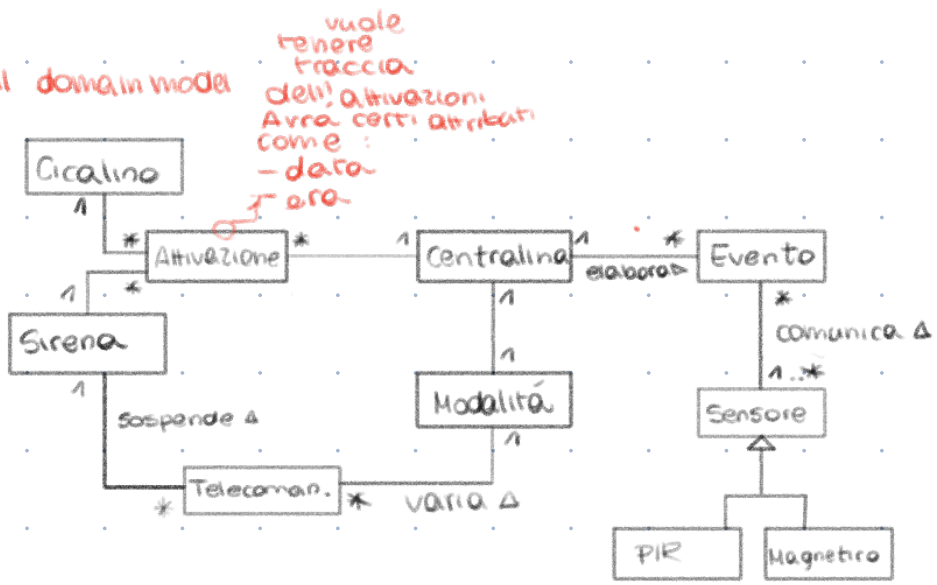
Domain model

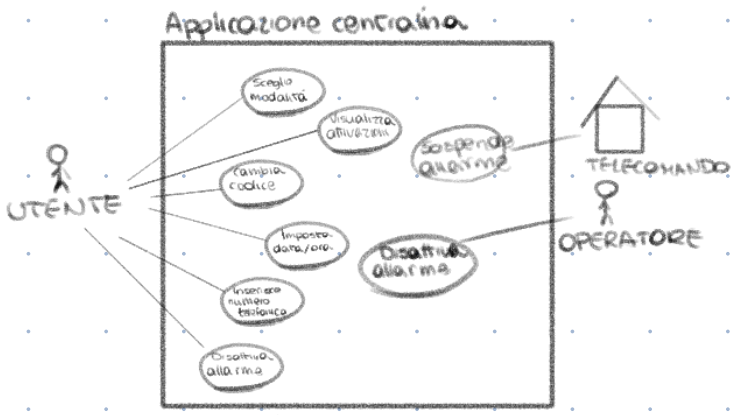


Final domain model



Use case

Ponendo che sia già immesso il codice => Pre.



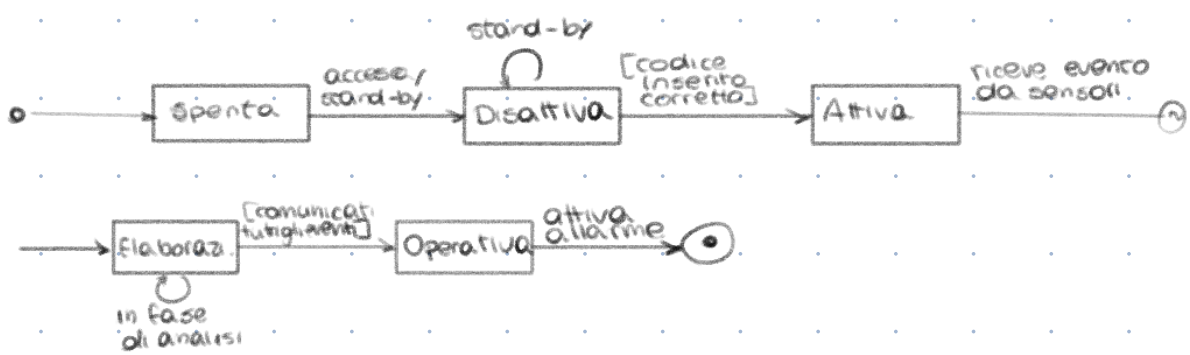
UC: Scegli modalità
Attore: utente
Pre: inserito codice di accesso alla centralina
1. Utente digita tasto per accedere al cambiamento di modalità
2. Sistema visualizza nuovo form di interazione
3. Utente digita la modalità desiderata
4. Sistema visualizza modalità inserita
5. Sistema varia modalità rispetto a quella corrente

Post: variata modalità della centralina

NON FACCIO [sequence]!

State machine diagram

① Disattiva
② Attiva
③ Elaborazione → intesa per l'evento comunicato
④ spenta



② OBSERVER

Observer é un pattern comportamentale, attuato qualora debba essa risolta la pragmatica avversità Notification.
Un problema simile appare qualora si voglia automatizzare l'azione di aggiornamento di una classe osservata rispetto a class osservatrici. La soluzione in tal senso deve rispettare due principi del modello SOLID, il quale garantisce la corretta gestione delle dipendenze, principali artefici della disfatta di differenti sistemi software.
Uno dei due consiste in Dipendancy inversion, il quale ammette che classi software dovrebbero dipendere da astrazioni e non da classi concrete, favorendo in questo modo un layer di schermatura tra high level classes e low level classe, in cui modifiche non abbiano ripercussioni sull'entità corrisposte
Infine, l'ulteriore principle risulta interface segregation, il quale sostiene che l'uso di interfacce dovrebbe essere il più ridotto possibile, evitando un abuso di notazione, poiché potrebbe comportare all'aumento della complessità architetturale
Infatti observer si basa sull'impiego di un'interfaccia sovrapposta tra classe notificante e osservatori, automatizzando le azioni di aggiornamento.
L'entità astratta adotta al suo interno un certo metodo, update(), implementato dalle classi osservatrici ed adoperato dalla classe notificante; quest'ultimo passaggio favorisce l'automatismo richiesto, poiché la classe osservata provvederà a richiamare il metodo update() per ogni observer che componga la propria struttura dove per soddisfare i principi SOLID, sono tutte istanze dell'interfaccia.
In questo modo classi osservatrici potranno ricevere ogni qualvolta si verifichi una modifica una notifica.