Analisi e Progettazione del sistema SmartHome

Alessandro Isceri - 879309

Martina Elli - 886077

Riccardo Ghilotti - 879259

Università degli Studi Milano Bicocca
Dipartimento di Informatica,
Sistemistica e Comunicazione

Corso di Ingegneria del Software
Appello di Gennaio 2024

Indice

Capitolo 1

INTRODUZIONE

In questo documento viene presentato il sistema di gestione domotica SmartHome ed il lavoro di analisi e progettazione svolto.
Mentre l'analisi si basa su una visione più vicina al mondo reale, la progettazione si avvicina di più al framework di iCasa, che abbiamo sfruttato per l'implementazione applicata al simulatore web utilizzato per il testing.

Il lavoro è stato suddiviso in due iterazioni, in cui sono stati effettuati degli *Stand Up meeting* all'inizio e alla fine della maggior parte delle sessioni di lavoro giornaliere, organizzate tramite l'uso del diagramma di Gantt. La maggior parte degli elaborati creati durante le fasi di analisi e progettazione sono il risultato della collaborazione dell'intero gruppo, mentre per la fase di implementazione il gruppo è stato suddiviso per lavorare su due *feature* alla volta: una coppia collaborava in *Pair Programming* e un membro lavorava in singolo con eventuale supporto AI. La composizione dei sottogruppi variava di *feature* in *feature* per favorire una distribuzione equa del lavoro.

Il gruppo ha sfruttato la sua dimensione ridotta massimizzando la comunicazione tra i suoi membri, anche durante le sessioni di lavoro.

Durante lo sviluppo del programma sono stati utilizzati SonarCloud e GitHub Actions per effettuare un controllo automatico di qualità del prodotto. Inoltre, l'uso del *tool* Understand si è mostrato utile ai fini dell'analisi del codice scritto grazie al calcolo delle metriche e del tracciamento delle dipendenze su grafi.
In tal modo è stato possibile effettuare delle sessioni di *refactoring* parallelamente all'implementazione di nuove funzionalità, per mantenere il codice quanto più pulito possibile durante lo sviluppo del progetto.

Per una descrizione del funzionamento del sistema leggere questa guida utente.

Capitolo 2

ITERAZIONE UNO

2.1 Requisiti

I requisiti che sono stati valutati nella prima iterazione sono i seguenti:

- Requisiti funzionali:

    1. Il sistema deve essere in grado di accendere e spegnere le luci in maniera completamente automatica, basandosi sulla percezione della luce naturale in un'area (una stanza o un corridoio) e sulla presenza di residenti.
    2. Il sistema deve essere in grado di regolare automaticamente la temperatura delle aree a seconda dei valori (range) impostati di default.
    3. Il sistema deve essere in grado di valutare il rischio comportato da un aumento di concentrazione di anidride carbonica secondo certe soglie.
    4. Il sistema deve essere in grado di accendere il sistema antincendio in automatico quando percepisce un principio di incendio.
    5. Il sistema deve essere in grado di avvertire i residenti quando rileva un rischio di intossicazione da anidride carbonica.
    6. Il sistema deve essere in grado di allertare l'utente quando rileva un allagamento in un'area.

- Requisiti non funzionali:

    1. La gestione delle luci deve essere volta alla minimizzazione dello spreco di energia (ad esempio, impedendo alle luci di poter essere accese in una stanza vuota e/o sufficientemente illuminata durante il giorno).
    2. La gestione delle luci deve minimizzare l'interazione fisica degli utenti con il sistema (ad esempio, evitando di premere gli interruttori).
    3. Il sistema deve accendere e spegnere le luci istantaneamente.
    4. Il sistema deve allertare rapidamente i residenti in caso di possibili pericoli (incendio, allagamento, intossicazione).

2.1.1 Casi d'uso

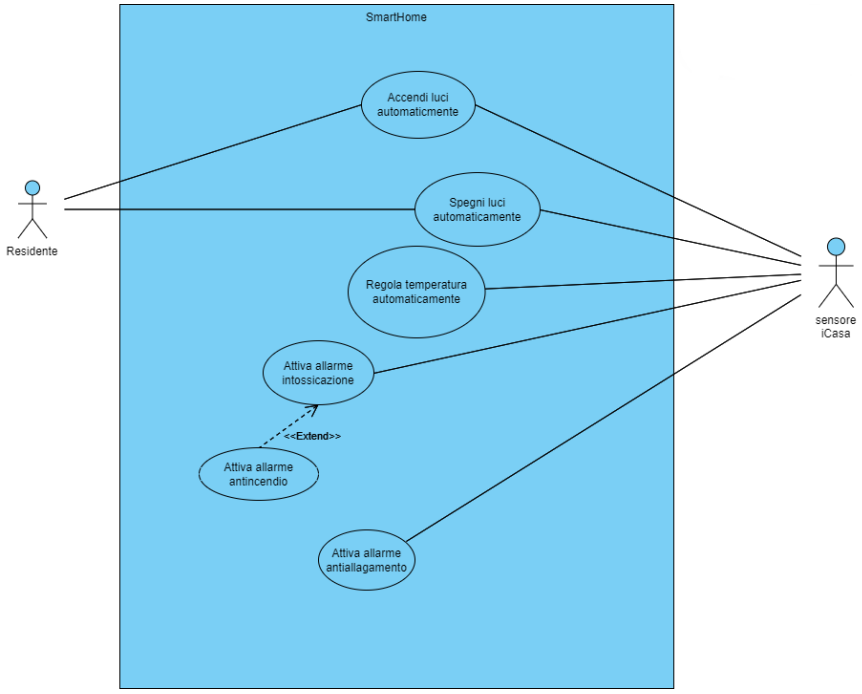

Figura 2.1: Diagramma dei casi d'uso.
Per i primi tre casi d'uso partendo dall'alto Residente è l'attore primario, mentre sensore iCasa è un attore di supporto. Nei restanti casi Residente rimane un attore finale, mentre sensore iCasa è l'attore primario.

Caso d'uso: Accendi luci automaticamente

Scenario principale di successo:

1. Il Residente entra in un'area.
2. Il Sistema nota la sua presenza tramite un Sensore di presenza.
3. Se la luce naturale della stanza non è sufficiente.
   (a) Il Sistema accende le luci.

Caso d'uso: Spegni luci automaticamente

Scenario principale di successo:

1. Il Residente esce dalla stanza.
2. Il Sistema non rileva nessuno nella stanza tramite il Sensore di presenza.
3. Il Sistema spegne le luci.

Caso d'uso: Regola temperatura automaticamente

Scenario principale di successo:

1. Se il controllo della temperatura è acceso.
   (a) Se la temperatura è minore del range indicato.
      i. Il Sistema accende i caloriferi e spegne i condizionatori.
      ii. Il Sistema attende fino a quando non viene raggiunta una temperatura accettabile per il range.
   (b) Se la temperatura è maggiore del range indicato.
      i. Il Sistema accende i condizionatori e spegne i caloriferi.
      ii. Il Sistema attende fino a quando non è un valore accettabile.

   Ritorna al passo 1.

Caso d'uso: Attiva allarme intossicazione

Scenario principale di successo:

1. Un Sensore rileva una variazione della concentrazione di CO2 presente in zona.
2. Se la concentrazione supera una certa soglia ($1 mg/m^3$).
   (a) Il Sistema fa partire una sirena per avvisare i Residenti di una possibile intossicazione nell'area in cui è stata rilevata la variazione della concentrazione.
3. Se la concentrazione supera una certa soglia ($500 mg/m^3$).
   (a) *Attiva allarme antincendio*

2.2 Analisi

2.2.1 Modello di dominio

In questa sezione viene mostrato come è stato derivato il modello di dominio (figura 2.2) a partire dai casi d'uso in forma breve. Principalmente, sono stati considerati in un'Area dei Sensori che rilevano cambiamenti nell'ambiente e dei dispositivi Attuatori che agiscono in risposta a tali cambiamenti. Un'Area può avere più Sensori, ma deve avere solo un Sensore di Presenza e un Fotometro e può avere al più un Rilevatore di CO2 e un Sensore di Allagamento.

Accendi luci automaticamente

Dal caso d'uso Accendi luci automaticamente è possibile identificare le seguenti classi concettuali:

- Luce: la/e luce/i nell'area.
- Area: la sezione della casa in cui si vuole accendere la luce.

- Sensore di presenza: il sensore di presenza nella stanza.
- Fotometro: il sensore che permette di misurare la luce naturale della stanza.

Spegni luci automaticamente

Nel caso d'uso Spegni luci automaticamente non è possibile identificare altre classi concettuali, rispetto al caso precedente.

Regola temperatura automaticamente

Dal caso d'uso Regola temperatura automaticamente è possibile identificare le seguenti classi concettuali:

- Termometro: il termometro nell'area.
- Calorifero: il/i calorifero/i nell'area.
- Condizionatore: il/i condizionatore/i nell'area.

Attiva allarme intossicazione

Dal caso d'uso Attiva allarme intossicazione è possibile identificare le seguenti classi concettuali:

- Rilevatore CO2: il sensore presente nell'area.
- Sirena: la/le sirene nell'area.

Attiva allarme antincendio

Dal caso d'uso Attiva allarme antincendio è possibile identificare le seguenti classi concettuali:

- Sprinkler: lo/gli sprinkler nell'area.

Attiva allarme antiallagamento

Dal caso d'uso Attiva allarme antiallagamento è possibile identificare le seguenti classi concettuali:

- Sensore allagamento: il sensore nell'area.

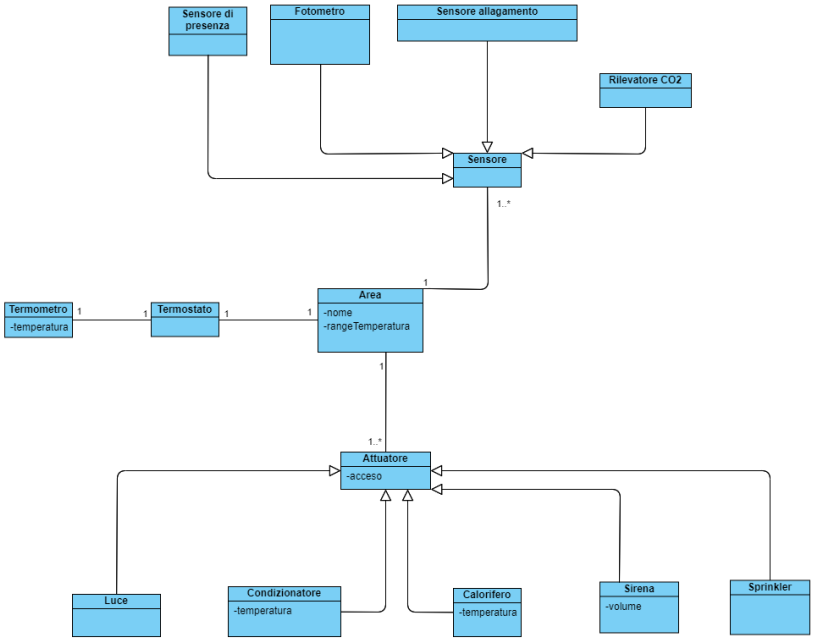

Figura 2.2: Modello di Dominio sviluppato nella seconda iterazione

2.2.2 Diagramma di sequenza di sistema

Si considerino i seguenti casi d'uso:

- Accendi luci automaticamente.
- Spegni luci automaticamente.
- Regola temperatura automaticamente.

Di seguito, sono presentati i relativi diagrammi di sequenza di sistema:

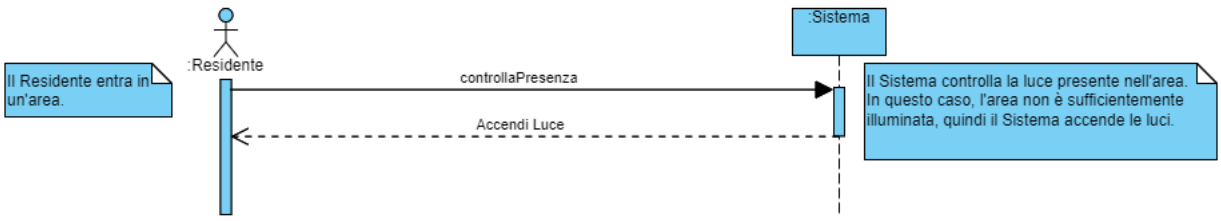

Figura 2.3: SSD Accendi luci automaticamente

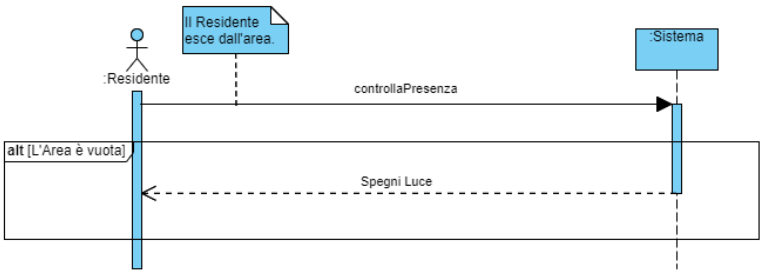

Figura 2.4: SSD Spegni luci automaticamente

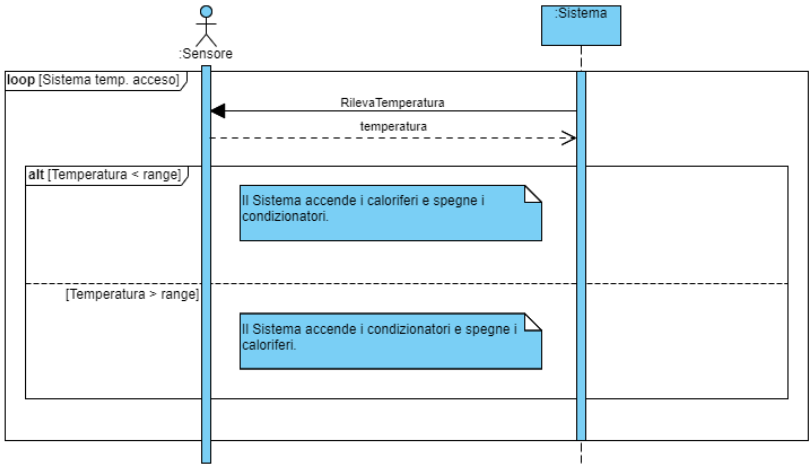

Figura 2.5: SSD Regola temperatura automaticamente

2.3 Progettazione

2.3.1 Diagrammi di sequenza

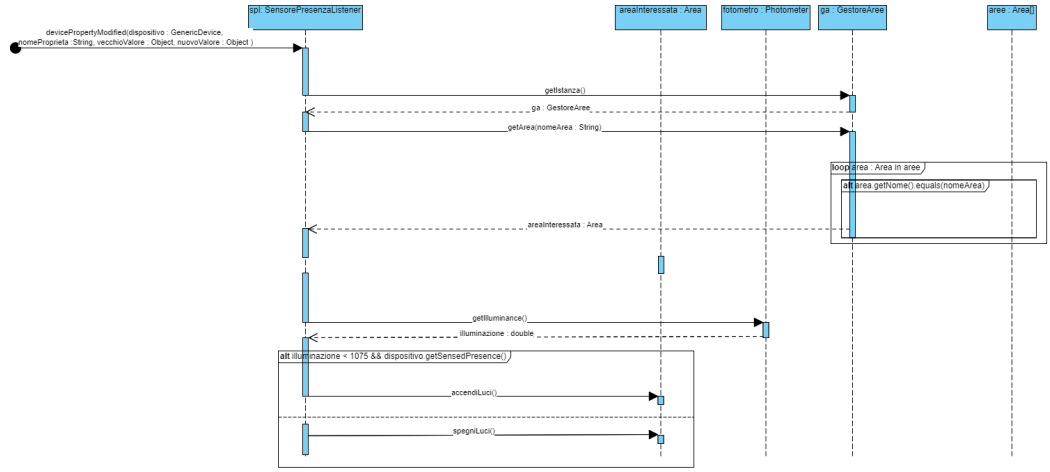

Figura 2.6: *Sequence Diagram* controllaPresenza

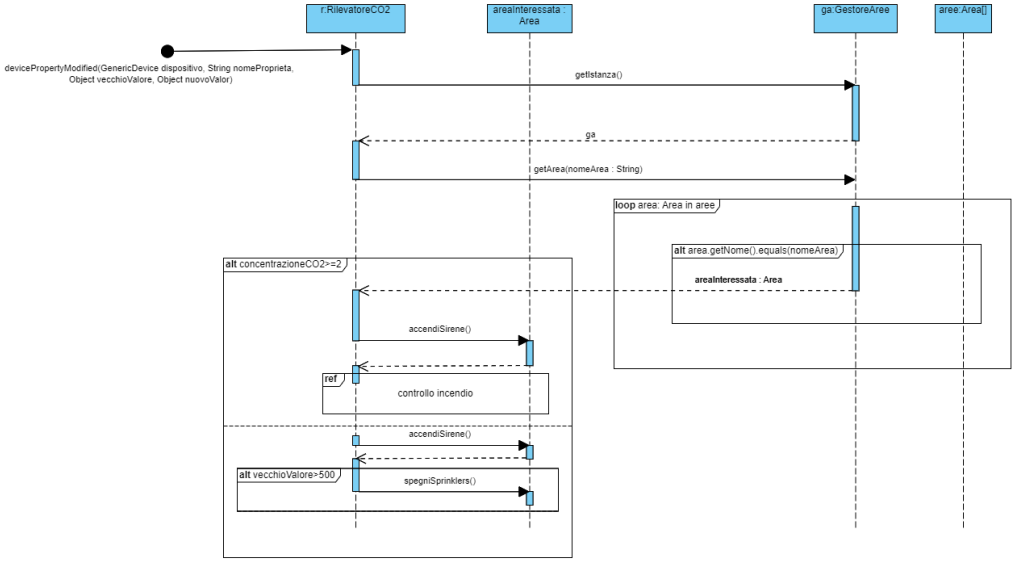

Figura 2.7: *Sequence Diagram* controllaIntossicazione

2.3.2 Diagramma degli stati

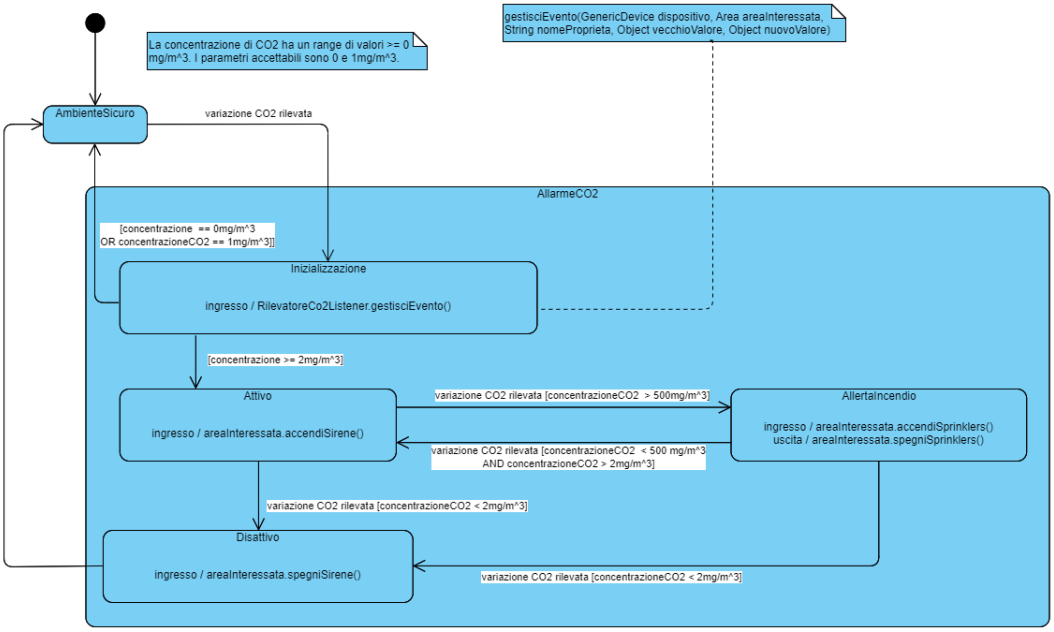

Figura 2.8: Macchina a stati per il sistema di antincendio

2.3.3 Diagramma delle attività

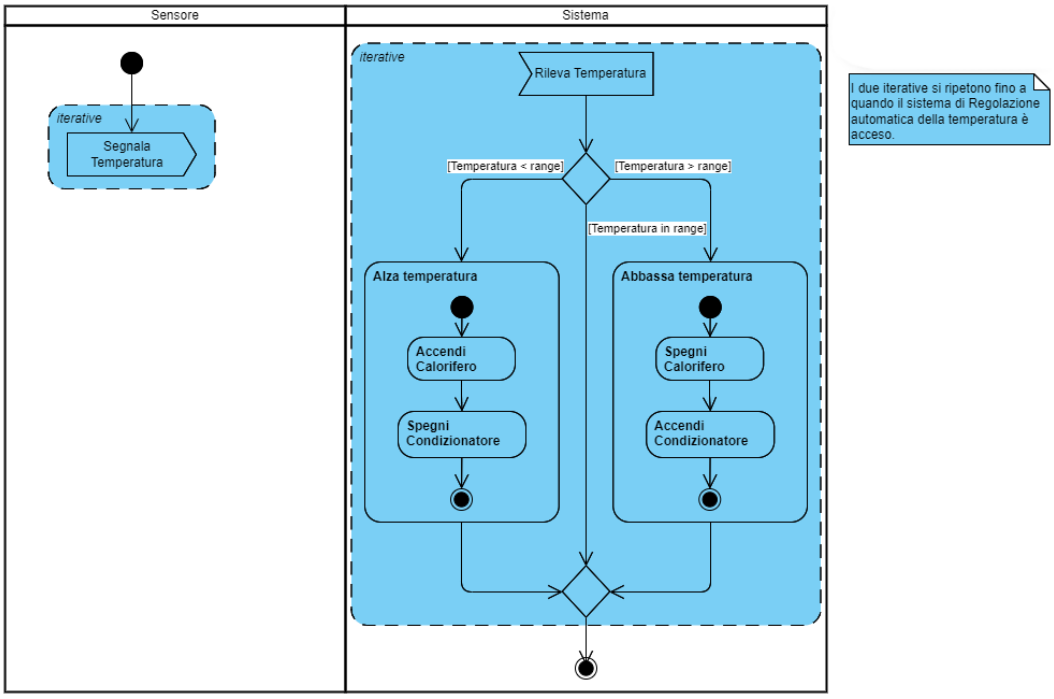

Figura 2.9: Diagramma delle Attività per la regolazione automatica della temperatura

2.3.4 Diagramma delle classi software

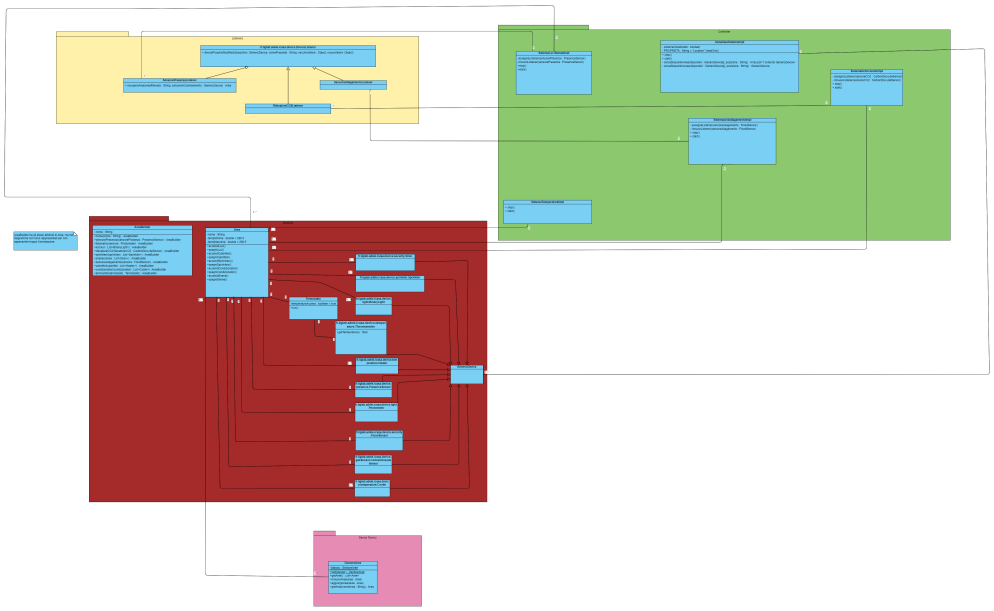

Figura 2.10: Diagramma delle classi software (Se non si dovesse vedere bene, l'immagine è visualizzabile a questo link)

Capitolo 3

ITERAZIONE DUE

3.1 Requisiti

I requisiti valutati nella seconda iterazione sono i seguenti:

- Requisiti funzionali:

    1. Il Residente deve avere la possibilità accendere e spegnere le luci tramite un pulsante.
    2. Il Residente deve poter accendere e spegnere il sistema di gestione automatico della temperatura tramite un pulsante.
    3. Il Residente deve poter accendere e spegnere il sistema di sicurezza tramite un pulsante.
    4. Il Sistema deve attivare l'allarme del sistema di sicurezza quando rileva un'intrusione.

- Requisiti non funzionali:

    1. Il Sistema deve gestire istantaneamente la pressione di un pulsante.

Rispetto all'iterazione uno, il focus è stato su delle *feature* che richiedono all'utente un'interazione diretta con il sistema tramite pulsanti situati nelle varie aree della casa.

3.1.1 Casi d'uso

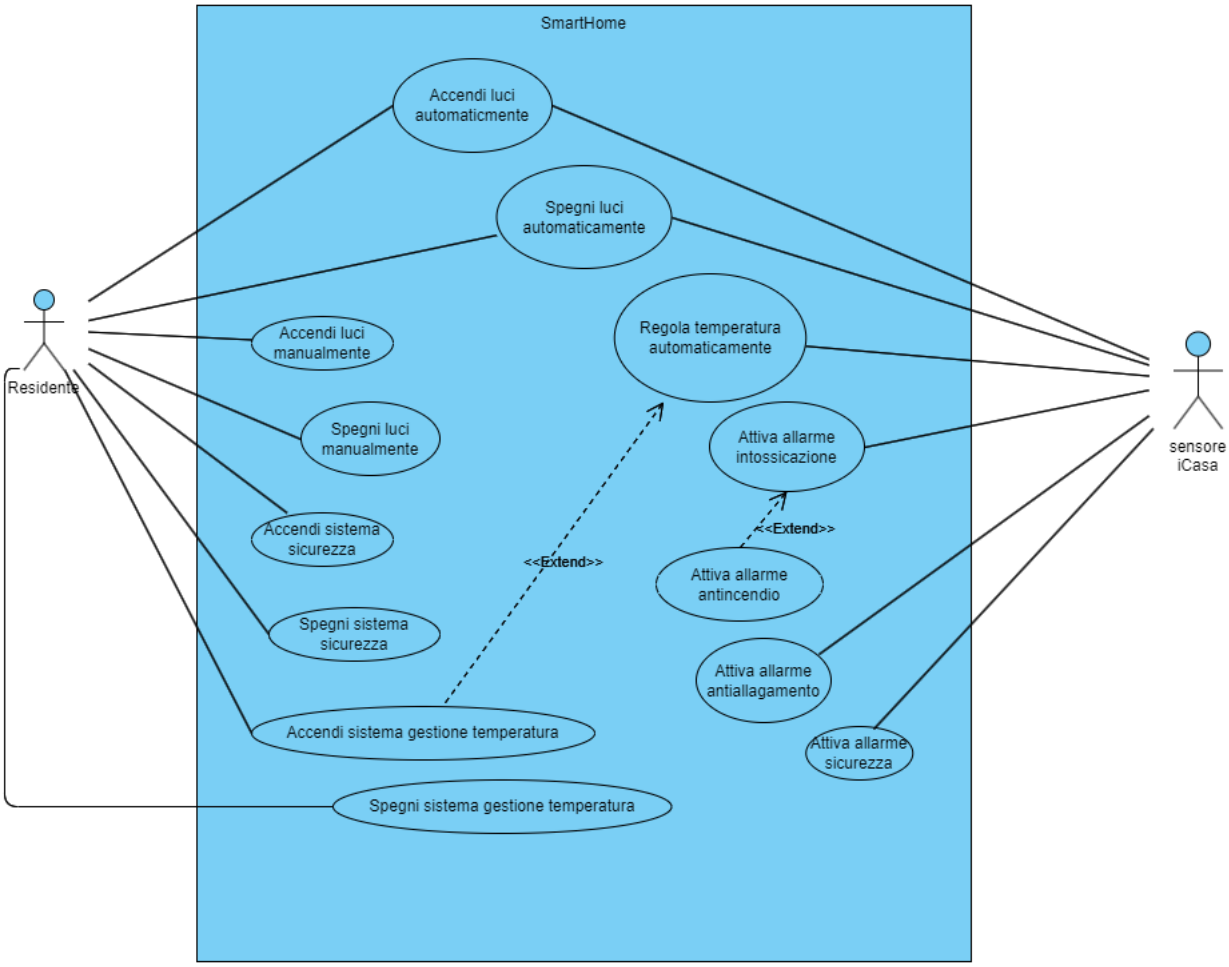

Figura 3.1: Diagrammi dei casi d'uso finale

In aggiunta ai casi d'uso studiati nella prima iterazione, sono stati valutati i seguenti:

Caso d'uso: Accendi luci manualmente

Scenario principale di successo:

1. Il Residente preme il pulsante di accensione delle luci in un'area.
2. Se le luci dell'area sono spente.
   (a) Il Sistema accende le luci dell'area.

Caso d'uso: Spegni luci manualmente

Scenario principale di successo:

1. Il Residente preme un pulsante di spensione delle luci in un'area.
2. Se le luci dell'area sono accese.
   (a) Il Sistema spegne le luci dell'area.

Caso d'uso: Accendi sistema gestione temperatura

Scenario principale di successo:

1. Il Residente preme il pulsante di accendimento del sistema di regolazione della temperatura in un'area.
2. Se il sistema di regolazione della temperatura nell'area è spento.
3. Il Sistema accende il sistema di regolazione della temperatura nell'area.

   Regola temperatura automaticamente.

Caso d'uso: Spegni sistema gestione temperatura

Scenario principale di successo:

1. Il Residente preme il pulsante di accendimento del sistema di regolazione della temperatura in un'area.
2. Se il sistema di regolazione della temperatura nell'area è acceso.
   (a) Spegni il sistema di gestione automatico della temperatura in quell'area.

Caso d'uso: Accendi sistema sicurezza

Scenario principale di successo:

1. Il Residente preme un pulsante di accensione dell'allarme.
2. Se il sistema di sicurezza è spento.
   (a) Il Sistema attiva il sistema di sicurezza.
   (b) Il Sistema notifica l'utente che ha acceso il sistema di sicurezza.

Caso d'uso: Spegni sistema sicurezza

Scenario principale di successo:

1. Il Residente preme un pulsante di accensione dell'allarme.
2. Se il sistema di sicurezza è acceso.
   (a) Il Sistema chiede il codice all'Residente.
   Ripeti passo a. fino a quando non è corretto il codice o fino ad un massimo di 3 volte.
   (b) Se il codice è corretto.
      i. Il Sistema spegne l'allarme.
      ii. Il Sistema spegne il sistema di sicurezza.
      iii. Il Sistema notifica il Residente che il sistema di sicurezza è stato disattivato.

Caso d'uso: Attiva allarme sicurezza

Scenario principale di successo:

1. Il Ladro è entrato in casa.
2. Il Ladro è stato rilevato da un sensore porta/finestra in un'area.
3. Se il sistema di sicurezza è acceso.
   (a) Il Sistema fa partire le sirene nell'area.
   (b) Il Sistema fa partire la registrazione delle telecamere in tutta la casa.
   (c) Il Sistema avvisa i Residenti che qualcuno è entrato nell'area.

3.2 Analisi

3.2.1 Diagramma di sequenza di sistema

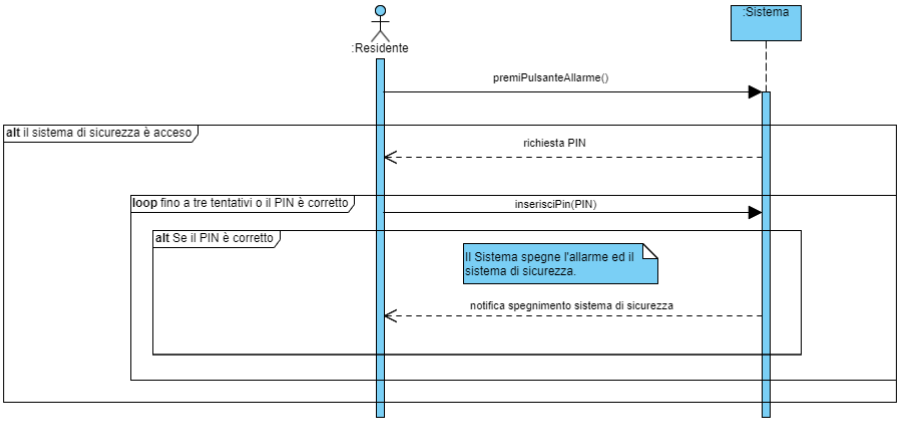

Figura 3.2: SSD per Spegni sistema sicurezza

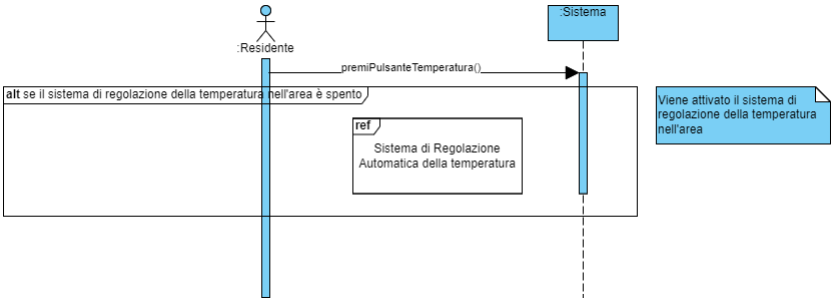

Figura 3.3: SSD per Accendi sistema gestione temperatura

3.2.2 Modello di dominio

Come nella scorsa iterazione, in questa sezione viene mostrato come è stato derivato il modello di dominio (vedi figura 3.4 a partire dai casi d'uso in forma breve.

Accendi luci manualmente

Dal caso d'uso Accendi luci manualmente è possibile identificare le seguenti classi concettuali:

- Pulsante: il pulsante da premere.

Attiva allarme sicurezza

Dal caso d'uso Attiva allarme sicurezza è possibile identificare le seguenti classi concettuali:

- sensore porta-finestra: il sensore che rileva se qualcuno è entrato da una porta/finestra.
- telecamera: la/e telecamera/e presente/i nell'area.

Dai restanti casi d'uso non è possibile ricavare altre classi concettuali.

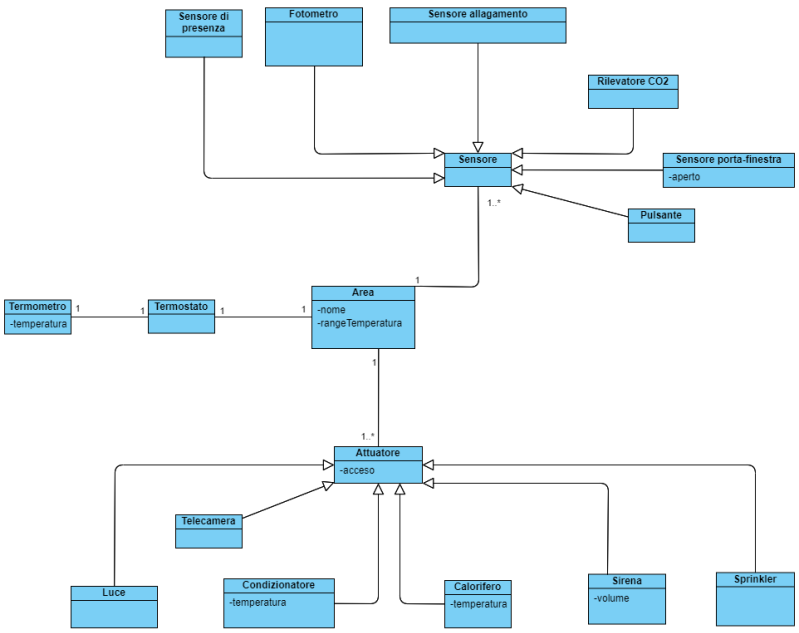

Figura 3.4: Modello di Dominio sviluppato nella seconda iterazione

3.3 Progettazione

3.3.1 Diagrammi di sequenza

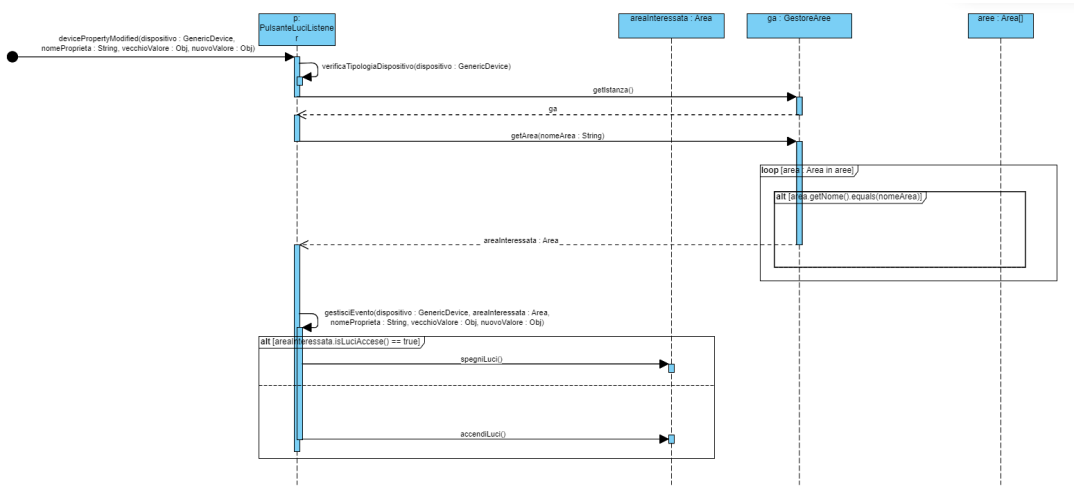

Figura 3.5: *Sequence Diagram* per premiPulsanteLuci

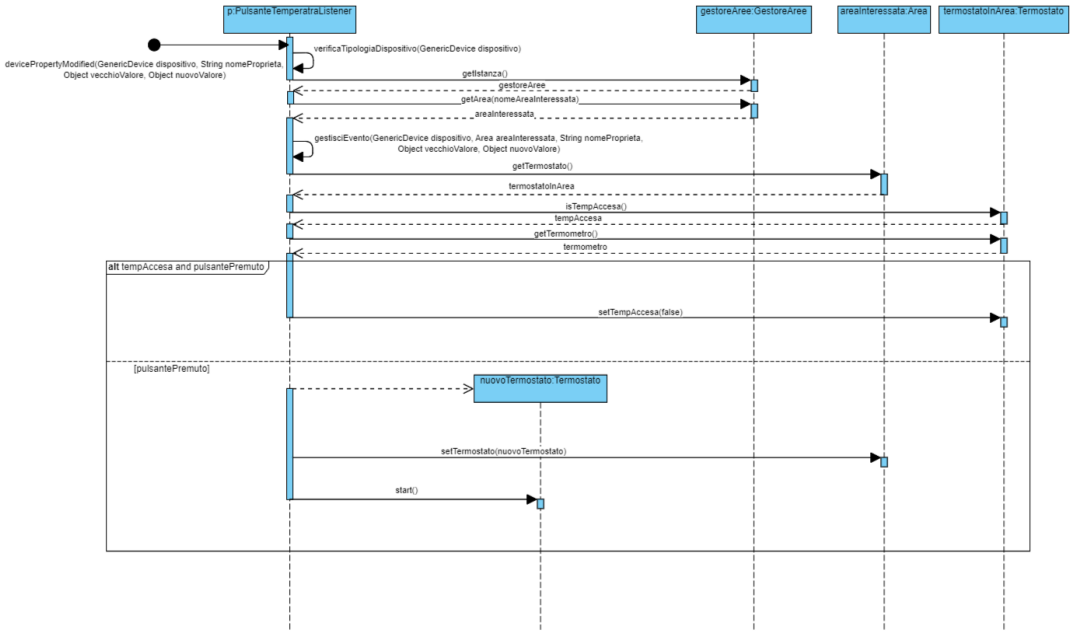

Figura 3.6: *Sequence Diagram* per premiPulsanteTemperatura

3.3.2 Diagramma delle attività

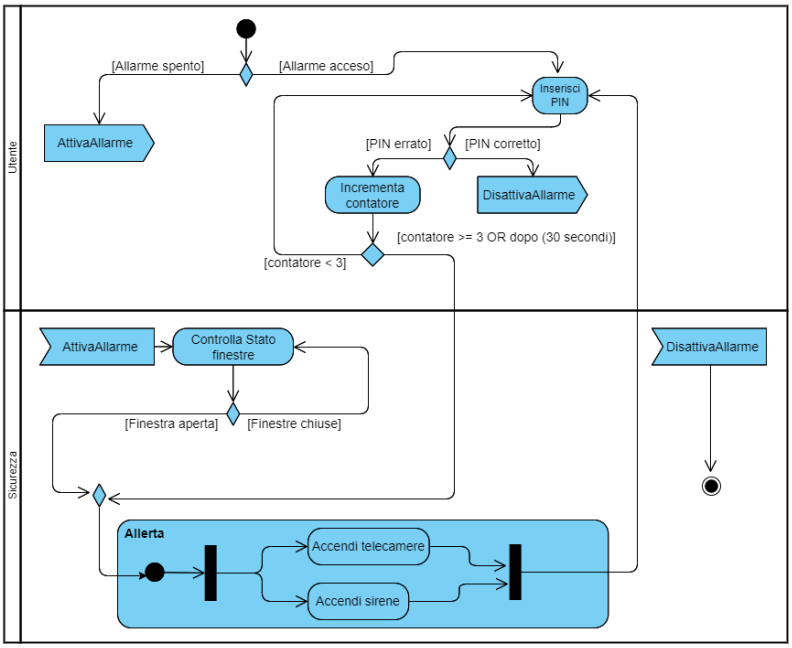

Figura 3.7: Diagramma delle Attività per il sistema di sicurezza

3.3.3 Diagramma delle classi software

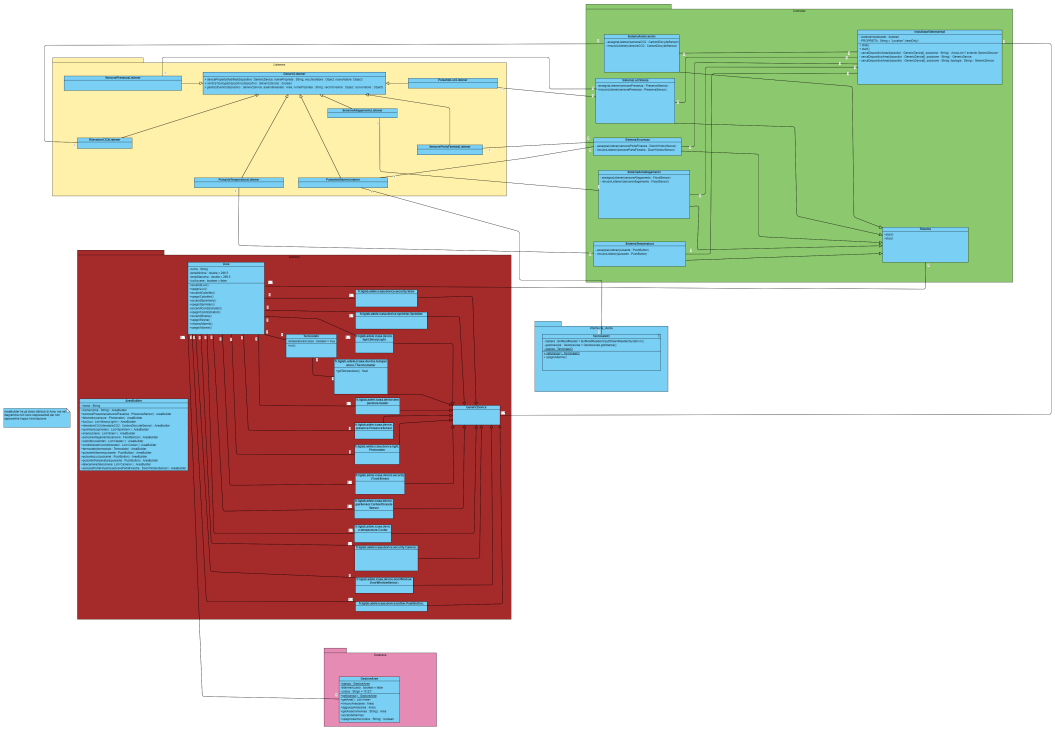

Figura 3.8: Diagramma delle classi software (Se non si dovesse vedere bene, l'immagine è visualizzabile a questo link)

Capitolo 4

CONCLUSIONE

4.1 Design Principles utilizzati

Sono stati applicati i seguenti *Design Principles* di Martin:

- È stato sfruttato il principio **Open Closed**, siccome per inserire una nuova componente non bisogna modificare codice già scritto, ma solamente aggiungerne di nuovo, come si può notare dalla guida all'estensione.
- **Common Closure**: particolarmente evidente nelle classi `Area` e `AreaBuilder`. Infatti appartengono entrambe al package `dominio` e vengono modificate insieme man mano che vengono aggiunti nuovi tipi di dispositivo.
- È stato sfruttato il principio **Acyclic Dependencies** per la maggior parte del codice. Le uniche eccezioni sono le dipendenze reciproche analizzate al punto 4.4.
- Sulla base dei grafi generati da Understand, risulta rispettato il principio di **Stable Dependency**.
- Il principio **Separation of Concerns** è stato sfruttato nei package `controllers` e `listener`, dove ogni Listener ed ogni Sistema ha la propria classe in base alla funzionalità che implementa.

4.2 Design Pattern utilizzati

Sono stati sfruttati i seguenti *Design Pattern GoF*:

- **Builder** per la classe `Area`, in modo da facilitare e standardizzare la creazione di una nuova area.
- **Observer**, sfruttando le interfacce del framework iCasa. Sono state implementate le classi concreteObserver sotto forma di `Listener`, mentre le classi concreteSubject sono già implementate nel framework come `Sensori`.
- **Singleton** nelle due classi `GestoreAree` e `TerminaleIO`, in quanto si necessita di una sola istanza di queste durante l'esecizione.
- **Template Method** grazie alla classe `GenericListener` che definisce lo "scheletro" del metodo `devicePropertyModified`. Invece, le classi che estendono

`GenericListener` forniscono l'implementazione specifica per i metodi astratti `verificaTipologiaDispositivo` e `gestisciEvento`.

- **Controller**: siccome le classi nel package listener gestiscono gli eventi generati dall'utente e inoltrano le richieste alle istanze della classe Area.

4.3 Architectural Patterns utilizzati

- La classe `GestoreAree` costituisce una **Unit Of Work**, in quanto conserva una lista di Aree in modo da gestire l'accesso a tali Aree da parte delle operazioni che le utilizzano come risorsa.
- La classe `Area` è organizzata come un **Transaction Script**, in quanto è formata da operazioni (transazioni) che gestiscono le richieste della presentazione (i listeners).

4.4 Problemi aperti

Antipattern strutturali: La classe Area dà origine ad un external Butterfly. Dipendenze cicliche:

1. tra `Area` e `GestoreAree`, che non risulta particolarmente problematica in quanto viene usata solo per accendere e spegnere le telecamere quando scatta il sistema di allarme. Come possibile soluzione, si potrebbe seguire quanto indicato da Robert C. Martin nel suo libro *"Design Principles and Design Patterns"*, in particolare nella sezione "Breaking a Cycle".
2. tra `Area` e `Termostato`, che è facilmente risolvibile (con Move Method e Move Attribute).

Refactor

Siccome il package controllers non contiene dei veri e propri Controllers come pensato inizialmente allora andrebbe rinominato, un nome più adatto potrebbe essere `init`.

La possibilità di aggiungere sempre più dispositivi di diverso tipo porta a considerare le classi Area e InizializzaSistemaImpl come delle God Class. Per risolvere il problema di Area, si dovrebbe passare ad un altro pattern architetturale, e per InizializzaSistemaImpl si dovrebbero utilizzare delle Extract Method e dei Move Method.