

# Domotic Room Server

Benini - Casadei - Benedetti

University of Bologna

*e.benini5@studio.unibo.it*

April 23, 2016

## 1 Domain Analysis

- Use Cases
- Domain Model

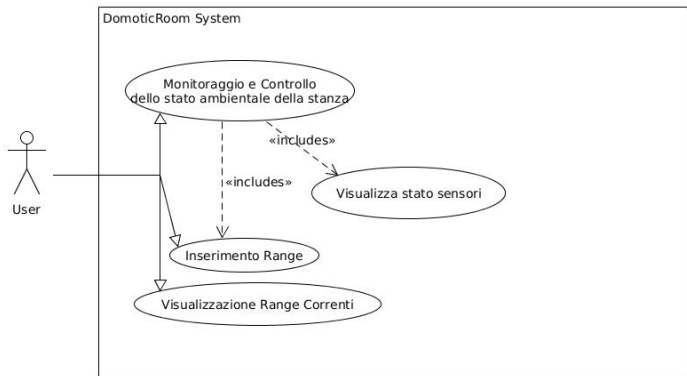
## 2 Problem Analysis

- Abstraction Gap
- Logic Architecture

## 3 Project

- Technologies and Tools
- Project Architecture Differences

## Utilizzo principale della parte server.



Entità principali riguardanti il dominio applicativo

- Sensori: coloro che inviano i dati al server.
- Ranges: Intervalli su cui controllare la validità dei dati raccolti.

Entità dai requisiti non funzionali

- PersistenceStore: Dove salvare i dati.
- Analyzer: computa analisi sui dati.

## Prime considerazioni riguardo l'interazione

- *Polling*: come metodo per ottenere dati dai sensori.
- *Procedure Call*: Viene invocata dall'esterno una funzionalità del sistema.

## Prime considerazioni riguardo il comportamento

- *Wait*: Attesa dell'arrivo di una richiesta.
- *Compute*: Viene effettuata una computazione sulla base della richiesta
- *Output*: Viene mostrato il risultato.

# Abstraction Gap

## Paradigma

L'utilizzo di un'interazione a polling é molto primitivo ed é preferibile, per questioni di design del codice e mantenibilit , passare ad un paradigma che meglio modella il nostro caso di studio, il **paradigma reactive**

## Reattivit 

Le motivazioni che hanno portato alla scelta del paradigma reactive risiedono nelle propriet  che il [?] presenta, che coincidono con i requisiti che vogliamo garantire nel sistema.

## Interoperabilit 

  necessario che il server sia indipendente dal client a meno di un protocollo interoperabile tra loro, di conseguenza si   scelto di effettuare le comunicazioni attraverso il **protocollo HTTP e il formato JSON**.

## Packages:

- *Persistence*: Contiene tutto ciò che riguarda li salvataggio dati.
- *StreamBuilder*: Contiene entità che costruiscono gli steam che trattano i dati in input/output
- *Manager*: effettuano computazioni sui dati, come ad esempio il controllo della coerenza con i range attivi.
- *DataFormatter*: trasformano i dati in ingresso in rappresentazioni interne e viceverse.

Alcuni dei pacchetti presenti nell'architettura logica sono stati omessi visto che, in fase di progetto sono stati sostituiti principalmente per motivi tecnologici e per semplificare l'architettura. Ciò non significa che siano inutili perché hanno contribuito al processo di analisi e possono essere utili come base per altri progetti simili.

# Logic Architecture - Interaction and Behaviour

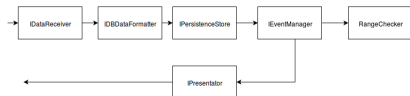


Figure: New Range Stream

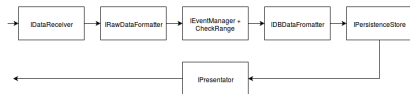


Figure: Incoming Data Stream

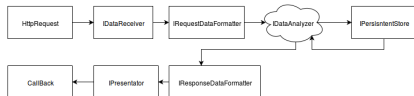


Figure: Request Data Stream

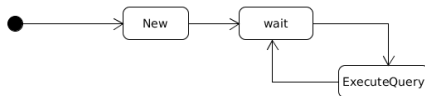


Figure: Persistence Store Behaviour

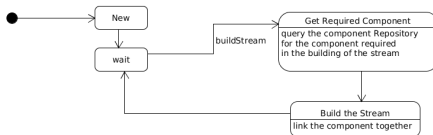


Figure: Stream Builder Behaviour



# Technologies and Tools

# Project Architecture Differences



## Reactive Manifesto

<http://www.reactivemanifesto.org/>