Activity 04
Smart room with agents

Elena Yan
elena.yan@studio.unibo.it

26 dicembre 2022

Indice

1 Analisi del problema **3**

2 Architettura **3**

3 Comportamento **5**

1 Analisi del problema

L'obiettivo del primo punto dell'Activity-04 è quello di progettare e implementare il problema della *smart-room* utilizzando gli agenti Jason all'interno si un sistema multi-agente *JaCaMo*.

JaCaMo[2] è un *framework* che consente di creare sistemi multi-agente e include tre diversi strumenti di programmazione: *Jason* per programmare gli agenti, *CArtAgO* per programmare l'ambiente e *Moise* per programmare l'organizzazione degli agenti all'interno del sistema.

2 Architettura

L'architettura del sistema è composta da un insieme di agenti autonomi e dall'ambiente, come mostrato nella figura 1. L'ambiente include la stanza fisica, i sensori e gli attuatori installati e gli artefatti del sistema. Gli agenti sono entità che incapsulano un flusso di controllo logico e sono progettati per raggiungere uno o più obiettivi specifici, noti come *goal*. Gli agenti sono situati logicamente all'interno di un ambiente (*environment*), in cui essi possono percepire eventi e compiere azioni su di essa.[1] Il sistema è progettato con un'ottica *event-driven*, ovvero gli eventi vengono generati dall'ambiente e percepiti dagli agenti.

Come si può vedere nella figura 1, gli artefatti del sistema sono:

- **LampThingProxyArtifact**, questo artefatto rappresenta un *proxy*, un intermediario tra l'agente `lamp_thing` e il dispositivo fisico delle luci. L'artefatto riceve le istruzioni dall'agente `lamp_thing` sull'accensione o lo spegnimento delle luci e le inoltra al dispositivo, gestendo anche la comunicazione tra l'agente e il dispositivo.
- **LightThingProxyArtifact**, questo artefatto rappresenta un *proxy* tra l'artefatto `light_thing` e il sensore fisico della luminosità. Riceve i valori di luminosità rilevati dal sensore e li inoltra all'agente.
- **PresDetectThingProxyArtifact**, questo artefatto rappresenta un *proxy* tra l'agente `pres_detect_thing` e il sensore fisico per la rilevazione della presenza. L'artefatto riceve le informazioni sulla presenza di utenti nella stanza dal sensore e li inoltra all'agente.

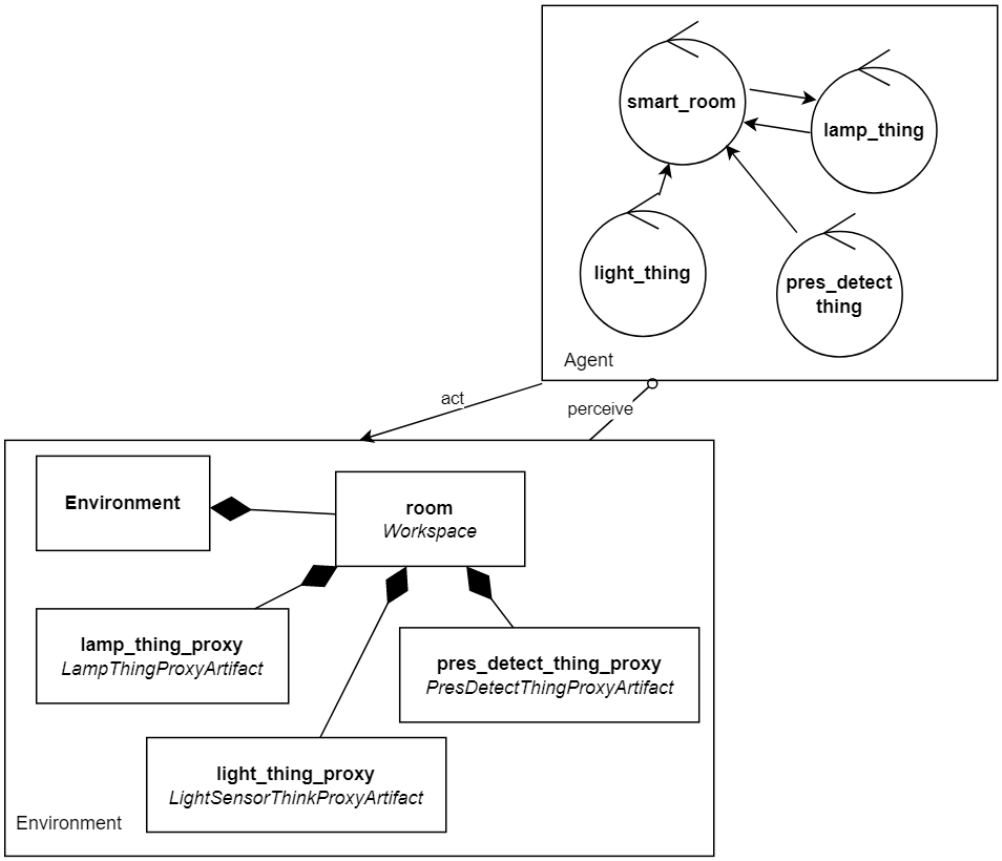

Figura 1: Architettura del sistema

Mentre per quanto riguarda gli agenti, abbiamo:

- **smart_room**, è l'agente che si occupa della gestione della stanza: percepisce i valori generati dai vari dispositivi e in base a quelli, invia la richiesta alle luci di accenderli o spegnerli.
- **lamp_thing**, è l'agente che rappresenta il dispositivo delle luci. Riceve le azioni da compiere dall'agente della `smart_room` e li esegue.
- **light_thing**, è l'agente che rappresenta il sensore di luminosità. Riceve i valori di luminosità dall'artefatto `LightThingProxyArtifact` e li inoltra all'agente `smart_room`.
- **pres_detect_thing**, è l'agente che rappresenta il sensore di rilevazione della presenza di utenti nella stanza. Riceve le informazioni sulla presenza di utenti dall'artefatto `PresDetectThingProxyArtifact` e le inoltra all'agente `smart_room`.

3 Comportamento

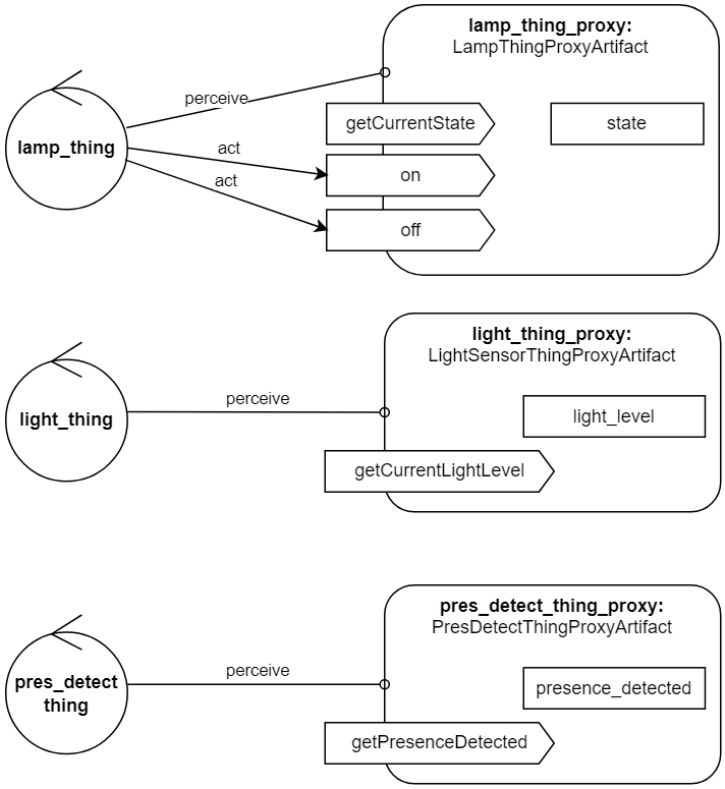

Figura 2: Interazione tra gli artefatti e gli agenti degli dispositivi

Il comportamento del sistema è descritto come segue:

1. I sensori e gli attuatori presenti nella stanza comunicano con gli agenti tramite gli artefatti *proxy*, ovvero `LampThingProxyArtifact`, `LightThingProxyArtifact` e `PresDetectThingProxyArtifact` come illustrato nella figura 2. Ogni artefatto *proxy* contiene una proprietà osservabile che rappresenta lo stato del dispositivo, ossia lo stato della lampada, il livello di luminosità o la presenza di utenti nella stanza.

2. Gli agenti dei sensori e degli attuatori (`lamp_thing`, `light_thing` e `pres_detect_thing`) osservano le proprietà osservabili degli artefatti *proxy* e reagiscono di conseguenza. Ad esempio, se il sensore di presenza rileva la presenza di utente nella stanza, questo viene percepito dall'agente `pres_detect_thing` come una *belief* e invia un messaggio di tipo *achieve* all'agente `smart_room` chiedendo di aggiornare lo stato di presenza. Il messaggio di tipo *achieve* genera un evento che viene aggiunto alla coda di eventi dell'agente `smart_room`.

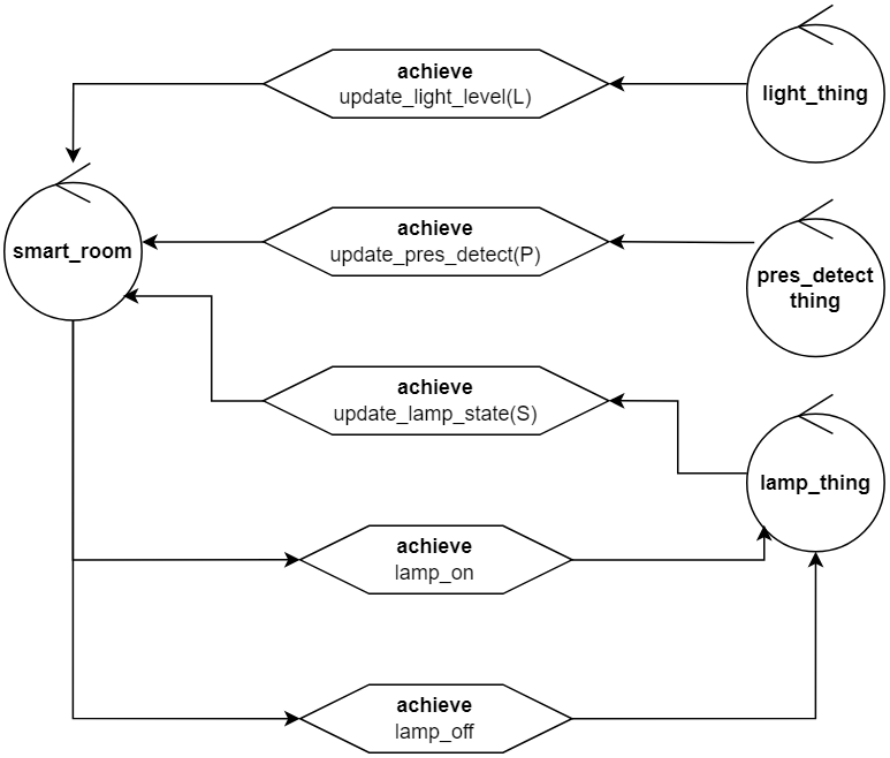

Figura 3: Interazione tra i vari agenti

3. Come illustrato nella figura 3, l'agente `smart_room`, dopo aver aggiornato i valori, li controlla e, se necessario, decide di inviare una richiesta di accensione o spegnimento delle luci all'agente `lamp_thing`.

4. L'agente `lamp_thing` riceve il messaggio di tipo *achieve* dall'agente `smart_room` e, in base al *goal* specificato (ad esempio, `lamp_on` o `lamp_off`), esegue l'accensione o lo spegnimento delle luci.

Riferimenti bibliografici

[1] Olivier Boissier, Rafael H. Bordini, Jomi Hübner, Alessandro Ricci, and Andrea Santi. Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*, 78(6):747–761, June 2013. Special section on Agent-oriented Design Methods and Programming Techniques for Distributed Computing in Dynamic and Complex Environments.

[2] JaCaMo. Jacamo project — multi-agent programming framework. `https://jacamo.sourceforge.net/`. Accessed December 2022.