

DOCUMENTAZIONE CASO DI STUDIO INGEGNERIA DELLA CONOSCENZA  
AA 2024-2025

“AGENDA INTELLIGENTE”

Gruppo di lavoro

· Valeria Agostinacchio, 758264, v.agostinacchio1@studenti.uniba.it

· Chiara de Mauro, 785181, c.demauro7@studenti.uniba.it

Repository GitHub

<https://github.com/Vastrusa/ICON24-25.git>

Introduzione

Questo progetto si concentra sullo sviluppo di un sistema di **pianificazione automatica** per la gestione di un'agenda intelligente, con supporto a **eventi ricorrenti**. L'obiettivo è fornire un sistema in grado di organizzare automaticamente eventi, rispettando vincoli di tempo, priorità e ricorrenza.

Il sistema integra tecniche di **ingegneria della conoscenza**, utilizzando **ontologie OWL, interrogazioni SPARQL** e algoritmi di **AI Planning** per generare piani ottimali.

## **Sommario**

Il sistema implementa un **Knowledge-Based System (KBS)** che integra i seguenti moduli:

* **Rappresentazione della conoscenza** con **ontologie OWL**.
* **Ragionamento automatico** attraverso interrogazioni **SPARQL**.
* **AI Planning** per l'ottimizzazione degli eventi con tecniche di pianificazione automatica.
* **Interfaccia web in Flask** per la gestione degli eventi in modo interattivo.

## **Sezione 1 - Rappresentazione della Conoscenza**

### **Sommario**

La base di conoscenza è costruita utilizzando **OWL** per modellare eventi, utenti, orari e luoghi. La rappresentazione include relazioni tra gli elementi e vincoli di precedenza per garantire una corretta pianificazione.

### **Strumenti utilizzati**

* **Protégé** per la creazione dell'ontologia OWL.
* **rdflib** per manipolare l'ontologia in Python.
* **SPARQL** per interrogare la KB e ottenere eventi e vincoli.

### **Decisioni di Progetto**

* Definizione della classe **EventoRicorrente** per gestire eventi ripetuti.
* Implementazione delle proprietà **haOrario**, **haLuogo**, **haRicorrenza**.
* Modellazione dei vincoli di precedenza tra eventi.

### **Valutazione**

* L'ontologia permette una modellazione flessibile e scalabile.
* Il sistema di query SPARQL garantisce estrazioni rapide di dati.
* Possibili miglioramenti: ottimizzazione delle query per grandi dataset.

## **Sezione 2 - Ragionamento Automatico**

### **Sommario**

Il motore di inferenza utilizza **SPARQL** per estrarre informazioni sugli eventi, verificare la disponibilità degli orari e rispettare i vincoli di precedenza.

### **Strumenti utilizzati**

* **SPARQL** per interrogazioni sulla KB.
* **rdflib (Python)** per manipolare RDF/OWL.

### **Decisioni di Progetto**

* Definizione delle query per:
  + Ottenere eventi di un utente.
  + Verificare eventi prioritari.
  + Identificare orari liberi.
* Ottimizzazione delle query con filtri per migliorare le performance.

### **Valutazione**

* Le query SPARQL permettono un ragionamento efficace sulla KB.
* Il sistema è in grado di verificare la validità di un piano in tempo reale.

## **Sezione 3 - Pianificazione Automatica (AI Planning)**

### **Sommario**

Il sistema utilizza tecniche di AI Planning per generare un piano ottimale degli eventi, rispettando vincoli di tempo e precedenza.

### **Strumenti utilizzati**

* **Fast Downward** per l'esecuzione della pianificazione su problemi PDDL.
* **Algoritmo Greedy** per allocare eventi negli orari migliori.

### **Decisioni di Progetto**

* Implementazione della logica di pianificazione in Python.
* Ordinamento degli eventi per priorità e assegnazione agli orari disponibili.

### **Valutazione**

* Il sistema genera piani validi ed efficienti.
* Possibili estensioni: utilizzo di planner più avanzati per scenari complessi.

## **Conclusioni**

Il progetto ha dimostrato come l'integrazione di **ontologie OWL, SPARQL e AI Planning** possa creare un sistema di **pianificazione automatica** efficace. Possibili sviluppi futuri includono:

* **Notifiche automatiche** per eventi pianificati.
* **Integrazione con Google Calendar** per una gestione avanzata.
* **Supporto a eventi con durata variabile e vincoli più complessi**.

## **Riferimenti Bibliografici**

[1] Protege OWL Ontology Editor - https://protege.stanford.edu/ [2] Fast Downward Planner - https://www.fast-downward.org/ [3] Python rdflib Library - https://rdflib.readthedocs.io/en/stable/