ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

DIPARTIMENTO DI

INGEGNERIA DELL'ENERGIA ELETTRICA E DELL'INFORMAZIONE

"*GUGLIELMO MARCONI*"

CORSO DI LAUREA IN INGEGNERIA BIOMEDICA

TITOLO DELL'ELABORATO

Tecnologie semantiche per il rilevamento, la rappresentazione e la condivisione di competenze digitali

Elaborato in

CALCOLATORI ELETTRONICI

Relatore:

Luca ROFFIA

Presentato da:

Chiara DELVECCHIO

Correlatrice:

Alessandra MARCONI

Gregorio MONARI

Anno Accademico 2022/2023

ABSTRACT

Interoperabilità, semantica e innovazione; questi sono i tre elementi che rendono My2Sec un progetto avanguardista. Il suo obiettivo è semplificare la gestione del lavoro da remoto tenendo traccia in tempo reale dei progressi fatti dal personale lavoratore. Semplifica la pianificazione di progetti e la definizione di obiettivi basandosi su una infrastruttura di tipo *publish/subscribe* data-driven. La tesi si pone l'obiettivo di descrivere la costruzione e la realizzazione dell'architettura semantica dell'applicativo. Racconta inoltre del processo di profilazione e settorializzazione del sistema per rispondere alle esigenze di rappresentazione del settore biomedicale industriale. Attraverso un'indagine sul campo è infatti stato possibile clusterizzare My2Sec fornendogli un caso d'uso concreto.

SOMMARIO

INTRODUZIONE

“Dietro ogni problema si cela un’opportunità”, questo è lo spirito con cui, in piena pandemia nasce VAIMEE. Questa start-up spin-off dell’Università di Bologna ha saputo cogliere la necessità di integrazione, di digitalizzazione e di interoperabilità nel mondo del lavoro, proponendo una soluzione innovativa: My2Sec.

L’obiettivo di questo progetto è realizzare una *suite open source* di supporto pensata per coloro che lavorano da remoto. Attraverso l’applicazione è possibile non solo tenere traccia del tempo speso su ciascuna attività, ma anche definire un profilo attendibile delle competenze di ciascun lavoratore. Per rendere possibile tutto questo, My2Sec si serve di diverse soluzioni *open source* quali: *ActivityWatch* (per tener traccia del tempo dedicato a ciascuna attività), *OpenProject* (per consentire la gestione dei progetti) e *SuperSet* (come piattaforma di Business Intelligence aziendale utile alla valutazione dei KPI).

Gli eventi generati da *ActivityWatch* vengono archiviati localmente sul dispositivo del lavoratore, che potrà scegliere quali dei dati raccolti condividere nel pieno rispetto della normativa GDPR in materia di dati sensibili. Successivamente i dati condivisi vengono elaborati da un algoritmo di intelligenza artificiale per ottenere due importanti informazioni: il tempo speso per ciascuna attività e la descrizione del profilo professionale del lavoratore stesso (**Figura 1**)[1].

My2Sec facilita il processo decisionale all’interno delle aziende semplificando la pianificazione dei progetti e la valutazione dei risultati.

Come annunciato dalla Presidente della Commissione europea, Ursula von der Leyen, nel suo discorso sullo stato dell'Unione 2023[2], il 2023 è l’Anno delle competenze. Il progetto My2Sec si inserisce dunque all’interno di un sistema del valore più grande che ha come obiettivo quello di dotare l’Unione Europea delle competenze necessarie per permettere la crescita e l’innovazione verso un futuro digitale e sostenibile. Investire nello sviluppo delle competenze e nel riconoscimento delle qualifiche è infatti necessario per consentire la crescita economica e sociale nel mercato Europeo[3].

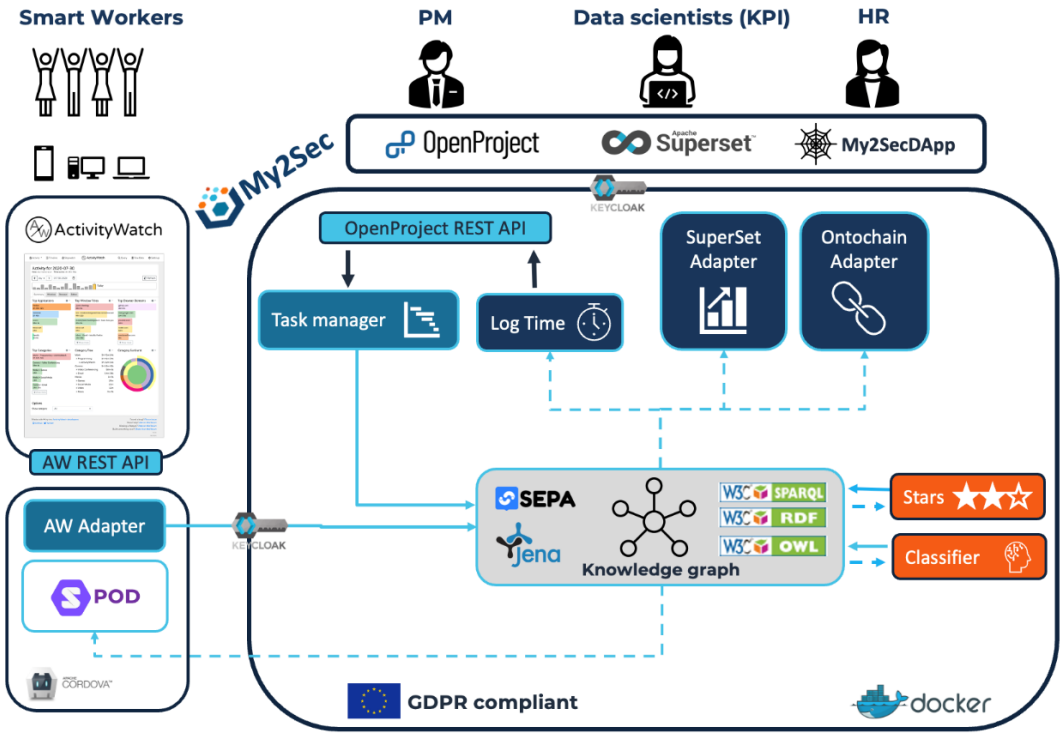


Figura 1: Architettura dell'applicativo My2Sec.

La tesi è la naturale prosecuzione di un tirocinio formativo durante il quale ho avuto la possibilità di approfondire la conoscenza del Web Semantico e dei suoi linguaggi. Nello specifico ho imparato ad utilizzare il linguaggio SWRL[1] per la costruzione di un motore a regole capace di produrre inferenza sulla conoscenza descritta mediante una architettura semantica di tipo ontologico.

La potenza di questo strumento risiede nella possibilità di migliorare la qualità dell'integrazione dei dati, rendendo possibile un'analisi strutturata dell'informazione rappresentata. Dare ordine e struttura alla conoscenza, permette di poter dinamicamente modificare o adattare le relazioni tra i dati in base alle necessità, conservando sempre il valore proprio dell'informazione.

Per comprendere al meglio quale è il valore aggiunto derivante dalla descrizione semantica, mi avvalgo di un esempio pratico. Immaginiamo di essere i proprietari di un grande supermercato e di aver appena ricevuto tutta la merce che avevamo ordinato. Abbiamo due possibili alternative: inserire casualmente la merce negli scaffali senza prestare attenzione al posto in cui viene allocata, oppure definire dei volumi e degli spazi dedicati ai vari tipi di merce e posizionarla in base a tale schema. Osserviamo come la prima opzione renda difficoltosa sia

[1] Semantic Web Rule Language.

l’identificazione della locazione precisa dei beni al compratore, sia la valutazione delle rimanenze al personale addetto. Risulta evidente come la seconda opzione si configuri come l’unica ragionevolmente possibile. Aver stabilito un ordine e delle relazioni permette infatti di poter dedurre evidenze banali come “l’assenza del prodotto nello scaffale a lui dedicato implica l’assenza del prodotto in altri reparti del supermercato”, o ancora “se il prodotto non è presente nel suo scaffale e neanche nel magazzino allora è necessario riordinarlo”.

L’esempio proposto, nella sua semplicità, evidenzia come uno studio accorto delle relazioni che legano gli attori di un ambito applicativo, renda possibile una valutazione strutturata dei dati che da quel contesto si ricavano.

A conclusione del percorso di tirocinio, dopo aver sviluppato l’ontologia e il motore a regole per il progetto My2Sec, è sorta spontanea una domanda di carattere applicativo: quali sono le potenzialità del progetto nei diversi settori lavorativi? Come è possibile ampliare la potenza descrittiva includendo quanti più contesti possibile?

Nel tentativo di rispondere a queste domande ci siamo soffermati soprattutto su quale potesse essere il valore aggiunto dato dall’utilizzo di My2Sec nell’ambito biomedicale. L’elaborato di tesi si configura dunque come il tentativo di settorializzazione dell’applicativo per l’azienda SIMAD operante nel settore della fabbricazione di apparecchiature biomedicali.

Per raggiungere questo obiettivo si è resa necessaria un'indagine sul campo finalizzata ad individuare le esigenze descrittive specifiche che l’applicativo deve soddisfare per poter effettuare una analisi accurata e puntuale dei dati raccolti. In conclusione, questo elaborato si propone di analizzare da un punto di vista semantico l’evoluzione dell’applicativo, illustrando e descrivendo le modifiche che sono state necessarie per permetterne l’utilizzo nel settore biomedicale.

Infine analizzeremo quelli che sono i risultati ottenuti allo scopo di valutarne i limiti e le prospettive di crescita.

CAPITOLO 1 - **Web Semantico e Ontologie**

1.1 Dal Web of links al Web of meaning

La nascita del World Wide Web si deve ai due informatici Tim Berners-Lee e Robert Cailliau che con l'obiettivo di realizzare un sistema utile alla condivisione di materiale scientifico informatizzato, hanno aperto la strada verso la creazione di una "rete mondiale" che raccoglie e collega una moltitudine di contenuti eterogenei. In parallelo alla nascita del WWW[2], si affermano anche un insieme di protocolli e linguaggi utili alla formattazione e visualizzazione dei documenti.

Da questa breve descrizione è possibile immaginare il web come una grande biblioteca di pagine HTML dove però si pone più attenzione alla presentazione del contenuto piuttosto che alla valorizzazione del significato semantico dello stesso. Una prova tangibile di questo limite la riscontriamo quotidianamente quando effettuiamo ricerche attraverso uno qualunque dei motori disponibili. Non tutte le risorse presentate rispondono alla nostra esigenza e viene lasciato all'utente il compito di estrapolare l'informazione dai contenuti presentati[4].

Questo accade perché la maggior parte dei contenuti presenti in rete sono progettati per essere *human readable,* ma non *machine readable*[5].

La soluzione a questo problema richiede una semantica *machine-understandable* per evolvere dal *Web of links* verso il *Web of meaning*[6]. Il Web semantico ha una struttura piramidale dove il significato delle risorse viene specificato man mano che si risalgono gli strati (**Figura 1.1**).

[2] World Wide Web.

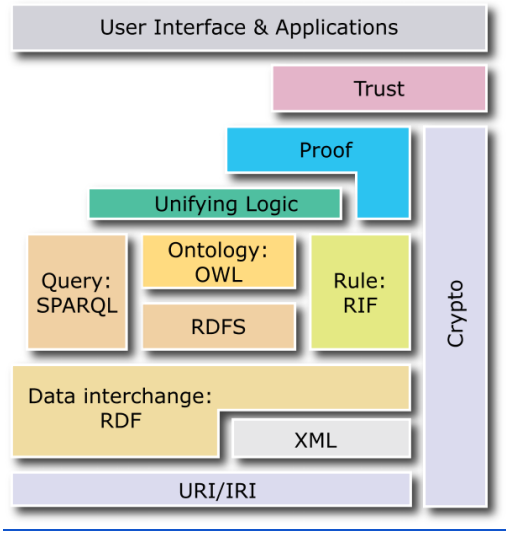


Figura 1.1: Piramide del Web Semantico. Fonte:https://it.wikipedia.org/wiki/Web_semantico.

Il primo livello della piramide è rappresentato dagli URI[3]. Questi sono gli identificatori univoci delle risorse presenti[7]. Al secondo livello troviamo RDF[4]. Questo linguaggio permette di rappresentare l'informazione attraverso triple del tipo soggetto-predicato-oggetto utili a definire le relazioni tra le risorse presenti. Il data model RDF non definisce però un sistema per dichiarare le proprietà e neanche per definire le relazioni tra proprietà e risorse[8]. Questo è infatti il ruolo di RDF *Scheme*. Risalendo ancora la piramide del web semantico troviamo OWL[5]. Questo è il linguaggio utilizzato per rappresentare la conoscenza attraverso modelli ontologici[9]. Sono disponibili editor gratuiti open-source come *Protégé* utili alla realizzazione e gestione delle ontologie.

Come se fosse una sorta di "catena del valore", ogni livello aggiunge progressivamente ricchezza semantica al precedente, favorendo l'integrazione dei dati e l'interconnessione delle risorse.

1.2 ONTOLOGIA

Con l'avvento del Web semantico si presenta la necessità di avere una struttura solida ed ordinata delle risorse intellettuali presenti nella rete. A questo scopo nasce l'ontologia che è una

[3] Uniform Resource Identifier.
[4] Resource Description Framework.
[5] Web Ontology Language.

"*rappresentazione formale, condivisa ed esplicita di una concettualizzazione di un dominio di interesse*"[10]. Possiamo intenderla dunque come una logica descrittiva, o ancora come un sistema per dare ordine alla conoscenza "modellando" la realtà. Questo termine è stato mutuato dalla filosofia dove invece rappresenta "*lo studio dell'essere in quanto tale, nonché delle sue categorie fondamentali*"[11].

Costruire un sistema organico in grado di rappresentare non solo tutta la realtà, ma anche tutte le relazioni tra le entità che la compongono resta un obiettivo ambizioso, soprattutto a causa della velocità con cui mutano i sistemi di informazione.

L'ontologia rappresenta uno strumento tassonomico efficace per ottenere una rappresentazione sistematica delle concettualizzazioni, intese come descrizioni astratte della realtà osservabile. L'ontologia permette di aumentare l'interoperabilità e l'esportabilità dei sistemi, infatti essi possono collaborare facendo riferimento alla stessa ontologia, anche se utilizzano meccanismi differenti per rappresentare l'informazione.

Il primo requisito per realizzare una ontologia è la conoscenza approfondita del dominio che si vuole rappresentare, solo in questa maniera sarà infatti possibile produrre astrazioni rappresentative della realtà, utili all'organizzazione della conoscenza. Poiché le ontologie sono costruite per essere riutilizzate, è inoltre importante valutare se è possibile integrare (*merging*) altre ontologie per ampliare e arricchire la descrizione. In conclusione è fondamentale aggiornare e manutenere l'ontologia affinché risulti allineata e coerente con la realtà che sta descrivendo (**Figura 1.2**).

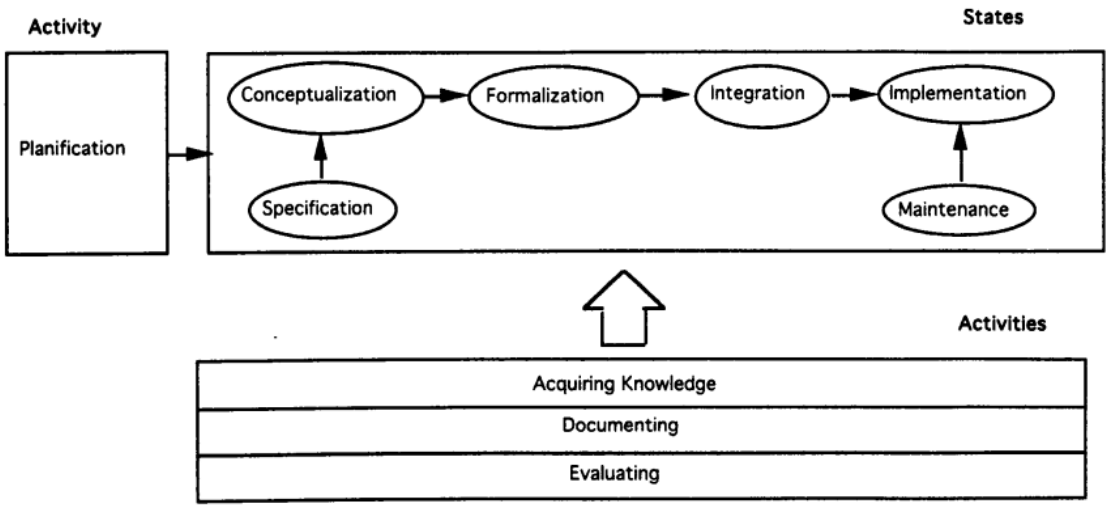


Figura 1.2: Come realizzare una ontologia. Fonte: https://oa.upm.es/5484/1/METHONTOLOGY_.pdf.

Diverse sono le metodologie per costruire un paradigma ontologico. Una possibile strategia prevede un approccio di tipo *top-down*. É infatti possibile ragionare per differenza, scomponendo il dominio di partenza in domini più piccoli che rappresentano parti del tutto. In questo modo si evita di lasciar fuori informazione appartenente al dominio di interesse e si procede andando ad aumentare il livello di specificità in relazione alle necessità descrittive. É altresì possibile adottare una strategia *bottom-up*, che prevede al contrario di partire dalla descrizione delle singole unità specifiche e procedere in verticale raggruppandole in classi via via più ampie sfruttando le caratteristiche comuni tra le entità rappresentate. É possibile in conclusione adottare una strategia che combina le due precedentemente presentate.

Per costruire una ontologia ci serviamo di alcuni elementi fondamentali:

- Classi e sottoclassi;
- Proprietà riferite ad oggetti (Object Properties);
- Attributi (Datatype Properties);
- Assiomi;
- Regole;
- Istanze.

Le classi di una ontologia rappresentano un insieme di oggetti con caratteristiche comuni. Sono organizzate in un sistema gerarchico dove ogni sotto-classe (classe figlia) "eredita" tutte le proprietà e le caratteristiche della rispettiva super-classe (classe padre). Il principio con cui possiamo capire se un oggetto appartiene ad una classe è domandarci se risponde alla domanda "is a". Osserviamo che sono possibili diversi sistemi di classi per descrivere un singolo dominio di interesse, la scelta ricade ovviamente su quello che meglio si adatta alle nostre esigenze di lavoro. La suddivisione in classi e la strutturazione della gerarchia resta uno dei passaggi più complessi della costruzione ontologica, è importante infatti definire bene i confini tra le classi e valutare le relazioni tra le stesse onde evitare incoerenze ed inconsistenze semantiche.

Per definire le relazioni abbiamo due tipologie differenti di proprietà: le *object property* e le *datatype property*. Le prime sono utili a definire le relazioni tra classi di oggetti, legano infatti due entità descritte all'interno dell'ontologia. Le seconde invece sono dei veri e propri attributi in quanto specificano le caratteristiche della classe o dell'istanza analizzata. Queste ultime collegano l'elemento descritto ad un particolare tipo di dato (*literal, dateTime, URI*).

Gli assiomi sono per definizione delle affermazioni sempre verificate. Sono necessari perché ci permettono di: dare dei vincoli all'informazione descritta dall'ontologia, fare inferenze logicamente strutturate e verificarne la correttezza.

Le regole rappresentano un ulteriore strumento per fare inferenza. L'implementazione di motori a regole permette di dedurre conoscenza nuova da quella precedentemente rappresentata sfruttando le relazioni che legano gli oggetti descritti (Capitolo 3).

La più piccola unità rappresentata all'interno dell'ontologia prende il nome di istanza. Rappresenta un singolo elemento di una classe e ne eredita tutte le proprietà.

1.3 Il SEPA ed i suoi strumenti

Il design pattern scelto per l'applicazione My2Sec è il SEPA (SPARQL[6] Event Processing Architecture)[12]. Esso rappresenta un'architettura di tipo *publish/subscribe* progettata per permettere l'interoperabilità e la comunicazione asincrona tra gli agenti che popolano il sistema.

Il SEPA si presenta come un middleware tra endpoint[7] SPARQL ed i suoi client e permette lo sviluppo di applicazioni dinamiche e distribuite. In questa posizione ha infatti l'opportunità di fornire vari servizi ai client per la memorizzazione, la gestione e la navigazione nella conoscenza. Tra i servizi offerti vi è anche la registrazione alle sottoscrizioni. Con questa operazione il client manifesta esplicitamente l'interesse verso una specifica parte della conoscenza memorizzata nel sistema e chiede di venir notificato rispetto alle sue variazioni[13]. La richiesta di sottoscrizione rappresenta quindi lo strumento utilizzato come filtro per individuare la conoscenza d'interesse.

Nella realtà sono due le componenti del SEPA che insieme collaborano per consentire l'elaborazione semantica degli eventi e la gestione delle triple RDF: SEPA Jena e SEPA Blazegraph. Il primo utilizza un framework Java (Apache Jena) utile alla creazione di applicazioni nel contesto del Web Semantico e dei LinkedData, permettendo la creazione di applicazioni data-driven che sfruttano l'elaborazione semantica dell'informazione[14]. Il secondo utilizza Blazegraph. Quest'ultimo è un database a grafo che supporta l'archiviazione e

[6] SPARQL è un linguaggio utilizzato per interrogare la conoscenza rappresentata sotto forma di triple RDF.
[7] Con endpoint intendiamo qualunque applicazione in grado di connettersi ad Internet.

l’interrogazione di triple RDF. La sua caratteristica peculiare è la capacità di ragionare e gestire sistemi contenenti una grande quantità di triple[15].

Il SEPA fornisce agli sviluppatori un modello di progettazione per i client chiamato PAC (Producer Aggregator Consumer). Analizziamo adesso i ruoli e le capacità di ciascun componente del modulo:

- Il produttore rappresenta un generatore di conoscenza, che produce e aggiunge informazione al SEPA, sotto forma di triple RDF, attraverso degli update.
- Il consumatore ha un ruolo duale rispetto a quello del produttore, infatti è interessato unicamente alla raccolta dei dati verso i quali ha mostrato interesse mediante una sottoscrizione (ed è quindi collegato ad una specifica query SPARQL).
- L’aggregatore invece riveste entrambi i ruoli, infatti manifesta l’interesse verso una parte della conoscenza mediante una sottoscrizione, ed ha anche la capacità di inserire nuova conoscenza nel SEPA dopo aver effettuato una elaborazione dei dati ricevuti. Si comporta dunque prima da consumatore e poi da produttore di conoscenza rispetto al sistema.

Il file che viene utilizzato per configurare le applicazioni basate sull’elaborazione di eventi SPARQL è il JSAP (JSON SPARQL Application Profile). All’interno di questo file troviamo tutte le informazioni fondamentali per descrivere l’applicazione, come ad esempio le queries e gli updates utilizzati. Poiché la logica di queste operazioni risulta condivisibile da tutte le applicazioni che sfruttano il SEPA, nello sviluppo di nuovi sistemi potremo riutilizzare lo scheletro del file cambiando i valori coinvolti in base alle esigenze[16].

Nella realizzazione di applicazioni data-driven come My2Sec, l’insieme degli strumenti fino ad ora presentati ci da la possibilità di sfruttare appieno il potere dell’informazione. Definire i ruoli delle applicazioni, capire come comunicano e che dati stanno scambiando ci permette di comprendere meglio il contesto d’origine dell’informazione e come può essere sfruttato per generare un vantaggio competitivo. Il valore aggiunto non è dunque nel sistema quanto nella struttura adottata dal sistema.

CAPITOLO 2 - **Architettura di MY2SEC**

2.1 Metodologia di costruzione dell'ontologia

Come accennato nel Capitolo 1.2, la fase più complessa della costruzione di una ontologia fa riferimento alla scelta degli elementi da rappresentare e alla gerarchia che li collega. Prima ancora di realizzare l'ontologia è stato infatti necessario comprendere: quale sarebbe stata la sua destinazione d'uso, le possibili destinazioni d'uso che avrebbe potuto assumere in futuro e quali erano gli attori e i contesti responsabili della produzione di informazione. In base a questi fattori infatti cambia drasticamente il sistema di priorità con cui si sceglie di strutturare le classi.

Nello specifico la valutazione della porzione di realtà da rappresentare segna i confini dei dati che in seguito si andranno ad analizzare. La politica di rappresentazione che ho scelto di adottare per la costruzione si riassume nell'espressione "*just enough*". Rappresentare solo quello che è strettamente necessario e correlato al contesto limita infatti la possibilità di generare confusione tra gli elementi descritti.

Definiti i confini dell'applicazione ed individuati gli attori principali del flusso operativo, ho cercato di definire le loro caratteristiche rifacendomi concettualmente alla regola anglosassone delle *Five Ws*[8]. Questa fase è stata molto utile per sintetizzare in maniera concisa ed essenziale le caratteristiche peculiari di ogni elemento. Conseguentemente mi sono domandata come questi attori interagiscono tra loro e quali informazioni scambiano, chiedendomi di volta in volta quale avrebbe potuto essere lo strumento semantico più adatto all'esigenza descrittiva specifica. Infine ho valutato la possibilità di integrare e collegare l'ontologia ad altre esistenti con l'obiettivo di standardizzare le definizioni dei concetti comuni.

[8] Le 5W stanno per: Who, What, When, Where e Why. La regola, nata nel contesto giornalistico anglosassone, rappresenta un efficace strumento di problem solving utile nella pianificazione dei processi[17].

Per realizzare l'ontologia è stato utilizzato *Protégé*[18], un software open source basato su Java sviluppato dal BMIR[9] presso la Scuola di Medicina dell'Università di Stanford. *Protégé* permette di creare, memorizzare, visualizzare e gestire soluzioni knoelwdge-based.

Un aspetto molto positivo relativo all'utilizzo di questa applicazione risiede nella possibilità di installare numerosi plug-in come ad esempio OntoGraf[19] e SWRLTab[20]. Nello specifico il primo permette di visualizzare in maniera precisa ed intuitiva la struttura dell'ontologia, mentre il secondo fornisce un ambiente di sviluppo per scrivere le regole SWRL (Capitolo 3) ed interrogare la conoscenza con le queries SQWRL.

2.2 Progettazione e costruzione dell'ontologia 2.0

Come abbiamo già anticipato, l'obiettivo di My2Sec è aiutare gli smart workers a monitorare le attività svolte sulle varie piattaforme software. L'ontologia in questo contesto ha lo scopo di rappresentare gli attori del contesto lavorativo ed i sistemi coinvolti nel monitoraggio; e di organizzare strutturalmente le relazioni che li legano.

Il flusso di lavoro che caratterizza l'applicativo (**Figura 2.3**) vede i lavoratori di una azienda (*Member*) che collaborano insieme alla realizzazione di progetti (*Project*). I singoli progetti sono formati da un insieme di task (*Task*), ed i singoli task possono essere suddivisi in sotto-task (*SubTask*) (**Figura 2.1**).

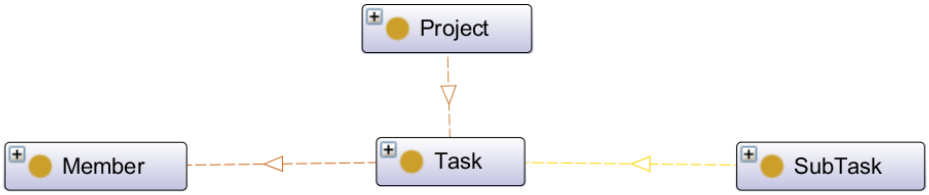


Figura 2.1: Dettaglio del grafo relativo all'ontologia di My2Sec.

Durante la loro normale giornata lavorativa, i *Member* producono eventi (*Event*) che sono caratterizzati dal timestamp (istante in cui l'evento è stato generato), dall'apptitle (titolo dell'applicazione da cui l'evento è stato generato) e dall'ActivityType (il tipo di attività associato all'evento specifico). Gli eventi vengono successivamente aggregati in attività

[9] Stanford Center for Biomedical Informatics Research.

(*Activity*). Queste ultime rappresentano la quantità di tempo, intesa come durata, spesa su uno specifico incarico (**Figura 2.2**).

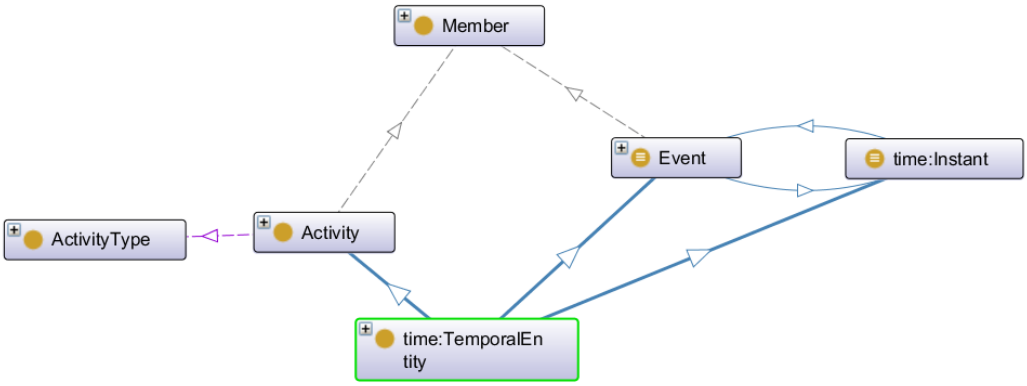


Figura 2.2: Dettaglio del grafo relativo all'ontologia di My2Sec.

Come è possibile osservare dalla Figura 2.2, abbiamo collegato l'ontologia di My2Sec alla Time Ontology[21] rispetto alla definizione delle istanze temporali. Previa verifica di conformità rispetto al livello gerarchico e alla definizione formale delle due classi, abbiamo utilizzando relazioni di equivalenza e di subordinazione per collegare il concetto di *Event* (my2secOntology) a quello di *Istant* (timeOntology), ed inoltre abbiamo potuto considerare tutte le entità temporali come sottoclassi di *TemporalEntity* (timeOntology).

È stato inoltre possibile utilizzare la FOAF[10] Ontology[22] per definire la classe *Member*. Questa ontologia è nata infatti per descrivere le persone e le relazioni che le collegano principalmente nel mondo virtuale.

[10] FOAF è l'acronimo di Friend Of A Friend.

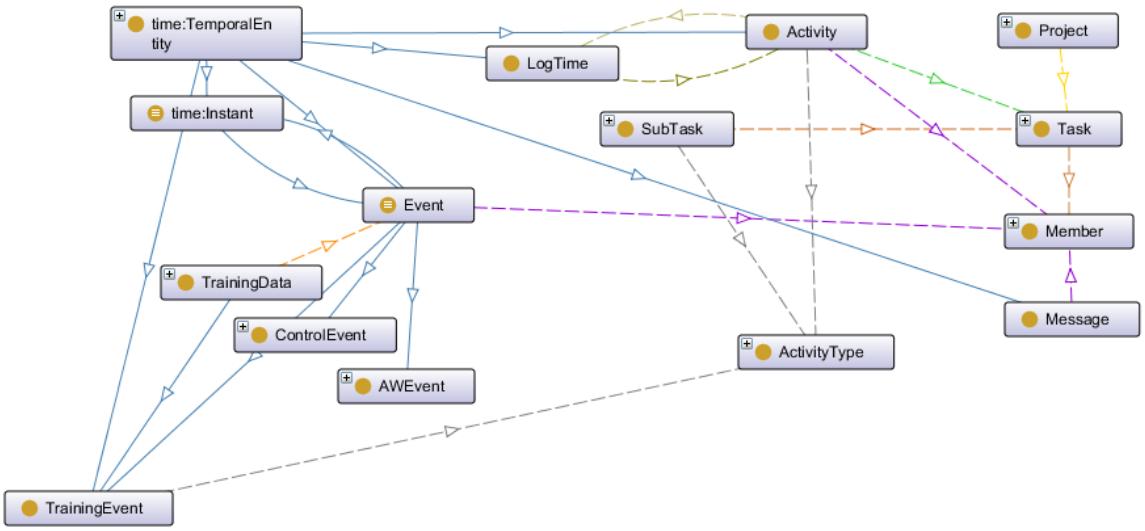


Figura 2.3: Grafo completo relativo all'ontologia di My2Sec.

Con una strategia di tipo *top-down* abbiamo ultimato la descrizione dell'ontologia definendo le sottoclassi e le istanze delle singole classi (**Figura 2.4**). Osserviamo che è presente anche una classe relativa agli eventi e ai dati di Training (*TrainingEvent* e *TrainingData*). È stato necessario inserire questa classe per tener conto di quelle informazioni memorizzate nel sistema utili ad allenare l'intelligenza artificiale nell'associazione dell'*ActivityType*. È possibile trovare maggiore dettaglio riguardo la definizione delle entità ontologiche nell'**Allegato I**.

In questo momento è importante sottolineare un aspetto inerente alla scelta degli *ActivityType* rappresentati. Per mantenere coerenza rispetto agli eventi realmente elaborati dall'applicazione e nel rispetto della filosofia di costruzione "*just enough*", ho scelto di inserire gli *ActivityType* relativi esclusivamente ai dati su cui l'applicativo veniva testato, strutturando l'ontologia in modo tale da poter essere ampliata in futuro senza perdere di coerenza con quanto descritto. Mantenere elastica l'architettura permettendo un ampliamento modulare è stato un requisito da subito essenziale per non porre limiti alla crescita dell'applicativo.

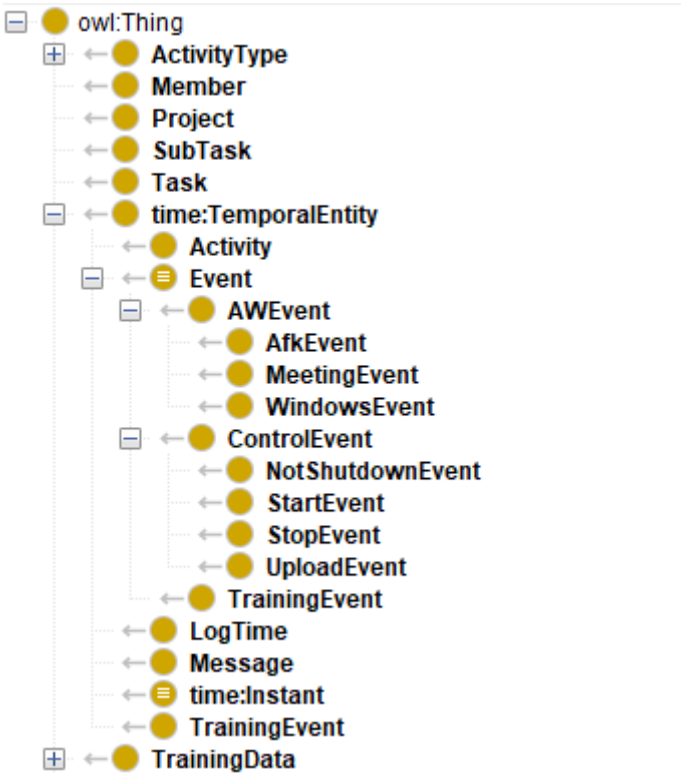


Figura 2.4: Elenco completo delle classi dell'ontologia di My2Sec.

La sfida posta a questo punto dello sviluppo dell'ontologia riguarda l'associazione tra l'attività svolta ed il task a cui appartiene. Ogni lavoratore infatti dedica il suo tempo a diverse attività nell'arco della stessa giornata, ognuna finalizzata al completamento di un task specifico. La conoscenza del tempo speso su ciascun task rappresenta una informazione molto importante soprattutto per effettuare delle valutazioni prospettiche da un punto di vista economico ed organizzativo. Questo genere di informazione va costruita a partire dai dati raccolti, non viene fornita in automatico da nessun sistema connesso all'applicativo. Vedremo come realizzare questo collegamento nel prossimo capitolo (Capitolo 3).

2.3 Modulo PAC My2Sec

Soffermiamoci adesso sull'analisi del flusso dei dati all'interno dell'applicativo (**Figura 2.5**). Come anticipato nel Capitolo 1.3, il SEPA interagisce con una moltitudine di moduli PAC per acquisire, elaborare e visualizzare l'informazione memorizzata.

Una volta che i *Member* si sono registrati nel sistema, il Project Manager può procedere con la creazione dei *Task*. Questi ultimi vengono caricati nel SEPA e rappresentano i dati a

disposizione del *TaskAI consumer* e delle applicazioni di Frontend. Queste azioni caratterizzano la fase iniziale di definizione del progetto e di ripartizione dei compiti nel contesto aziendale. I lavoratori, durante la loro giornata lavorativa, producono degli eventi che vengono memorizzati in locale da *ActivityWatch*. Con cadenza regolare il lavoratore carica i dati prodotti sul SEPA attivando il *AwMapper*. Prima che i dati vengano memorizzati nel SEPA è infatti necessario effettuare una trasformazione degli stessi da eventi memorizzati in un file json in triple semantiche. In questo momento i dati caricati nel SEPA vengono consumati da un classificatore che utilizza l'intelligenza artificiale per due principali scopi: ricavare dalla moltitudine di eventi a disposizione le attività, ed assegnare l'*ActivityTyp*e corretto alla specifica attività. L'intelligenza artificiale rappresenta lo strumento più idoneo per questa operazione per la sua capacità adattiva. È infatti capace di migliorare le sue capacità predittive apprendendo dai dati in modo autonomo. Le predizioni fatte dall'*ActivityTypeAI consumer* vengono inviate all'utente che le ha generate per essere validate. L'utente dunque valida le attività attraverso un consumatore e inserisce i dati corretti nel SEPA. L'*ActivityTypeAI consume*r carica i dati nel suo database interno di conoscenza. Per fare questa operazione i dati vengono trasferiti dal SEPA-Jena al SEPA-Blazegraph attraverso un bridge creando la cronologia dei dati dell'utente. Questa viene copiata in un database SQL ed utilizzata per realizzare, con SuperSet, delle Dashboard di visualizzazione dei dati. In conclusione il *TaskAI aggregator* (Capitolo 5) associa alle attività validate il task corrispettivo e calcola il tempo speso (*LogTime*) su ciascuna attività. Infine anche questi dati vengono inviati al SEPA. Nello specifico i *LogTime* calcolati vengono inviati alla *OpenProjectAPI* che calcola lo *SpentTime* complessivo del task e memorizza anche questa informazione nel SEPA.

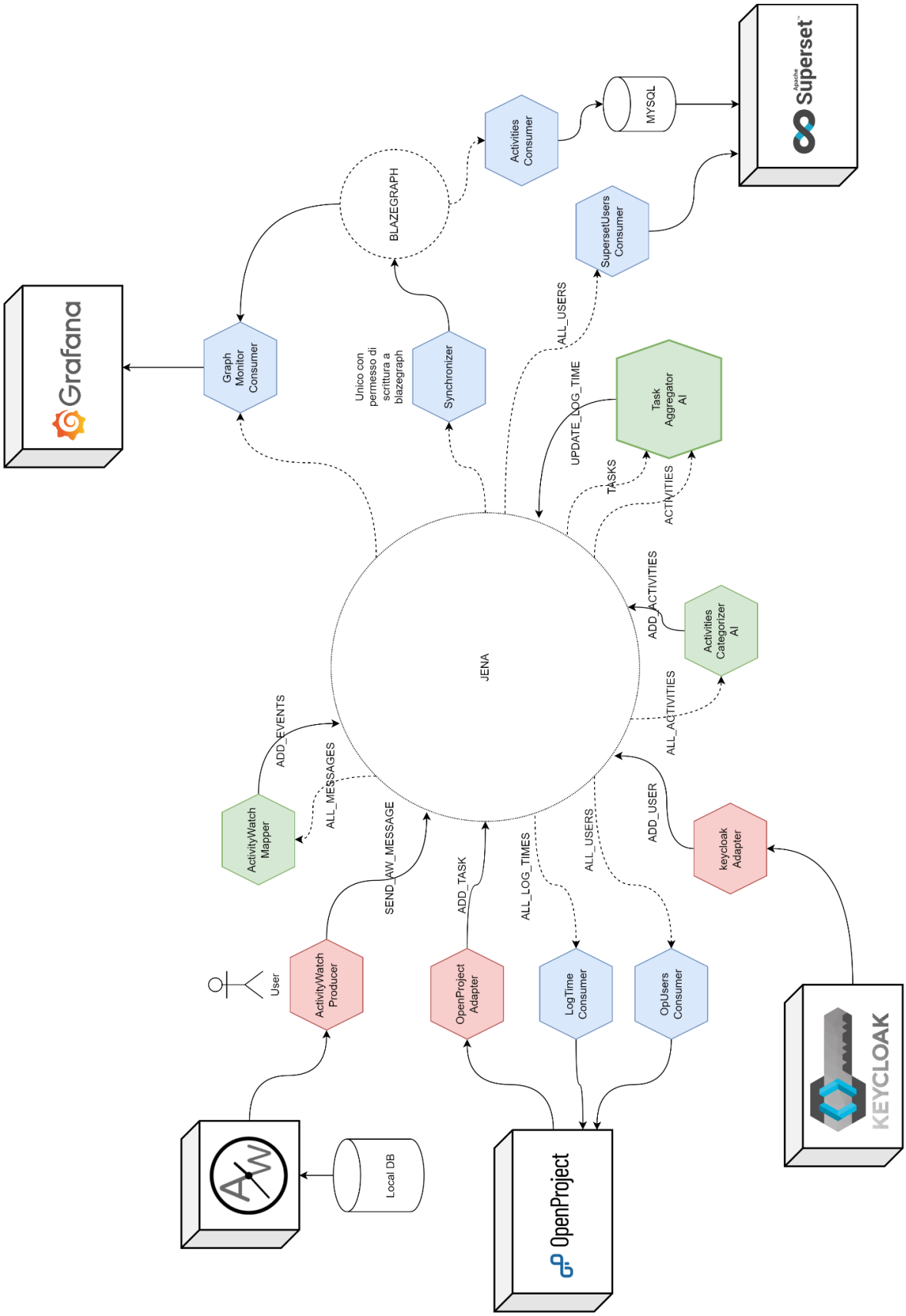


Figura 2.5: Modulo PAC di My2Sec.

CAPITOLO 3 - **Rules**

Come accennato nel capitolo precedente, il problema relativo all'associazione tra *Activity* e *Task* non è di banale risoluzione. Non è infatti possibile risalire a quale è il task a cui fanno riferimento le attività raccolte a partire dalle poche informazioni che il sistema memorizza in relazione ad eventi e attività. Per risolvere questo problema abbiamo deciso di costruire un motore a regole in grado di lavorare con i dati semantici rappresentati dall'ontologia. Il linguaggio utilizzato dal motore a regole è SWRL[11][23] e la piattaforma utilizzata per scrivere le regole è Protégé.

3.1 SWRL ed il Motore a Regole

Il linguaggio SWRL è basato su una combinazione dei sottolinguaggi OWL DL e OWL Lite del OWL Web Ontology Language[9] con i sottolinguaggi Unary/Binary Datalog RuleML del Rule Markup Language.

Le regole scritte con questo linguaggio si configurano come una implicazione tra un antecedente (che viene indicato con *body*) ed un conseguente (indicato invece con *head*). Come se fosse uno statement di tipo if-then, possiamo spiegare quello che accade nella valutazione della regola come segue: ogni qual volta siano verificate le condizioni esplicitate nell'antecedente, allora sono valide anche le condizioni indicate nel conseguente.

$$body \Rightarrow head$$

[11] Semantic Web Rule Language.

Sia l'antecedente che il conseguente sono formati da atomi $a_1 \wedge \ldots \wedge a_n$ che rappresentano termini espressi in OWL . Le variabili vengono indicate convenzionalmente anteponendo al nome della variabile il punto interrogativo (e.g. ?x). Nel caso in cui l'antecedente sia composto da zero atomi (caso antecedente vuoto), dal sistema viene trattato come banalmente vero; viceversa nel caso in cui sia il conseguente ad essere vuoto, viene trattato come banalmente falso. Osserviamo che regole che hanno un antecedente vuoto possono essere utilizzate per dichiarare concetti validi sempre in maniera incondizionata[23].

Nel caso in cui siano presenti più atomi nell'antecedente o nel conseguente (caso atomi multipli), questi vengono elaborati considerandoli legati da una relazione di tipo congiunzione (AND). L'antecedente (o analogamente il conseguente) verrà dichiarato vero o falso in base alla valutazione complessiva di tutti gli atomi che lo compongono. Esempi di atomi possono essere:

- ***C(x)*** - dove *C* rappresenta una classe o più in generale la descrizione di un datarange in OWL. In questo caso l'atomo *C(x)* viene considerato vero se *x* rappresenta un'istanza della classe *C* o se appartiene al range di valori descritti da *C*.
- ***P(x,y)*** - dove *P* rappresenta una proprietà in OWL che lega la variabile *x* alla variabile *y*. In questo caso l'atomo *P(x,y)* viene considerato vero nel caso in cui le variabili *x* e *y* siano correlate attraverso la proprietà *P*.
- ***sameAs(x,y)*** e ***differentFrom(x,y)*** - dove *x* e *y* sono le due variabili che vengono rispettivamente dichiarate come la stessa variabile o come variabili differenti. In questo caso gli atomi vengono ritenuti validi se *x* e *y* sono rispettivamente lo stesso oggetto ontologico o due oggetti ontologici differenti.
- ***builtIn(r,x,...)*** - dove *r* è una relazione conosciuta dal sistema ed *x,y* rappresentano le variabili. In questo caso l'atomo viene ritenuto valido se la relazione *r* è in grado di interpretare ed elaborare le variabili indicate. È possibile notare che sono presenti un insieme di builtIns implementati in ottica modulare da SWRL strutturati per supportare estensioni future. Più precisamente i builtIns proposti rappresentano il riutilizzo di builtIns di XQuery[24] e XPath[25], basati a loro volta su XML Scheme. I builtIn vengono identificati con lo spazio dei nomi http://www.w3.org/2003/11/swrlb. É

importante osservare che i builtIns lavorano soltanto sul risultato della query effettuata dal sistema e non su tutta l'ontologia sottostante, mantenendo coerenza con l'*open world assumption*[12] su cui si basa OWL.

Per meglio comprendere il modo in cui agisce una regola, mi avvalgo di un esempio concreto. Immaginiamo di star realizzando una ontologia che descrive la famiglia e le relazioni di parentela tra gli individui. Il mio obiettivo è costruire una regola che sia in grado di dedurre che la combinazione delle proprietà *hasParent* e *hasBrother* implica la proprietà *hasOncle* (**Figura 3.1**). Possiamo scrivere questa regola con la seguente sintassi:

$$hasParent(?x1, ?x2) \wedge hasBrother(?x2, ?x3) \Rightarrow hasOncle(?x1, ?x3)$$

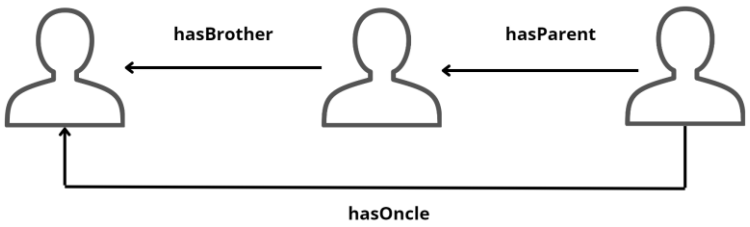


Figura 3.1: Rappresentazione grafica della regola descritta.

É possibile esprimere la stessa relazione nella sintassi concreta XML[13] (**Figura 3.2**):

[12] Questa assunzione afferma che non è possibile dedurre dall'assenza di un enunciato la non veridicità dello stesso[26].
[13] EXtensible Markup Language.

```
<ruleml:imp>
 <ruleml:_rlab ruleml:href="#example1"/>
 <ruleml:_body>
   <swrlx:individualPropertyAtom  swrlx:property="hasParent">
     <ruleml:var>x1</ruleml:var>
     <ruleml:var>x2</ruleml:var>
   </swrlx:individualPropertyAtom>
   <swrlx:individualPropertyAtom  swrlx:property="hasBrother">
     <ruleml:var>x2</ruleml:var>
     <ruleml:var>x3</ruleml:var>
   </swrlx:individualPropertyAtom>
 </ruleml:_body>
 <ruleml:_head>
   <swrlx:individualPropertyAtom  swrlx:property="hasUncle">
     <ruleml:var>x1</ruleml:var>
     <ruleml:var>x3</ruleml:var>
   </swrlx:individualPropertyAtom>
 </ruleml:_head>
</ruleml:imp>
```

Figura 3.2: Rappresentazione della regola precedentemente utilizzata come esempio nella sintassi concreta di XML. Fonte:Semantic Web Rule Language - Wikipedia.

Possiamo adesso cominciare ad intuire la potenza di questo genere di inferenze in sistemi organizzati attraverso architetture ontologiche. Aver memorizzato i dati in maniera semantica non solo aggiunge di per sé valore all'informazione, ma dà anche la possibilità di trarre dinamicamente nuova conoscenza a partire da quella memorizzata in relazione alle esigenze. Il dato può venir dunque elaborato ed interrogato in maniera differente in base alle necessità mantenendo la coerenza con la realtà che descrive. Anche nel settore biomedicale questo tipo di strumento può rivelarsi particolarmente utile per interrogare strutture ontologiche che rappresentano l'ambiente sanitario e tutti i suoi attori. Ad esempio, la visualizzazione dei legami tra gli episodi di malattia pregressi di un paziente ed il suo stato attuale di salute può rivelarsi un utile strumento di supporto alla diagnosi. La semantica ha un ruolo così decisivo nell'elaborazione dell'informazione perché non sono i numeri, ma il contesto ed i legami gli elementi decisivi per rappresentare la realtà.

Lo strumento che utilizziamo per rendere possibile questa catena di inferenze è il motore a regole. Nella creazione dell'applicativo My2Sec è stato realizzato un motore a regole SWRL

utilizzando una implementazione di Drools[14] di un reasoner RL OWL 2 basato sulla SWRLAPI[28].

Prima di descrivere il motore a regole di SWRL diamo alcune definizioni:

- **Human Expert**: con questo termine si intende una persona competente in un determinato dominio di interesse, capace di riorganizzare la conoscenza sotto forma di regole.
- **Expert System**: Nel contesto dell'intelligenza artificiale viene definito expert-system un sistema informatico capace di emulare le capacità decisionali di un *human expert*[29].
- **Inference Engine**: rappresenta il cuore dell'*expert system*. Nel contesto dell'intelligenza artificiale si configura come il componente del sistema che deduce nuove informazioni applicando delle regole alla base di conoscenza rappresentata[30].
- **Rule Engine**: chiamato anche *semantic reasoner* o più semplicemente *reasoner*, rappresenta un componente software in grado di dedurre conseguenze logiche da un insieme di fatti o regole dichiarate nel sistema. La nozione di *rule engine* generalizza il concetto di *inference engine*, fornendo la possibilità di lavorare con un insieme più ricco di strumenti[31]. Il *rule engine* rappresenta dunque una forma di AI semantica dove le regole dichiarano la logica desiderata nella connessione dei dati[32].

L'inferenza può essere dedotta mediante due principali strategie: *Goal-Directed/Backward Reasoning* e *Data-Driven/Forward Reasoning*. Nel primo caso il *reasoner* parte da un insieme di obiettivi (*goal*) e ragiona a ritroso individuando quali sono i sub-*goal* da verificare per raggiungere l'obiettivo desiderato. Il motore a regole dunque, cerca nel conseguente delle regole memorizzate quello che corrisponde all'obiettivo da verificare. Se l'antecedente è noto come vero l'inferenza è conclusa, viceversa l'antecedente viene aggiunto alla lista degli obiettivi (infatti risulta necessario individuare dei dati a conferma dell'antecedente per soddisfare il primo obiettivo)[33]. Nel secondo caso invece il *reasoner* parte dai dati disponibili

[14] É un sistema di gestione delle regole aziendali (BRMS) attraverso un motore a regole basato sulla concatenazione in avanti e all'indietro[27].

e utilizza le regole per estrarre nuova conoscenza. Il motore a regole dunque cerca, tra le regole disponibili, quelle che hanno l'antecedente valutato come vero rispetto ai dati disponibili e inferisce quello dichiarato nel corrispettivo conseguente[34].

I dati su cui verranno eseguite le regole sono contenuti in un database chiamato *working memory*, che rappresenta l'insieme delle informazioni inerenti al problema da risolvere. Con il termine *agenda* intendiamo invece il dispositivo utilizzato per contenere l'insieme delle regole attivate[35] (**Figura 3.3**). Convenzionalmente l'antecedente della regola viene anche chiamato LHS (left-hand side), mentre il conseguente RHS (right-hand side). Una regola si dice "attivata" quando la LHS viene valutata come vera, mentre si dice "Fired" quando vengono eseguite le operazioni indicate nella RHS. Il sistema termina la valutazione delle regole quando all'interno dell'agenda non si hanno più regole attive.

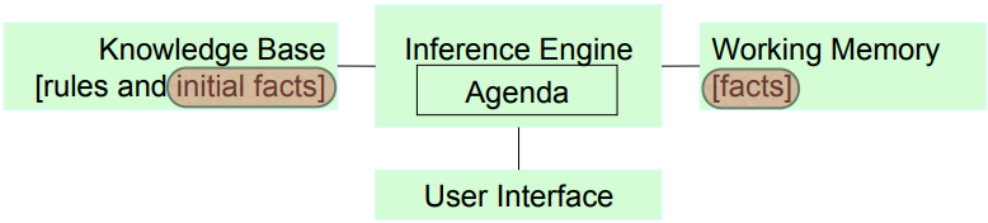


Figura 3.3: Rappresentazione grafica di un motore a regole.
Fonte:http://www.dia.uniroma3.it/~ia/docs/old/KR_Esercitazione.pdf .

Il motore a regole nella configurazione *Data-Driven* esegue le sue valutazioni attraverso le seguenti fasi (**Figura 3.4**):

1. Fase 1: **MATCH**. Il sistema valuta gli LHS delle regole per individuare quali sono soddisfatti dai dati attualmente presenti nella *working memory*. Osserviamo che potrebbero essere soddisfatti uno o più LHS. Questa fase restituisce come output l'insieme delle regole che hanno un antecedente vero (*conflict set*).
2. Fase 2: **CONFLICT RESOLUTION**. In questa fase il sistema ordina le regole nel *conflict set* e seleziona quella da eseguire. Questa fase può procedere attraverso differenti algoritmi. Nel nostro caso specifico, Drools 6 utilizza PHREAK. Questo algoritmo è un'evoluzione di RETEOO utilizzato nelle versioni precedenti[44].

3. Fase 3: **ACT**. Il sistema esegue le azioni indicate nel RHS della regola selezionata.
4. Fase 4: **GOTO 1**[36].

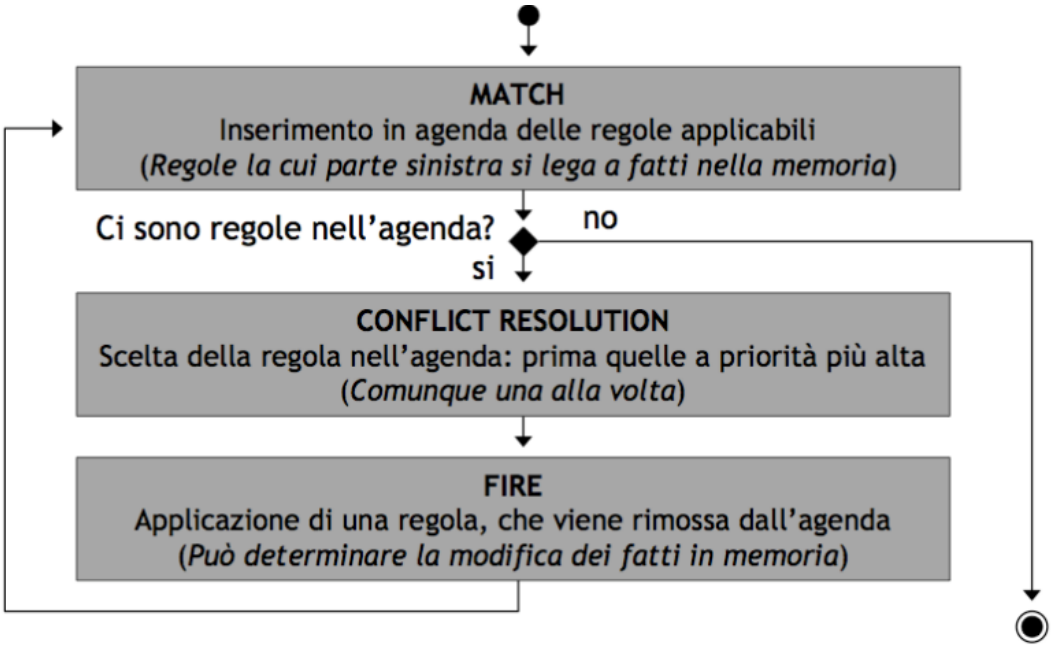


Figura 3.4: Flow-chart dell'esecuzione di una regola. Fonte:http://www.dia.uniroma3.it/~ia/docs/old/KR_Esercitazione.pdf .

3.2 Activity to Task in My2Sec

Durante la realizzazione della regola in grado di associare ad ogni attività il corrispettivo task sono state riscontrate numerose complessità relative all'associazione delle attività quando al lavoratore erano assegnati più task contemporaneamente. Per superare questa difficoltà si è scelto in prima istanza di semplificare il problema introducendo un'ipotesi di lavoro: ad ogni *Member* è assegnato un singolo *Task* ed ogni *Task* è composto da *SubTask* caratterizzati da differenti *ActivityType*.

Per strutturare la regola sono state sfruttate molte delle *object property* presenti nell'ontologia (**Figura 3.5**).

Name

Activity to Task

Comment

This rule associates each activity with its task.

Status

Ok

```
my2sec:attachedTo(?T, ?M) ^ my2sec:Member(?M) ^
my2sec:partOf(?S, ?T) ^ my2sec:hasActivityType(?AV, ?AT) ^
my2sec:hasActivityType(?S, ?AT) ^ my2sec:hasMember(?AV, ?M)
-> sqwrl:select(?M, ?AV, ?T) ^ my2sec:hasTask(?AV, ?T) ^
sqwrl:columnNames("Member", "Activity", "Task")
```

Figura 3.5: Regola scritta in SWRL per associare ad ogni Activity il suo Task.

Come è possibile osservare dall'immagine sottostante (**Figura 3.6)** la regola seleziona tutte le *Activity* associate ad un *Member* ed il *Task*. Successivamente vengono selezionati tutti i *SubTask* di cui è composto il *Task*. Infine vengono confrontati gli *ActivityType* associati al *SubTask* e gli *ActivityType* associati alle *Activity*, se sono uguali allora l'*Activity* appartiene a quello specifico *SubTask*, e di conseguenza al *Task*.

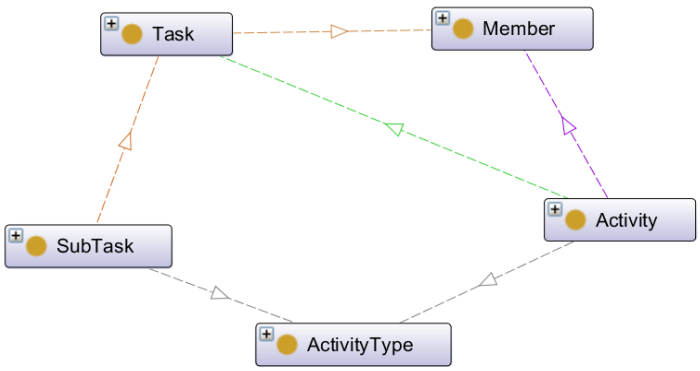


Figura 3.6: Rappresentazione grafica della regola scritta in SWRL per associare ad ogni Activity il suo Task.

CAPITOLO 4 - **Applicazione di My2Sec al contesto Biomedicale**

Come anticipato nell'Introduzione, una volta ultimata la costruzione dell'ontologia e completato lo sviluppo dell'applicazione, è stato naturale chiedersi quali sarebbero potuti essere i contesti applicativi in cui My2Sec avrebbe aggiunto valore. L'esplorazione delle potenzialità dell'applicativo è iniziata dal settore biomedicale industriale insieme all'azienda SIMAD[37]. Questa realtà si occupa dal 1989 della progettazione e della costruzione di strumentazione che facilità l'utilizzo e l'applicazione della visione RX in campo chirurgico. Il flusso di lavoro caratteristico dell'azienda segue l'apparecchiatura dall'ideazione fino alla produzione, distribuendo le responsabilità ed i ruoli su un sistema di lavoratori eterogeneo.

Avere l'opportunità di conoscere il processo di realizzazione dell'apparecchiatura biomedicale è stato non solo importante per settorializzare My2Sec fornendogli un caso studio concreto, ma anche l'occasione per mettere in discussione la solidità e la funzionalità della ontologia prodotta. Infatti la molteplicità delle figure professionali coinvolte, la divisione dei compiti tra i lavoratori e la varietà delle attività svolte ha richiesto il rimaneggiamento dell'ontologia (Capitolo 4.2).

4.1 Metodologia di analisi del flusso di lavoro

Il primo passo per comprendere come migliorare l'ontologia di My2Sec è stato individuare le specifiche esigenze descrittive dell'azienda e più in generale del settore. Per fare questo è stato necessario svolgere un'indagine *in loco* dove ho raccolto informazioni riguardo il flusso di lavoro caratteristico e le figure professionali coinvolte. Insieme all'azienda abbiamo convenuto

su quali fossero i tipi di lavoratori più interessanti da profilare. Questa scelta è strettamente legata ai limiti e alle potenzialità di My2Sec. L'applicativo infatti riesce a tener traccia di tutte e sole le attività svolte sul supporto informatico, ma non in tutti i lavori questo genere di attività è cruciale o semanticamente importante per raggiungere un vantaggio competitivo. La scelta è ricaduta su due figure professionali dell'area di ricerca e sviluppo: lo sviluppatore informatico ed il disegnatore meccanico. Entrambe queste figure infatti svolgono la maggior parte del lavoro su un supporto informatico e le informazioni prodotte hanno valore in relazione alla gestione del flusso operativo.

Selezionate le figure professionali è iniziata la fase di profilazione vera e propria. Ho avuto l'opportunità di confrontarmi singolarmente con i lavoratori per individuare:

- Il loro ruolo nel flusso operativo;
- L'insieme di figure o processi con cui scambiano informazione;
- Le modalità con cui l'informazione viene scambiata ed il valore che assume nel suo contesto;
- Gli strumenti software e hardware utilizzati per svolgere la normale attività lavorativa;
- Le attività caratteristiche, distinguendole da quelle di supporto;
- Il tempo mediamente dedicato alle singole attività.

Nello specifico per quanto riguarda i software utilizzati, ho chiesto ai lavoratori di compilare un report dove indicavano per ogni applicativo l'insieme delle attività per le quali viene utilizzato (**Allegato IV**). Questo passaggio è stato per me importante al fine di valutare la possibilità di realizzare un secondo motore a regole capace di associare univocamente le attività generate da un'applicazione ad uno specifico tipo di attività (*ActivityType*), e di conseguenza ad un *Task*. Dall'analisi dei dati è emerso che esistono attività concettualmente e funzionalmente differenti che però sono attualmente indistinguibili dalla macchina. Ad esempio, nel caso dello sviluppatore informatico, le attività di Testing, Developing e Debugging rappresentano tre tipologie di attività differenti con tre differenti finalità specifiche. Utilizzando però spesso le stesse applicazioni con le stesse modalità ed essendo attività che vengono eseguite in parallelo, non è sempre possibile riconoscerle singolarmente. Abbiamo dunque concluso che utilizzare

un motore a regole per questo genere di classificazione rischia di far diventare il sistema estremamente rigido e poco scalabile (nonché difficile da gestire per quanto riguarda la privacy ed il rispetto delle normative GDPR). Al contrario l'utilizzo di una AI adattiva per effettuare la classificazione permette, con l'aiuto dell'utente, di individuare e classificare in maniera sufficientemente efficace le attività.

Dai dati raccolti ho costruito un cluster per ogni professione (**Allegato II**) che descrive i tipi di attività caratteristici del lavoratore tenendo traccia sia delle attività utilizzabili che di quelle trascurate dal sistema (ovvero quelle non svolte sul supporto informatico). La scelta di inventariare tutte le attività del lavoratore a prescindere dalla possibilità di utilizzarle è legata alla volontà, ancora una volta, di individuare i limiti e le potenzialità dell'applicativo. Soltanto in questa maniera è infatti possibile identificare le prospettive di crescita. Ultimato il processo di profilazione abbiamo valutato insieme ai lavoratori se il livello di sintesi scelto nella costruzione del cluster fosse sufficientemente descrittivo della loro professione.

4.2 Progettazione e costruzione dell'ontologia 3.0

Dall'analisi svolta risulta evidente la necessità di associare ad ogni *Member* l'insieme di *ActivityType* che caratterizzano la professione in funzione del ruolo lavorativo che ricopre all'interno dell'azienda. Per questo motivo, nella progettazione dell'ontologia 3.0, ho ritenuto opportuno inserire una nuova classe che rappresentasse l'insieme delle professioni (*Profession*) descritte. Ogni *Profession* è associata a degli *ActivityType* caratteristici attraverso la proprietà *hasActivityType*. In questo modo, una volta che il sistema associa al *Member* la sua *Profession*, l'associazione tra il *Member* ed i tipi di attività svolti è implicita (**Figura 4.1**).

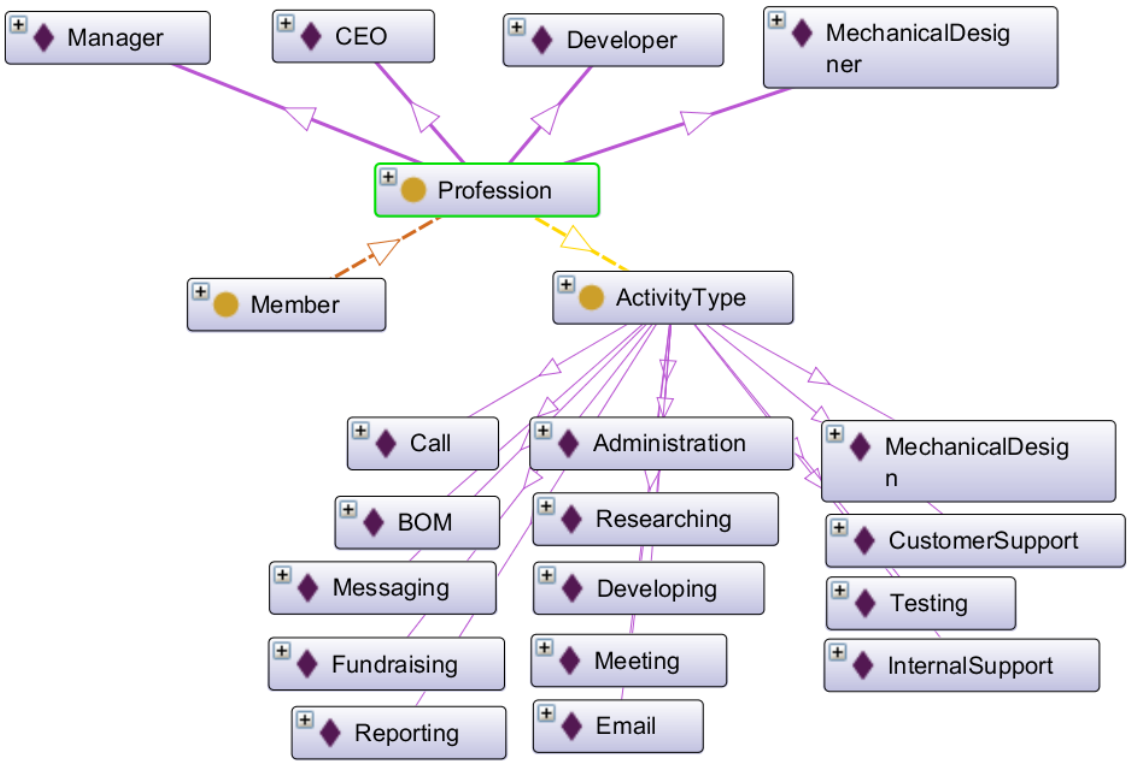


Figura 4.1: Rappresentazione grafica del collegamento tra Member-Profession-ActivityType.

Tra le alternative di organizzazione semantica possibili, l'altra ipotesi valutata prevedeva di creare una classe per ogni tipo di lavoro e di racchiudere tutte queste professioni nella superclasse *Profession*. Successivamente avrei indicato i tipi di attività svolti dalle varie professioni come istanze della sottoclasse specifica. Ho optato per la prima alternativa presentata per due principali motivazioni:

1. Rappresentare le professioni come istanze della classe *Profession* rende il sistema modulare e molto semplice da aggiornare. Per aggiungere una nuova professione è infatti sufficiente aggiungere una nuova istanza alla classe e collegarla agli *ActivityType* caratteristici.
2. Poiché più professioni condividono gli stessi tipi di attività, avere una classe di *ActivityType* parallela a quella delle *Profession* mi permette di poter associare lo stesso *ActivityType* a più professioni senza la necessità di creare duplicati semantici.

La restante parte della struttura dell'ontologia è rimasta sostanzialmente invariata, per un maggiore dettaglio si invita la visualizzazione dell'**Allegato III**.

Per aumentare il potere descrittivo abbiamo scelto di utilizzare l'ontologia ESCO[15] per definire le figure professionali. Questa ontologia classifica le abilità, le competenze e le occupazioni, assumendo il ruolo di un dizionario rispetto alle caratteristiche specifiche del mercato del lavoro. Tutti i concetti rappresentati sono *machine understandable* e possono essere utilizzati per facilitare il processo di ricerca del lavoro sulla base delle competenze. L'obiettivo di questa ontologia è infatti costruire un mercato del lavoro europeo integrato ed efficiente[38].

[15] European Skills, Competences, Qualifications and Occupations.

CAPITOLO 5 - **Realizzazione software del TaskAI**

Per la realizzazione del *TaskAI aggregator* il linguaggio di programmazione software scelto è Java. Il paradigma di programmazione proposto dal linguaggio è di tipo OOP[16] ed offre notevoli vantaggi rispetto alla tradizionale programmazione procedurale. Permette infatti la realizzazione di una struttura chiara dei programmi, facile da manutenere ed aggiornare, che rende il codice Java DRY[17]. Questo principio sottolinea la possibilità di evitare la ripetizione del codice, permettendo la riduzione sia del tempo di sviluppo che del codice stesso.

Le classi e gli oggetti sono gli elementi fondamentali di questo genere di programmazione. In perfetta analogia con quanto visto per le ontologie nel Capitolo 1.2, le classi rappresentano un insieme di enti con caratteristiche comuni, mentre gli oggetti rappresentano le istanze della classe. Lo strumento utilizzato per inizializzare gli oggetti è il costruttore. Possono inoltre venir definiti degli attributi per le classi, che rappresentano le variabili della classe, e dei metodi che vengono principalmente utilizzati per eseguire procedure. Java mette inoltre a disposizione degli sviluppatori un insieme di modificatori di accesso che regolano la possibilità di accedere ad una classe, ad un metodo, ad un attributo o ad un costruttore. Nello specifico i modificatori di accesso si dividono in due famiglie: Access Modifier e Non-Acces Modifier. I primi, al contrario dei secondi, controllano e regolano il livello di accesso. In compenso i secondi consentono altro genere di funzionalità. Tra gli Access Modifier utilizzati nel codice troviamo: *public* (che rende il codice accessibile da altre classi) e *private* (che rende il codice accessibile solo all'interno della classe in cui è stato dichiarato)[39].

[16] Object-Oriented Programming.
[17] Don't Repeat Yourself.

5.1 Metodica e fasi di sviluppo del TaskAI aggregator

L'obiettivo del *TaskAI aggregator* consiste nell'elaborazione, mediante la regola sviluppata, dei dati raccolti da *ActivityWatch* volta ad associare ad ogni *Activity* il corrispettivo *Task* e nella produzione del *LogTime* complessivo associato ad ogni *Task*. Per raggiungere questo obiettivo è stato necessario creare un nuovo aggregatore, il RuleManager, il cui ruolo è quello di implementare il motore a regole utilizzando la SWRLAPI. Parallelamente allo sviluppo del RuleManager è stato creato un ambiente di testing, TaskAITester, che ha reso possibile la validazione del codice prodotto. Avere un ambiente di testing controllato e sviluppato *ad hoc* per l'applicativo ha reso molto più semplice l'individuazione di bug nel codice ed il fixing degli stessi. In seguito abbiamo implementato il metodo *aggregate* nel TaskAI che, servendosi del RuleManager, elabora i dati prelevati dal SEPA e fornisce l'output richiesto.

La regola presentata nel Capitolo 3.3 è stata ulteriormente modificata per permettere di visualizzare il *Task*, il tempo speso da ciascuna *Activity* sul *Task*, ed il *Member* relativo (**Figura 5.1**). Come è possibile osservare è stata utilizzata la *datatype property numericDuration* (timeOntolology) per collegare ogni attività alla sua durata. Sottolineo inoltre che è stata introdotta una ulteriore modifica che permette alla regola di funzionare correttamente anche in assenza di *SubTask*. Non erano infatti presenti *SubTask* associati ai *Task* dei lavoratori che in fase di produzione stavano utilizzando l'applicativo.

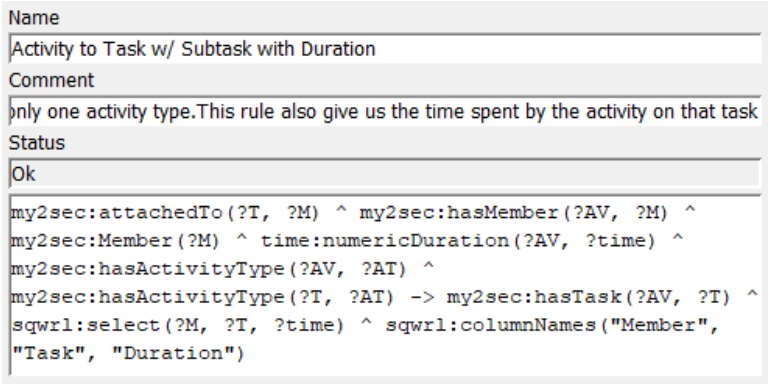


Figura 5.1: Regola scritta in SWRL per associare ad ogni Activity il suo Task (in assenza di SubTask) e per visualizzare la quantità di tempo speso da ciascuna Activity sul Task.

5.2 Analisi dei moduli software prodotti

Per quanto riguarda la realizzazione del **RuleManager** è stato necessario importare numerose classi della OWLAPI e della SWRLAPI per permettere l'utilizzo agevole degli oggetti ontologici. Successivamente sono state dichiarate ed inizializzate le istanze delle classi e delle proprietà che verranno utilizzate. Il modificatore di accesso scelto per questi oggetti è *private*. La scelta fa riferimento alla volontà di lasciare esclusivamente al RuleManager la possibilità di modificare tali istanze. All'interno del costruttore viene inizializzata l'ontologia che è fornita al sistema sotto forma di file OWL Functional Syntax. Questa è solo una delle possibili sintassi disponibili per OWL, ma si configura come quella più adatta per l'implementazione di strumenti come API e *reasoner*[40, p. 2]. Sono stati in conclusione definiti un insieme di metodi utili all'inserimento e alla rimozione dell'informazione dall'ontologia. É infatti necessario sia fornire i dati al *ruleEngine*, in modo tale che possa valutarli ed elaborare il risultato richiesto, ma anche eliminarli, in modo tale da procedere con le successive elaborazioni. Sarà proprio all'interno del TaskAI che questi metodi verranno invocati, fornendo in ingresso i parametri richiesti. Vi è inoltre un ultimo metodo che permette al *queryEngine* di iniziare a fare inferenza. Di seguito sono riportati i metodi definiti nel RuleManager:

- public void **add_member**(String usergraph);
- public void **remove_member**(String usergraph);
- public void **add_task**(String usergraph,String taskUri, String activity_type);
- public void **remove_task**(String usergraph,String taskUri, String activity_type);
- public void **add_activity**(String usergraph, String activityUri, String activity_type, String duration);
- public void **remove_activity**(String usergraph, String activityUri, String activity_type, String duration);
- public SQWRLResult **parseRule**().

Per quanto invece riguarda il **TaskAI**, questo estende la classe ITool. Questo genere di operazione permette numerosi vantaggi dal punto di vista applicativo semplificando il riuso del codice attraverso la creazione di relazioni gerarchiche. Consente infatti alla classe TaskAI

(sottoclasse) di ereditare campi e metodi presenti in ITool (superclasse) permettendo inoltre l'Override dei metodi.

Il cuore del TaskAI è il metodo *aggregate*. Osserviamo adesso i parametri in ingresso alla funzione:

- String userUri;
- HashMap<RDFTermURI,HashMap<RDFTermURI,Float>> activities;
- HashMap<RDFTermURI,HashSet<RDFTermURI>> tasks.

Sia le activities che i tasks sono rappresentati da strutture a grafo ottenute attraverso l'utilizzo di HashMap annidati. L'HashMap è una classe che implementa l'interfaccia Map e ci offre la possibilità di salvare informazioni in coppie chiave-valore. Risulta dunque possibile accedere al valore memorizzato nell'HashMap facendo riferimento alla sua chiave[41]. L'HashSet è invece una classe che implementa l'interfaccia Set e consente di creare una collezione di elementi unici[42]. Nello specifico nell'HashMap delle activities sono memorizzati: gli ActivityType, gli URI delle attività e le durate delle attività di un unico utente (**Figura 5.2, a**). Mentre nell'HashMap dei tasks sono memorizzati: gli *ActivityType* dei *Task* e l'URI del *Task* di un unico utente (**Figura 5.2, b**).

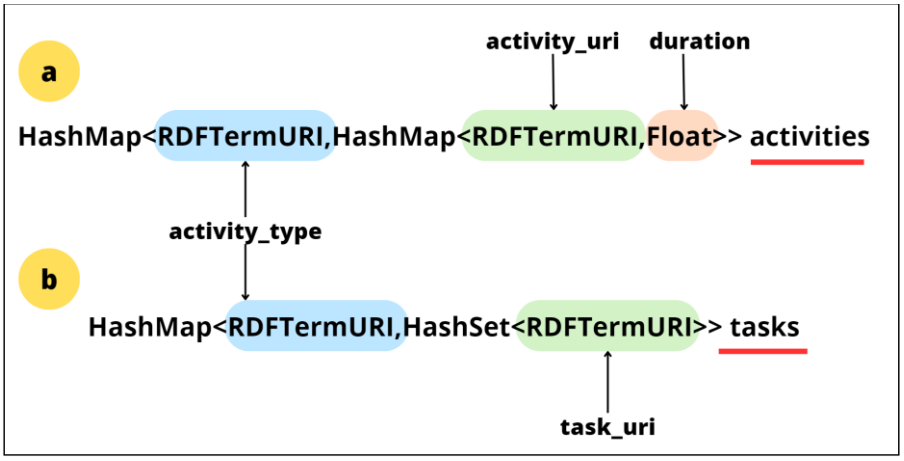


Figura 5.2: HashMap di activities e tasks.

Risulta dunque evidente la necessità di effettuare un pre-processing dei dati per estrapolare da queste strutture l'informazione necessaria da passare ai metodi della classe RuleManager. Una volta aggiunti all'ontologia il *Member*, le *Activity* ed il *Task*, viene invocato il metodo *parseRule* che mette in *result* il risultato del *queryEngine*. Attraverso un ciclo while vengono analizzati tutti i risultati della query. Poiché è necessario salvare i risultati dell'elaborazione della regola, abbiamo scelto di effettuare una deep-copy dinamica all'interno del ciclo. Il suo scopo è salvare in un HashMap tutti i risultati generati. In questo HashMap la struttura chiave-valore vede come chiave un RDFTermURI che rappresenta l'URI della *Task*, mentre come valore un Float che rappresenta il *LogTime*. Dopo che le *Activity* sono state associate ai *Task* è infatti necessario sommare le durate delle singole attività associate ad un *Task* per ottenere il tempo complessivamente speso sul *Task* stesso. Questo è reso possibile attraverso l'utilizzo di uno statement di tipo if-else che prende l'URI della *Task* e valuta se è già presente un elemento dell'HashMap che ha questo URI come chiave. Se non esiste allora viene aggiunto e viene inserita la durata associata a quel *Task* come valore. Se è già presente allora viene sommato al valore memorizzato nell'HashMap il valore della durata della nuova attività che è stata associata a quel *Task*. Così facendo si costruisce in maniera incrementale il tempo complessivo.

A conclusione della presentazione del codice è importante sottolineare un aspetto particolare del funzionamento dell'applicativo, strettamente legato alla sua natura data-driven. Poiché i dati vengono continuamente memorizzati in locale sul dispositivo del lavoratore, è importante segnalare al sistema quando è il momento di cominciare l'aggregazione degli stessi. Per questo motivo è stato ideato un sistema di "*download semantico*" che, attraverso l'utilizzo di un *flag* di sincronizzazione, segnala al sistema l'inizio della fase di aggregazione.

È possibile trovare l'intero codice sorgente su GitHub al seguente link: https://github.com/chiara-del/VaimeeToolsTaskAI.

Per quanto riguarda l'interfaccia utente, questa è interamente personalizzabile in base alle esigenze e permette di visualizzare tutte le informazioni utili al monitoraggio del progetto generate dall'utente (**Figura 5.3**). Utilizzando la *observability platform* Grafana[43] è stata inoltre realizzata una dashboard che rende possibile la visualizzazione complessiva dei dati al

Project Manager, con l'obiettivo di permettere la comprensione del flusso del lavoro aziendale (**Figura 5.4**).

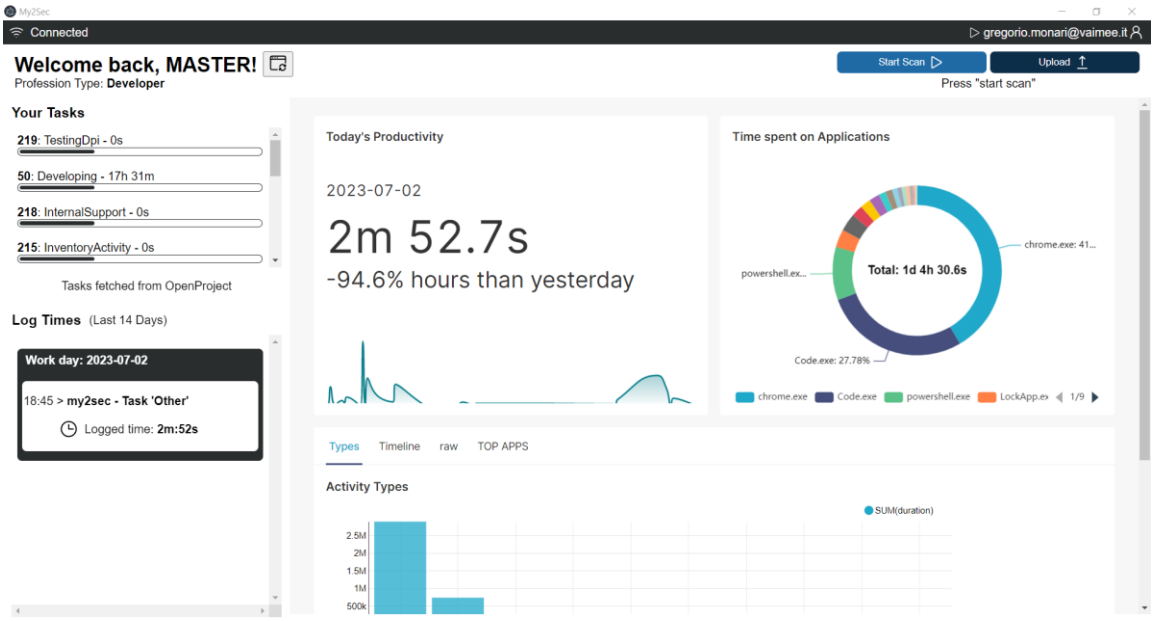


Figura 5.3: Interfaccia utente.

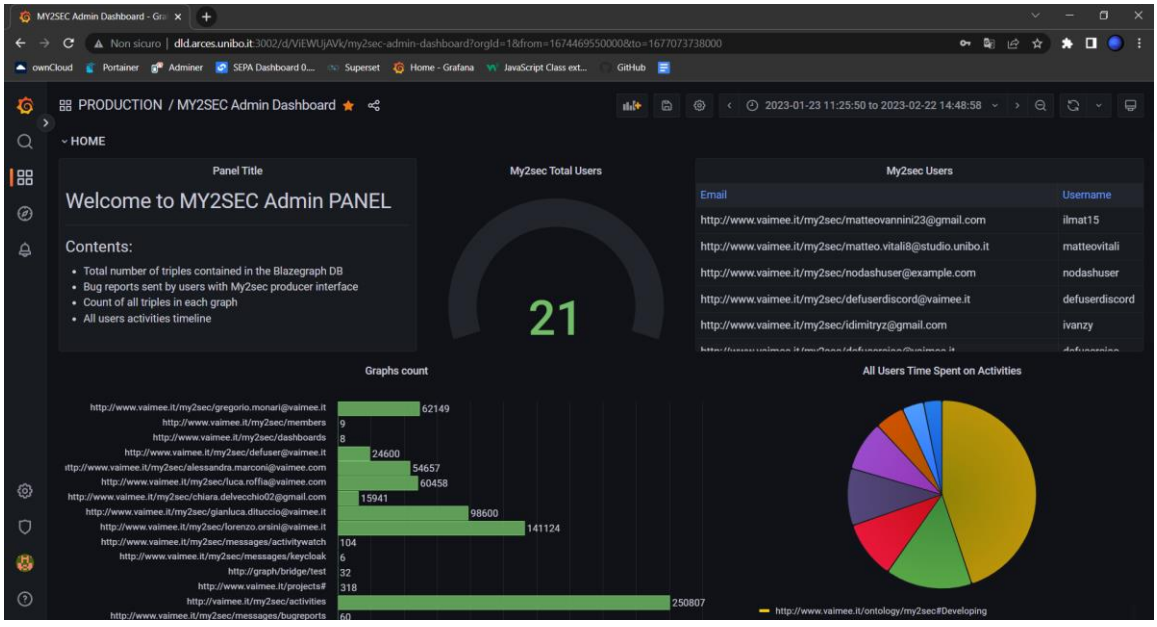


Figura 5.4: Dashboard di visualizzazione complessiva del progetto.

CAPITOLO 6 - **Conclusioni e Prospettive future**

A conclusione dell'elaborato di tesi possiamo osservare che avere introdotto una rappresentazione semantica dell'informazione e del contesto applicativo ha consentito lo sviluppo di una applicazione data-driven capace di fare inferenza in modo efficace ed efficiente. L'elasticità della rappresentazione apre inoltre la strada alla crescita dell'applicativo e all'espansione a quanti più domini applicativi possibili. La conoscenza del flusso di lavoro offerta da My2Sec permette di fare analisi complesse sulla produttività e sulla qualità del lavoro prodotto non in base a indicatori qualitativi e aspecifici, ma in maniera strutturata e accurata. L'apporto della semantica permette infatti una clusterizzazione puntuale dell'applicativo realizzata in relazione alle specifiche esigenze dell'utente.

Tra i limiti descrittivi e rappresentativi dell'applicativo riscontriamo sicuramente l'incapacità di tener traccia di quelle attività manuali o intellettuali che vengono svolte non attraverso il supporto informatico. Questo genere di attività è infatti spesso determinante per la buona riuscita del flusso operativo ma difficile da rilevare e analizzare in termini quantitativi. L'obiettivo per l'immediato futuro prevede di riuscire a integrare l'applicativo con un sistema più ampio di sensoristica fatto di reti IoT[18], che sicuramente amplieranno l'insieme dei dati a disposizione e permetteranno un'indagine ancora più scrupolosa e accurata.

Per la sua natura l'applicativo si presta alla descrizione e allo studio di quei flussi operativi aziendali dove l'utilizzo del calcolatore è imprescindibile e determinante. In questi contesti infatti l'insieme delle informazioni acquisite rappresenta inequivocabilmente la qualità e la

[18] Internet of Things.

produttività del lavoratore interessato. Osserviamo inoltre che My2Sec si propone come uno strumento utile e funzionale allo sviluppo di aziende diffuse. La possibilità di tracciare l'operato del lavoratore è infatti sia uno strumento di garanzia per il datore di lavoro, ma allo stesso tempo di tutela per il lavoratore stesso. In conclusione possiamo aggiungere che la condivisione, la trasparenza e l'interoperabilità dei sistemi riescono ad abbattere le barriere geografiche permettendo l'ampliamento delle prospettive aziendali e intellettuali.

RINGRAZIAMENTI

Quello della tesi è stato un percorso più che una conclusione. È stata una occasione per crescere ed ho avuto l'opportunità di apprendere molto rispetto al mondo del Web Semantico e alla vita lavorativa nel contesto aziendale. Colgo l'occasione per ringraziare di cuore il mio relatore Luca Roffia ed i miei correlatori Alessandra Marconi e Gregorio Monari. Vi ringrazio per la pazienza, per l'ascolto, per la cura e per l'entusiasmo che mi avete trasmesso in tutte le fasi del progetto. Vi ringrazio per la stima e per il confronto sempre onesto rispetto all'operato, con l'obiettivo comune di crescere e migliorare. Ringrazio il professor Giampiero Pirini per la gentilezza e la disponibilità. Ringrazio inoltre sentitamente Franco Fallavena, Andrea Bovina, Augusto Atti e tutta l'azienda SIMAD per lo splendido percorso fatto insieme. Vi ringrazio per l'immediata fiducia, per l'interesse sentito verso il progetto e per tutto il tempo che mi avete dedicato. Entrare in punta di piedi a far parte della vostra realtà è stata un'esperienza unica ed edificante. Condividere con voi idee e opinioni mi ha consentito di maturare da un punto di vista sia professionale che personale, permettendomi di concludere questo percorso con maggiore consapevolezza.

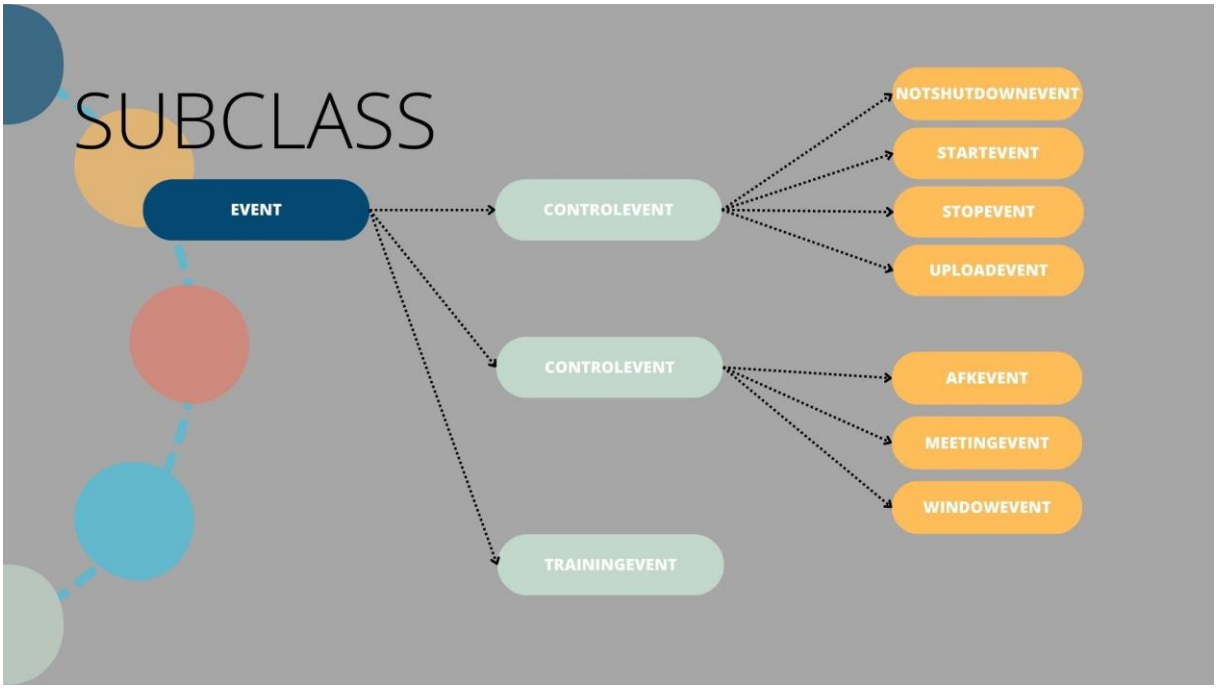
In conclusione vorrei ringraziare nuovamente tutti per aver creduto in me e per aver avuto fiducia nel mio operato. Non è assolutamente banale e scontato, ma per me è fonte sincera di soddisfazione. Grazie.

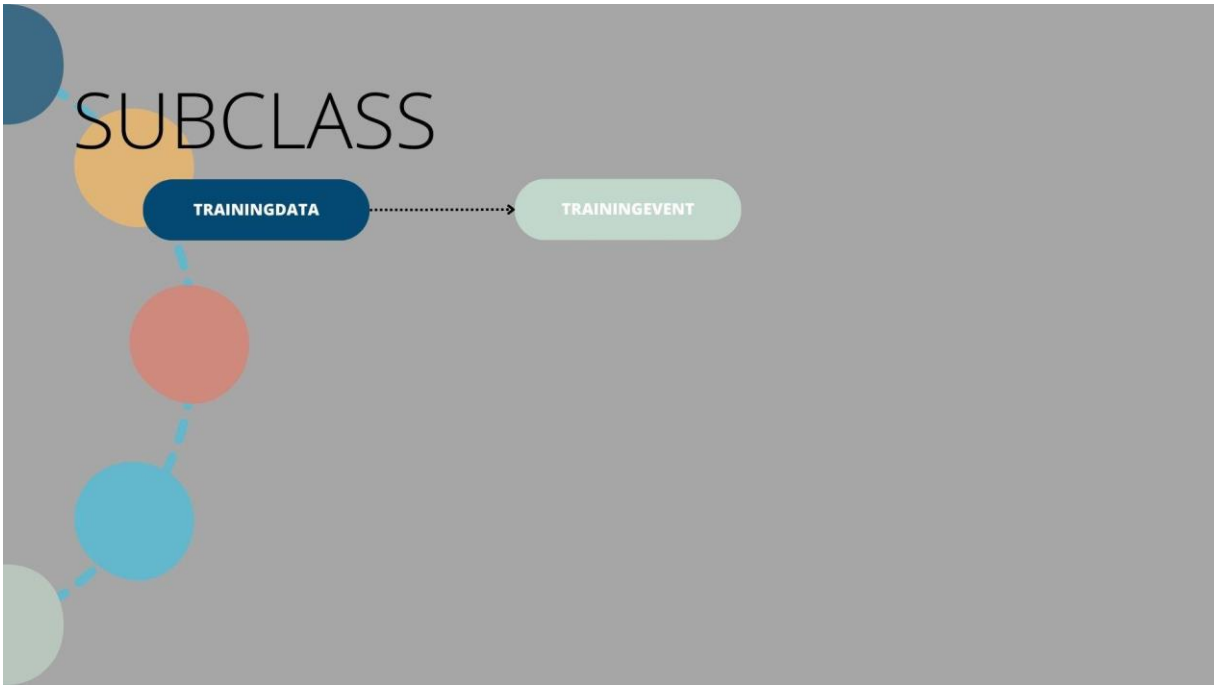
ALLEGATI

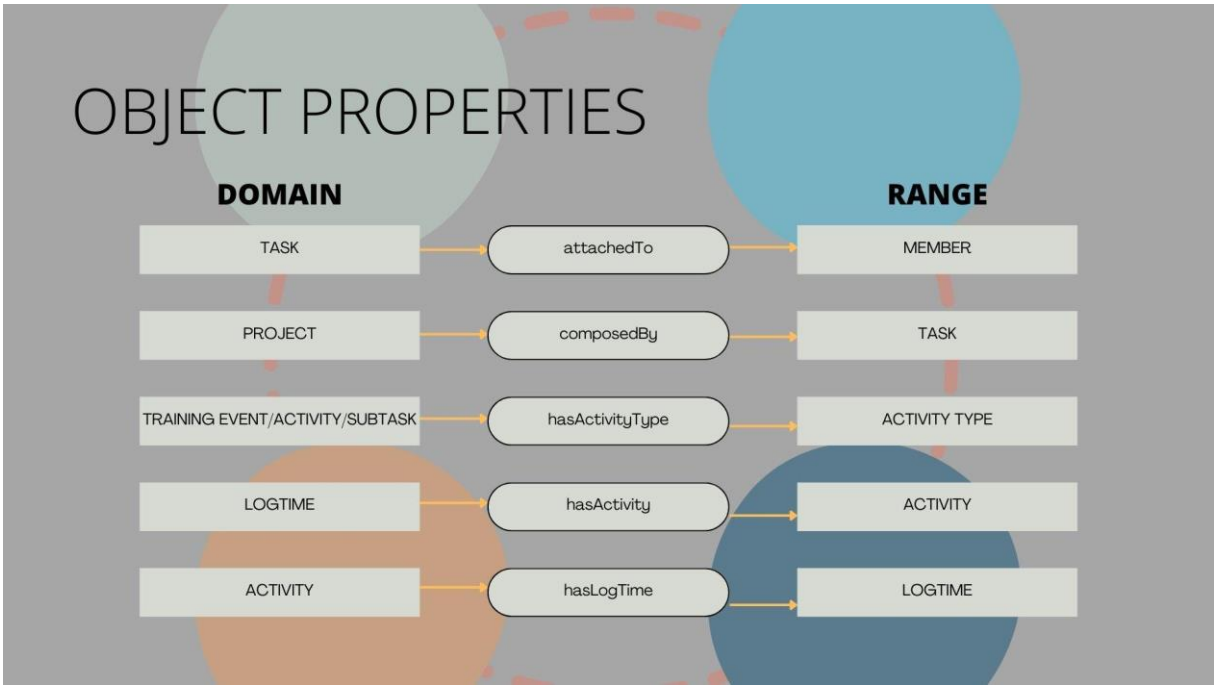
ALLEGATO I

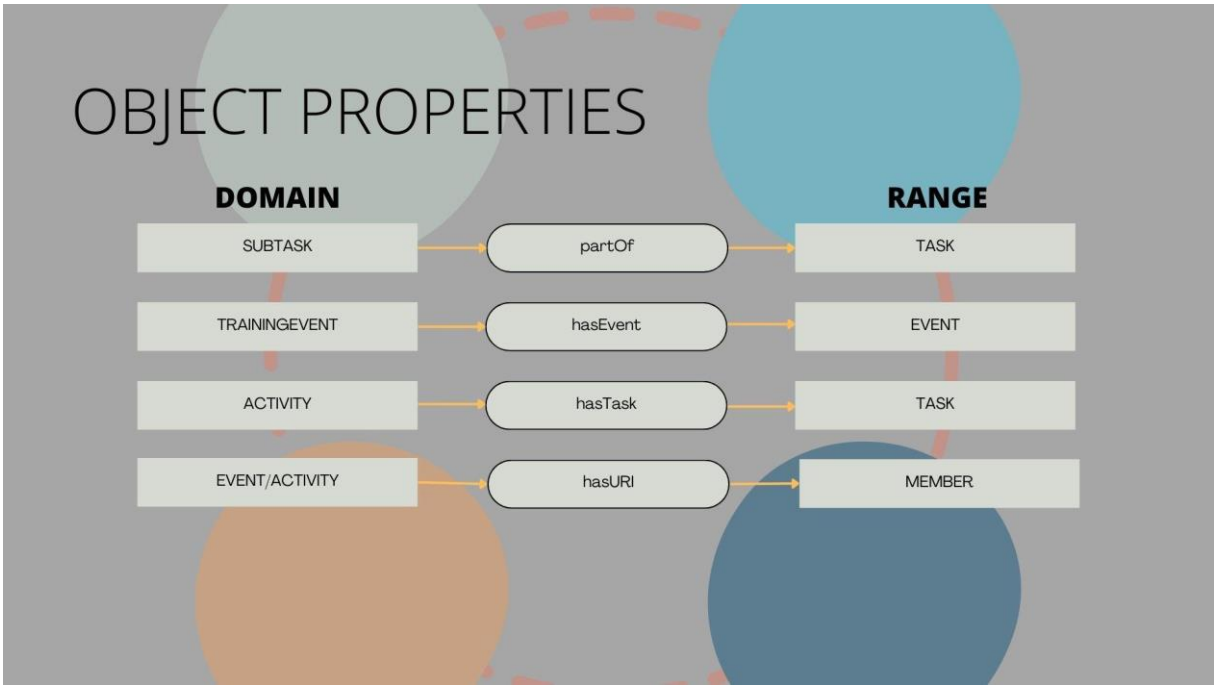
In questo allegato riporto lo schema descrittivo dell'ontologia di My2Sec prima dell'adattamento per SIMAD. Nello specifico sono riportate tutte le classi e tutte le proprietà descritte con le relative gerarchie.

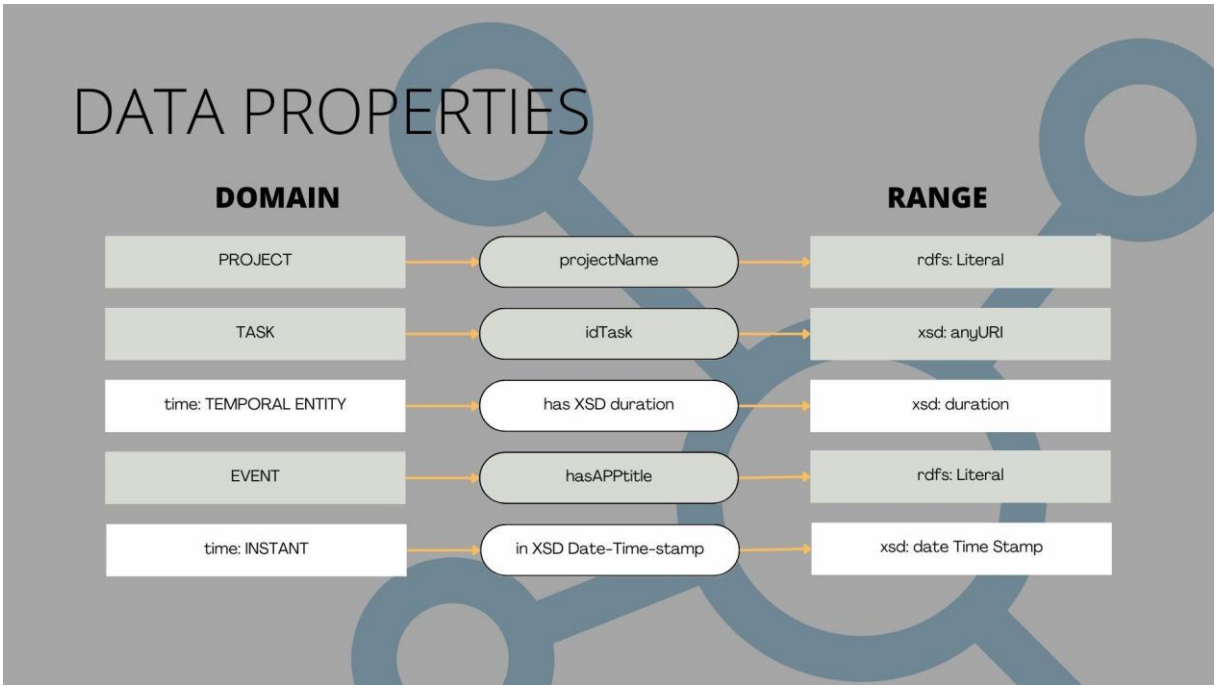

CLASS
ACTIVITY
Duration of time meanwile members doing one specific activity.
LOGTIME
Duration of time spent on a specific activity by one member. It is characterized by date, hours ad an activity.
TRAININGDATA
This class contain dataset used for training AI or something like that.
EVENT
An event is something that happen in one moment and has no temporal duration. It is characterized by timestamp, app title and activity type.

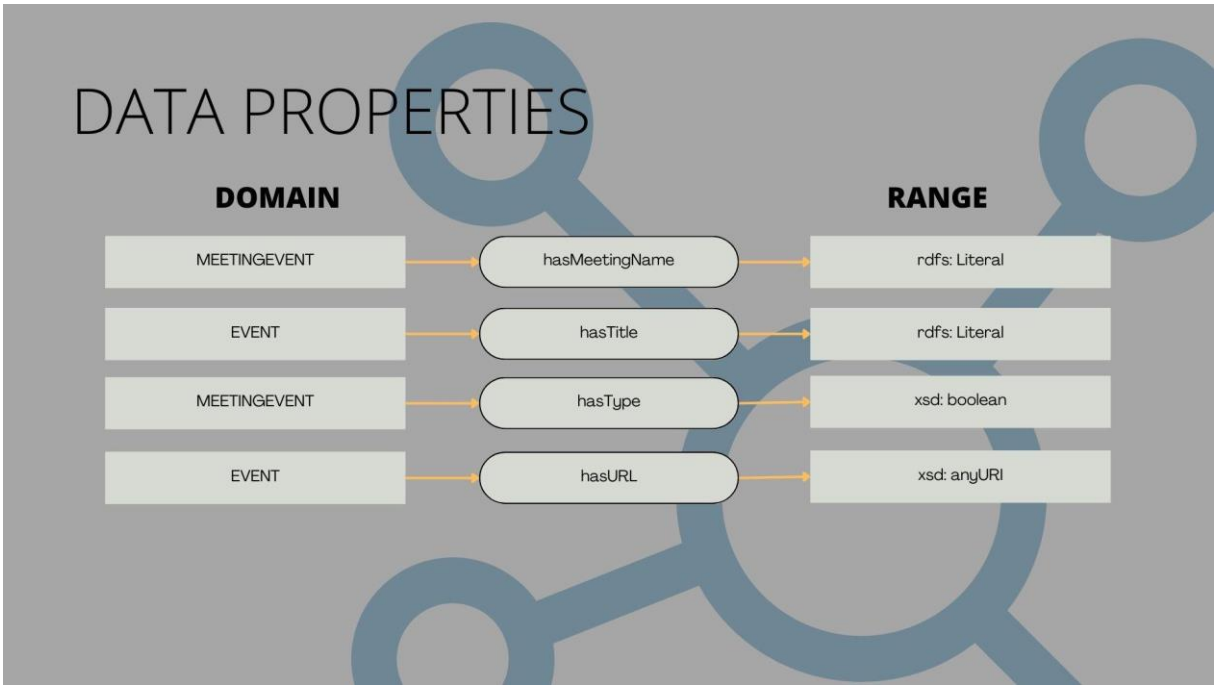

SUBCLASS
EVENT
CONTROLEVENT
CONTROLEVENT
TRAININGEVENT
NOTSHUTDOWNEVENT
STARTEVENT
STOPEVENT
UPLOADEVENT
AFKEVENT
MEETINGEVENT
WINDOWEVENT

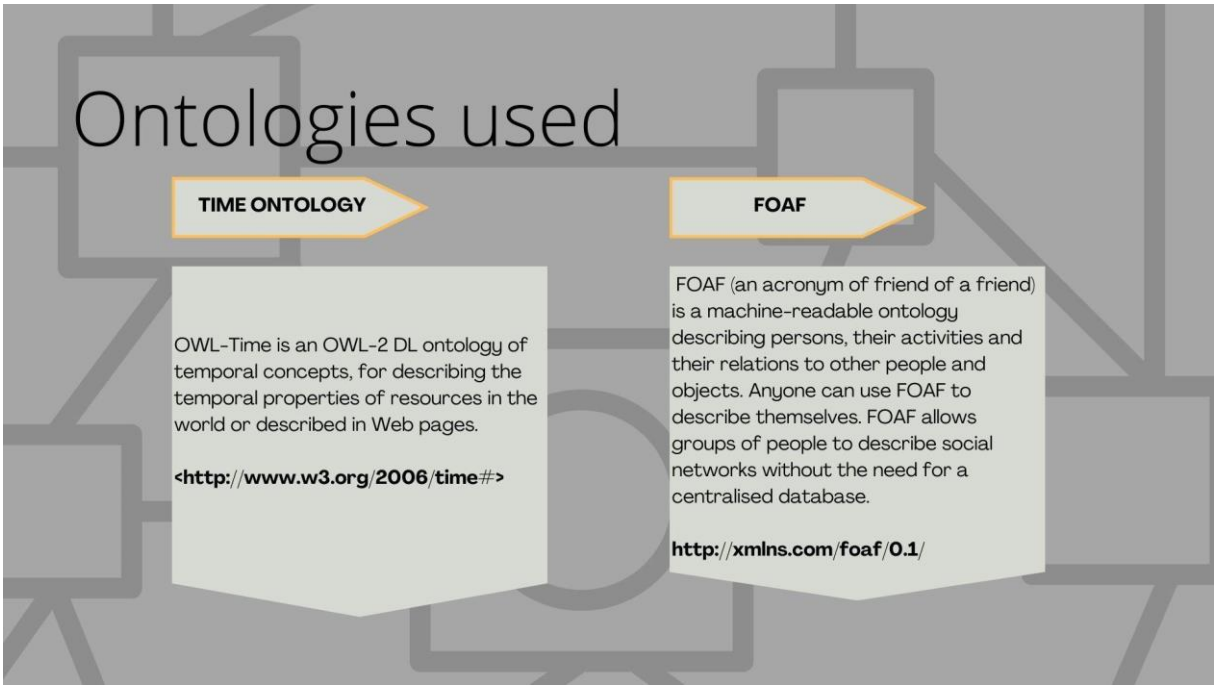

SUBCLASS
TRAININGDATA
TRAININGEVENT


OBJECT PROPERTIES
DOMAIN
RANGE
TASK
attachedTo
MEMBER
PROJECT
composedBy
TASK
TRAINING EVENT/ACTIVITY/SUBTASK
hasActivityType
ACTIVITY TYPE
LOGTIME
hasActivity
ACTIVITY
ACTIVITY
hasLogTime
LOGTIME


OBJECT PROPERTIES
DOMAIN
RANGE
SUBTASK
partOf
TASK
TRAININGEVENT
hasEvent
EVENT
ACTIVITY
hasTask
TASK
EVENT/ACTIVITY
hasURI
MEMBER


DATA PROPERTIES
DOMAIN
RANGE
PROJECT
projectName
rdfs: Literal
TASK
idTask
xsd: anyURI
time: TEMPORAL ENTITY
has XSD duration
xsd: duration
EVENT
hasAPPtitle
rdfs: Literal
time: INSTANT
in XSD Date-Time-stamp
xsd: date Time Stamp

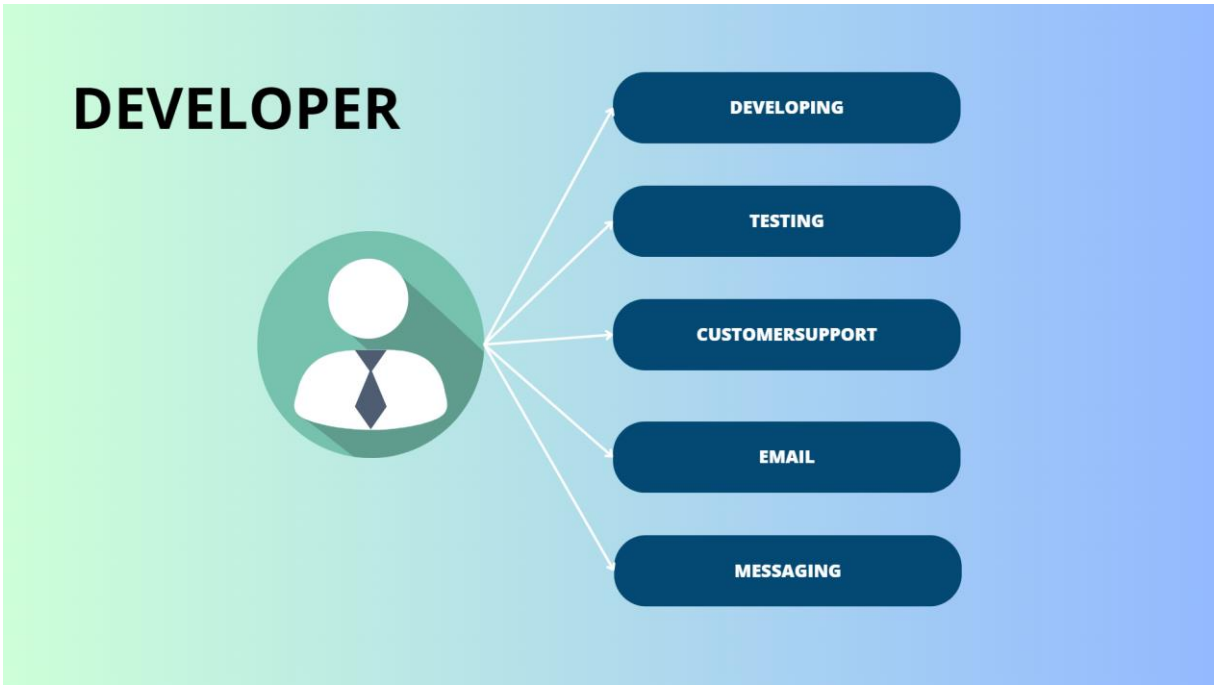

DATA PROPERTIES
DOMAIN
RANGE
MEETINGEVENT
hasMeetingName
rdfs: Literal
EVENT
hasTitle
rdfs: Literal
MEETINGEVENT
hasType
xsd: boolean
EVENT
hasURL
xsd: anyURI

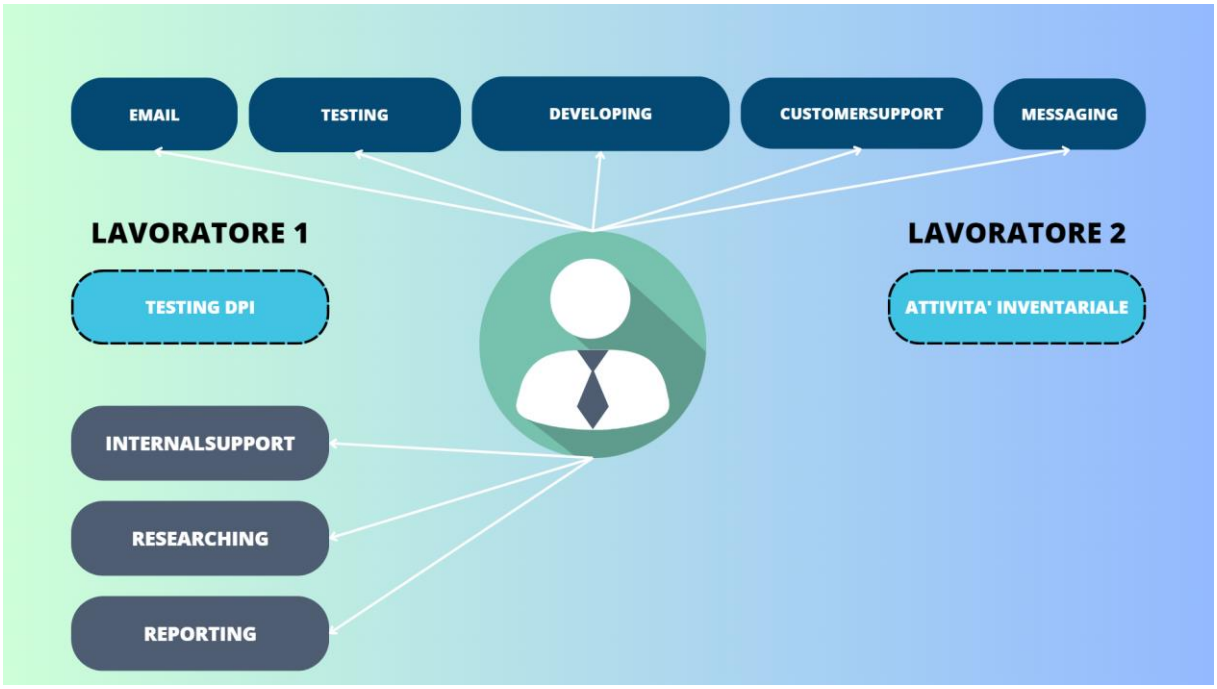

Ontologies used
TIME ONTOLOGY
OWL-Time is an OWL-2 DL ontology of temporal concepts, for describing the temporal properties of resources in the world or described in Web pages.
<http://www.w3.org/2006/time#>
FOAF
FOAF (an acronym of friend of a friend) is a machine-readable ontology describing persons, their activities and their relations to other people and objects. Anyone can use FOAF to describe themselves. FOAF allows groups of people to describe social networks without the need for a centralised database.
http://xmlns.com/foaf/0.1/

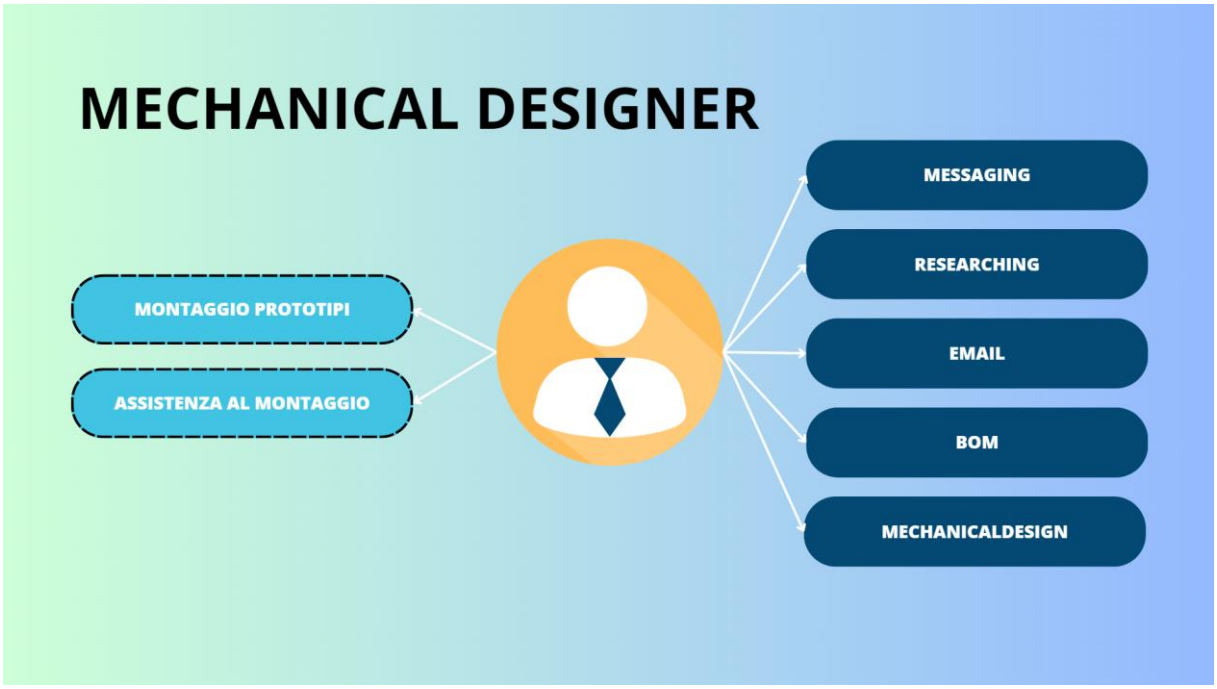
ALLEGATO II

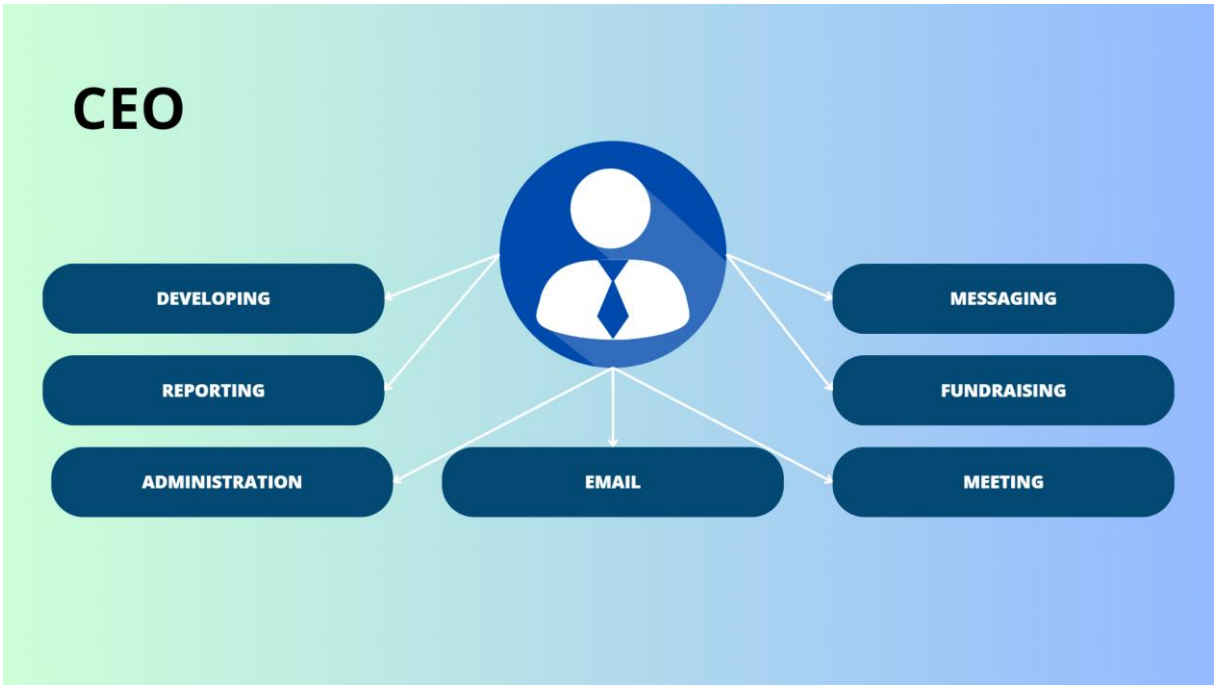
In questo allegato riporto l'esito della profilazione svolta presso l'azienda SIMAD. Nello specifico sono presenti tutti gli *ActivityType* associati ad ogni professione. Nelle ultime due diapositive sono riportati gli *ActivityType* relativi alle professioni di Manager e CEO. La profilazione di queste figure professionali è stata svolta presso la VAIMEE.

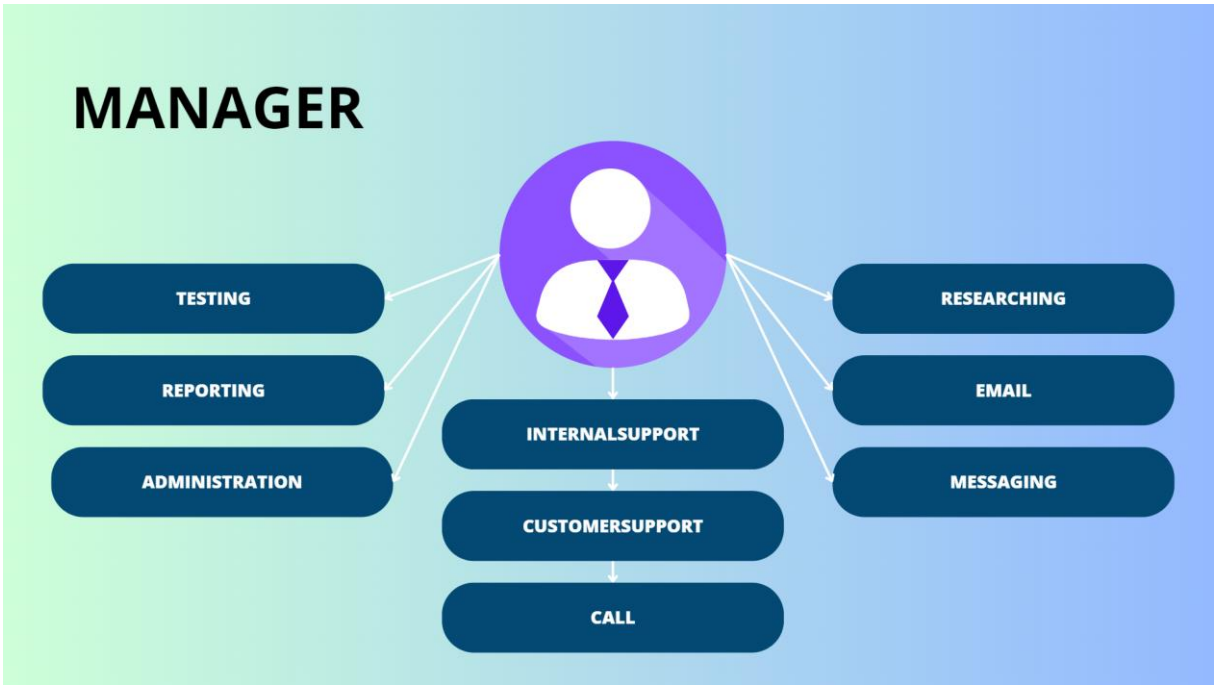

VAIMEE
SURPRISING THE FUTURE
ORGANIZZAZIONE SEMANTICA
MY2SEC
PROFILAZIONE


DEVELOPER
DEVELOPING
TESTING
CUSTOMERSUPPORT
EMAIL
MESSAGING


EMAIL
TESTING
DEVELOPING
CUSTOMERSUPPORT
MESSAGING
LAVORATORE 1
TESTING DPI
INTERNALSUPPORT
RESEARCHING
REPORTING
LAVORATORE 2
ATTIVITA' INVENTARIALE


MECHANICAL DESIGNER
MONTAGGIO PROTOTIPI
ASSISTENZA AL MONTAGGIO
MESSAGING
RESEARCHING
EMAIL
BOM
MECHANICALDESIGN


CEO
DEVELOPING
REPORTING
ADMINISTRATION
EMAIL
MESSAGING
FUNDRAISING
MEETING


MANAGER
TESTING
REPORTING
ADMINISTRATION
INTERNALSUPPORT
CUSTOMERSUPPORT
CALL
RESEARCHING
EMAIL
MESSAGING

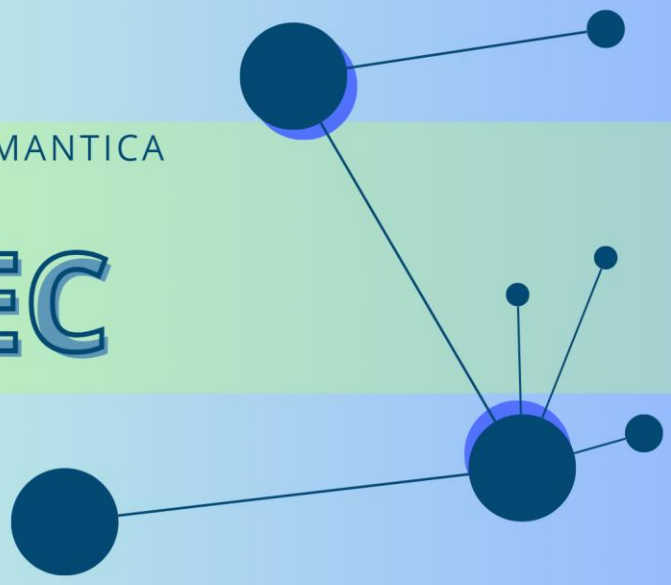
ALLEGATO III

In questo allegato riporto lo schema descrittivo dell'ontologia di My2Sec dopo l'adattamento per SIMAD. Nello specifico sono riportate tutte le classi e tutte le proprietà descritte con le relative gerarchie. Sottolineo inoltre che sono state aggiunte due professioni profilate presso VAIMEE (*Manager* e *CEO*).

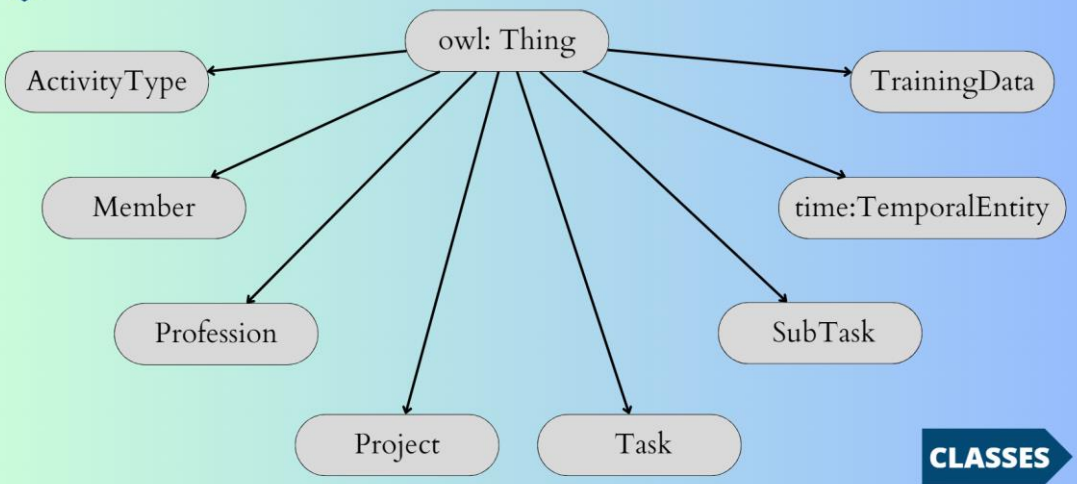
VAIMEE

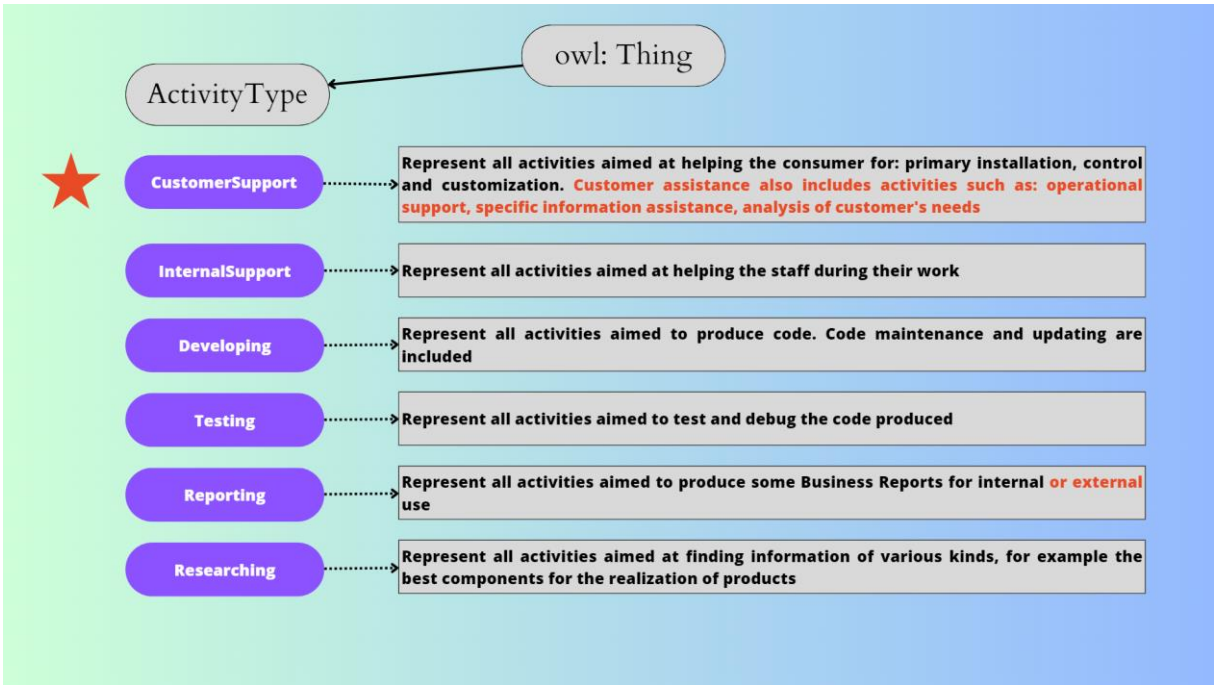
ORGANIZZAZIONE SEMANTICA

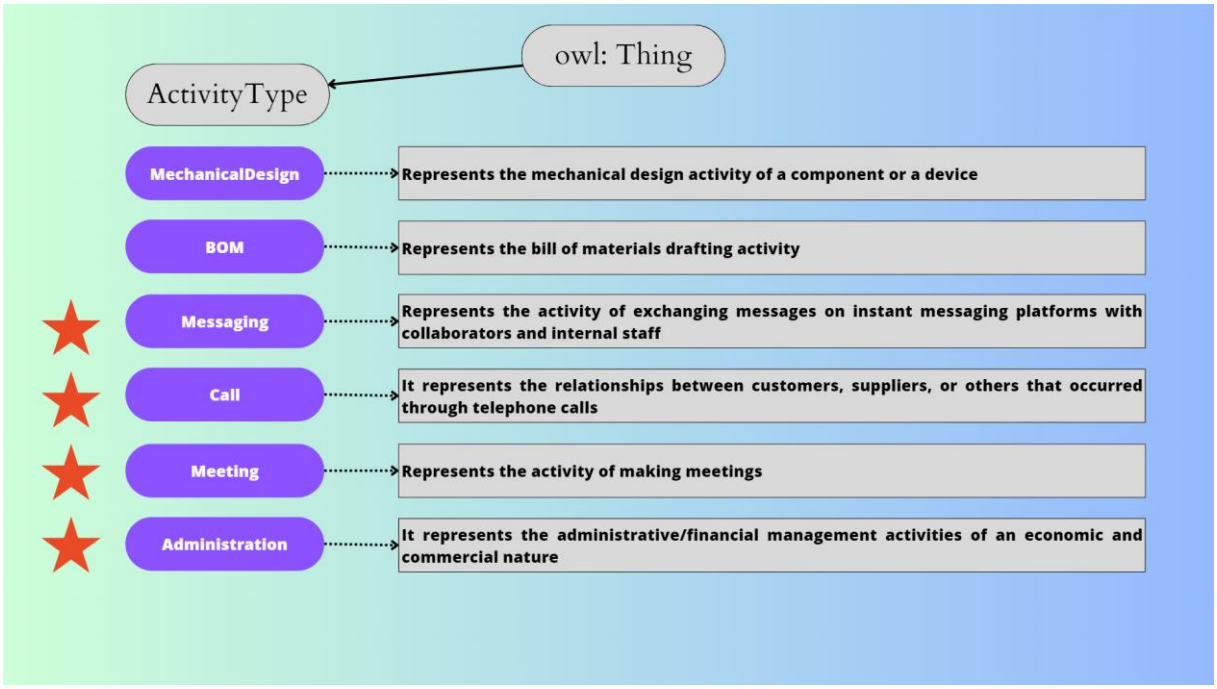
MY2SEC

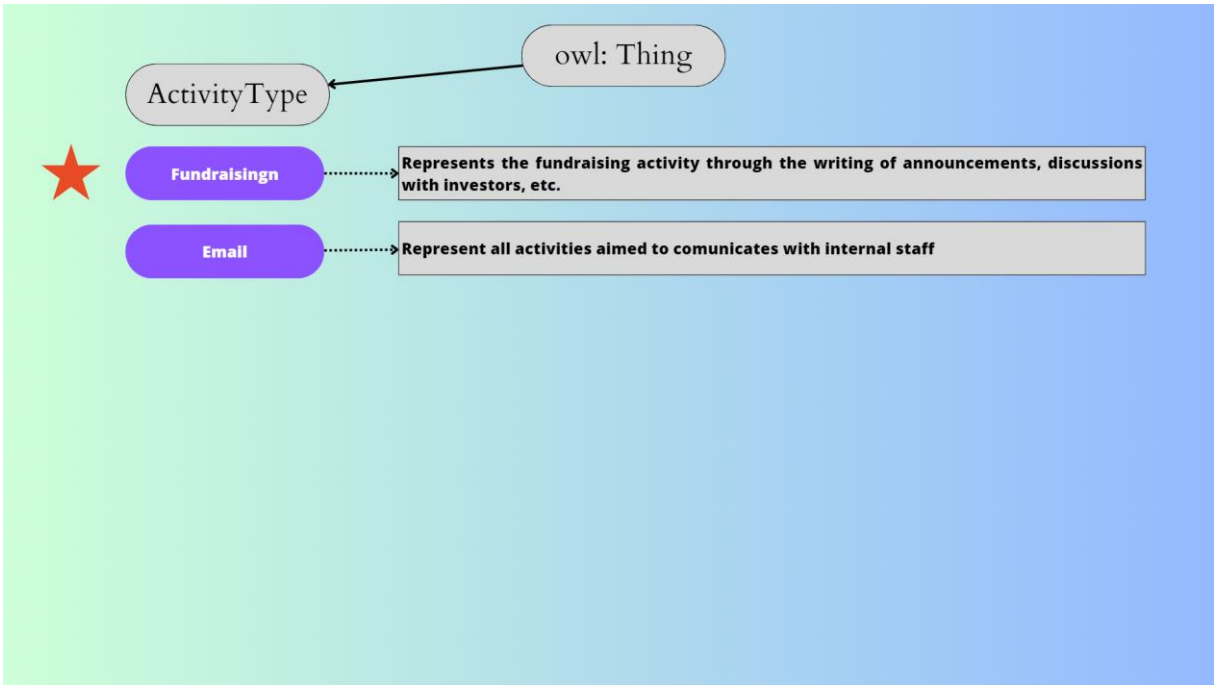


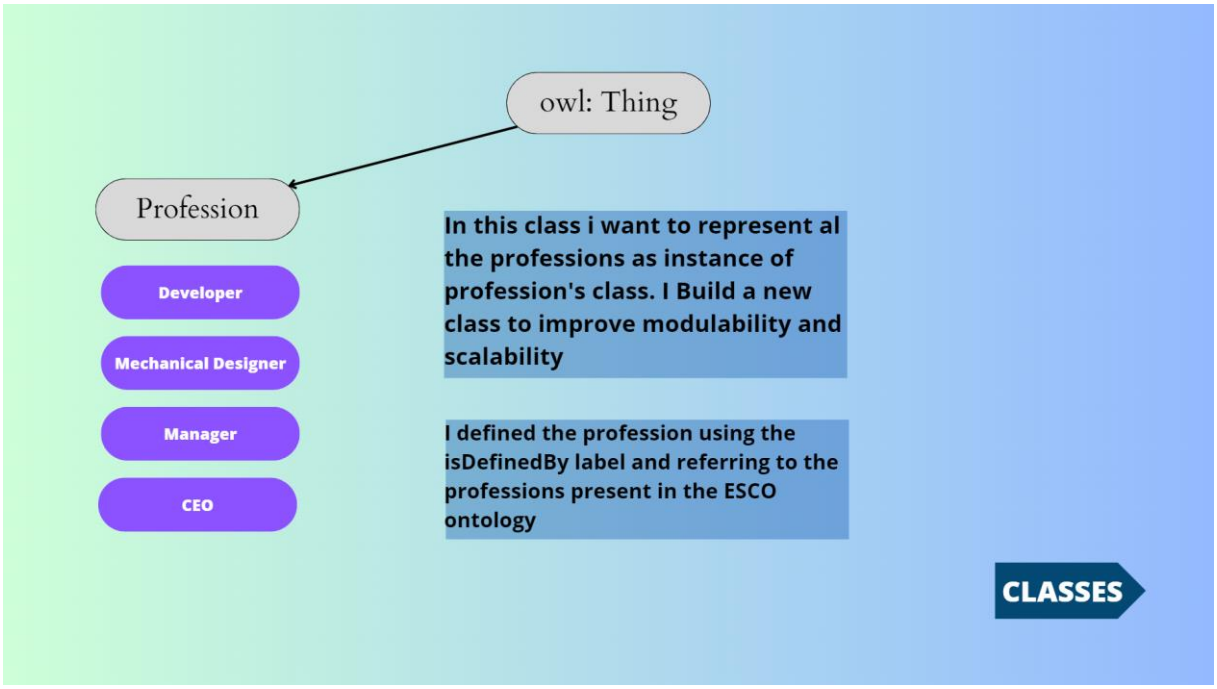
VAIMEE

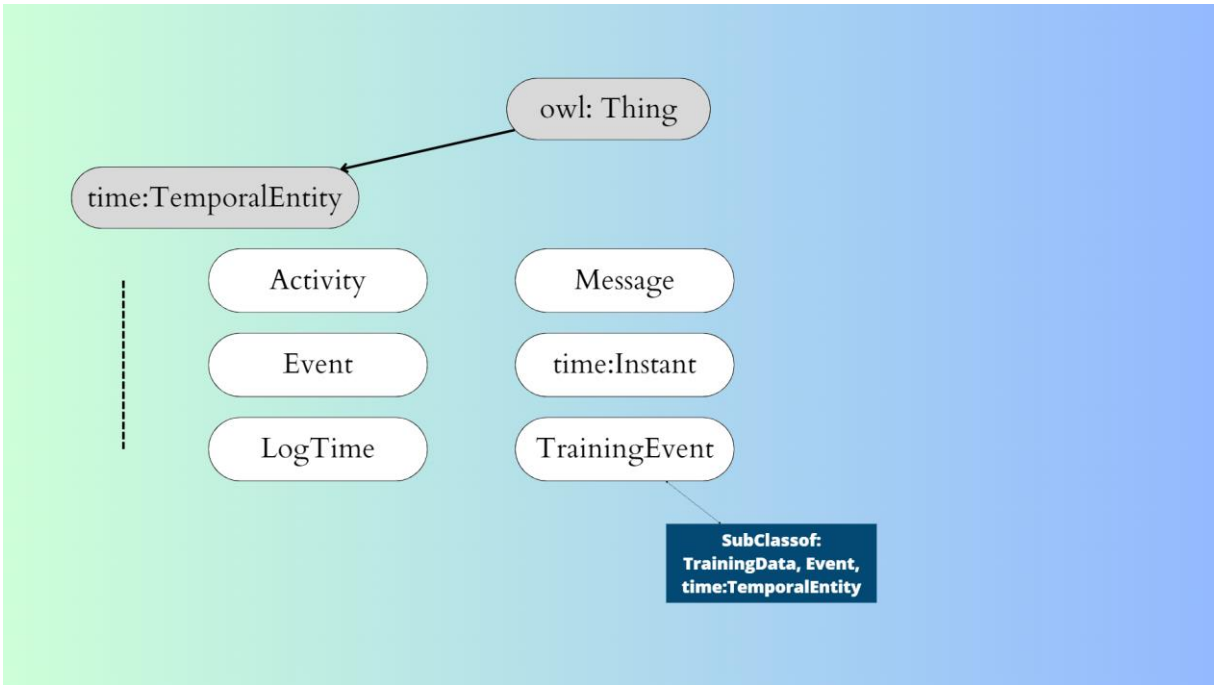


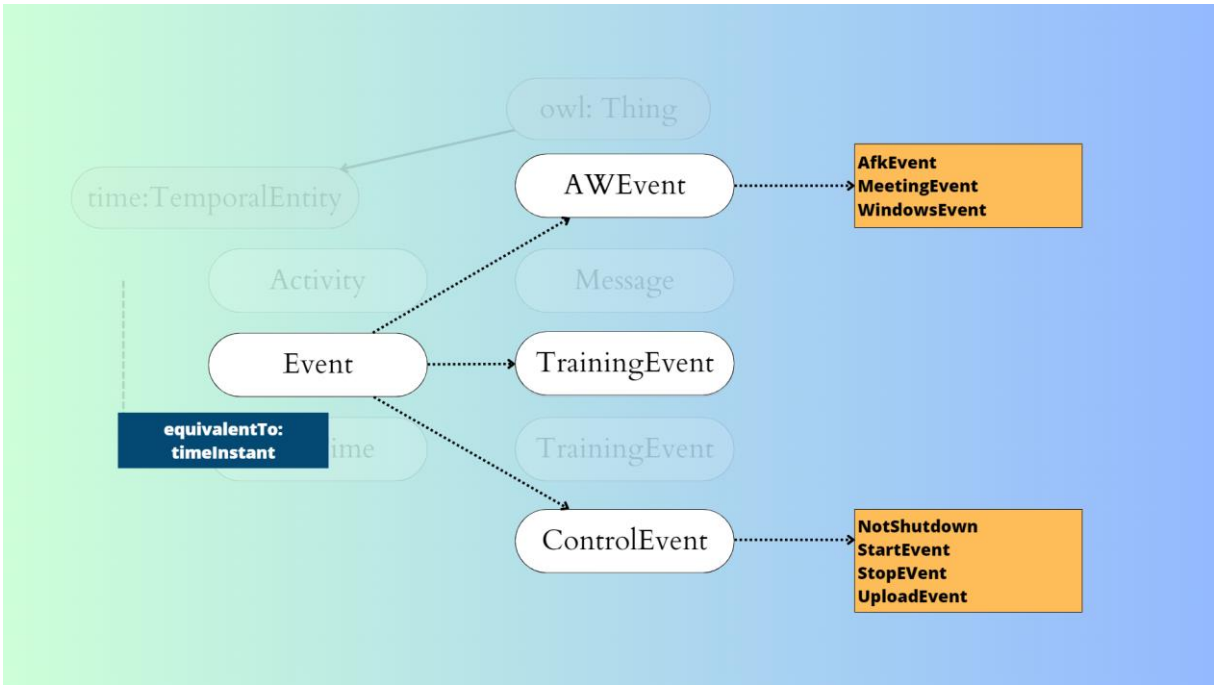

owl: Thing
ActivityType
CustomerSupport
Represent all activities aimed at helping the consumer for: primary installation, control and customization. Customer assistance also includes activities such as: operational support, specific information assistance, analysis of customer's needs
InternalSupport
Represent all activities aimed at helping the staff during their work
Developing
Represent all activities aimed to produce code. Code maintenance and updating are included
Testing
Represent all activities aimed to test and debug the code produced
Reporting
Represent all activities aimed to produce some Business Reports for internal or external use
Researching
Represent all activities aimed at finding information of various kinds, for example the best components for the realization of products

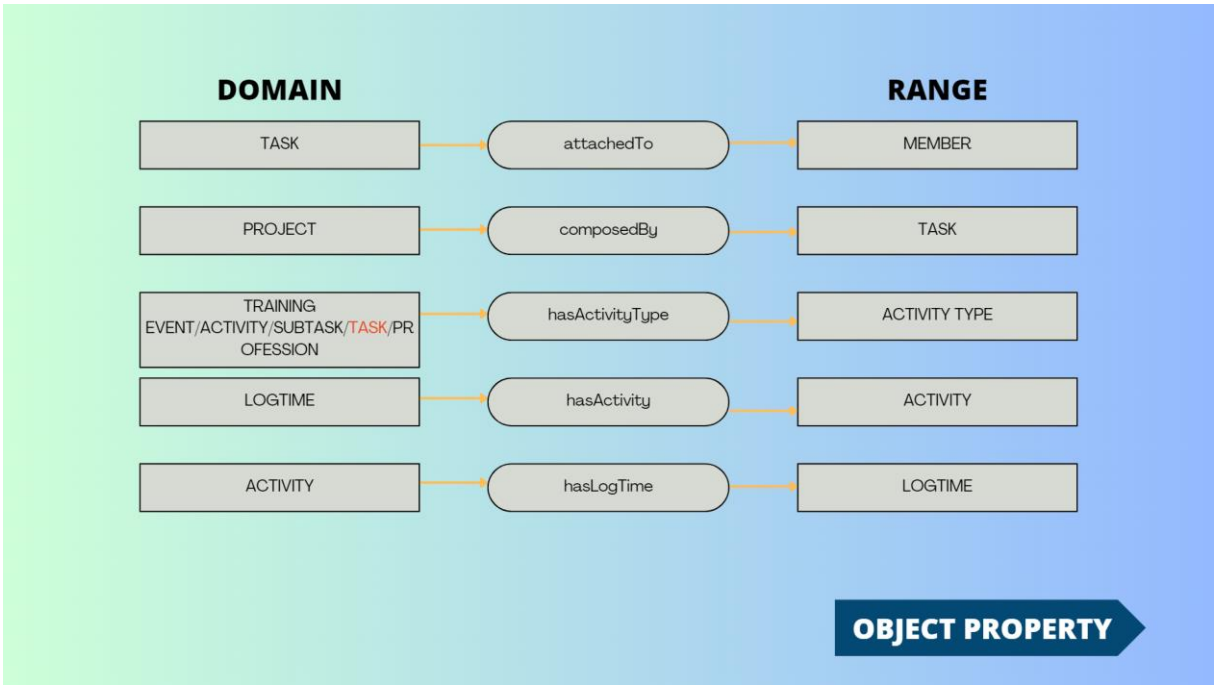

owl: Thing
ActivityType
MechanicalDesign
Represents the mechanical design activity of a component or a device
BOM
Represents the bill of materials drafting activity
Messaging
Represents the activity of exchanging messages on instant messaging platforms with collaborators and internal staff
Call
It represents the relationships between customers, suppliers, or others that occurred through telephone calls
Meeting
Represents the activity of making meetings
Administration
It represents the administrative/financial management activities of an economic and commercial nature

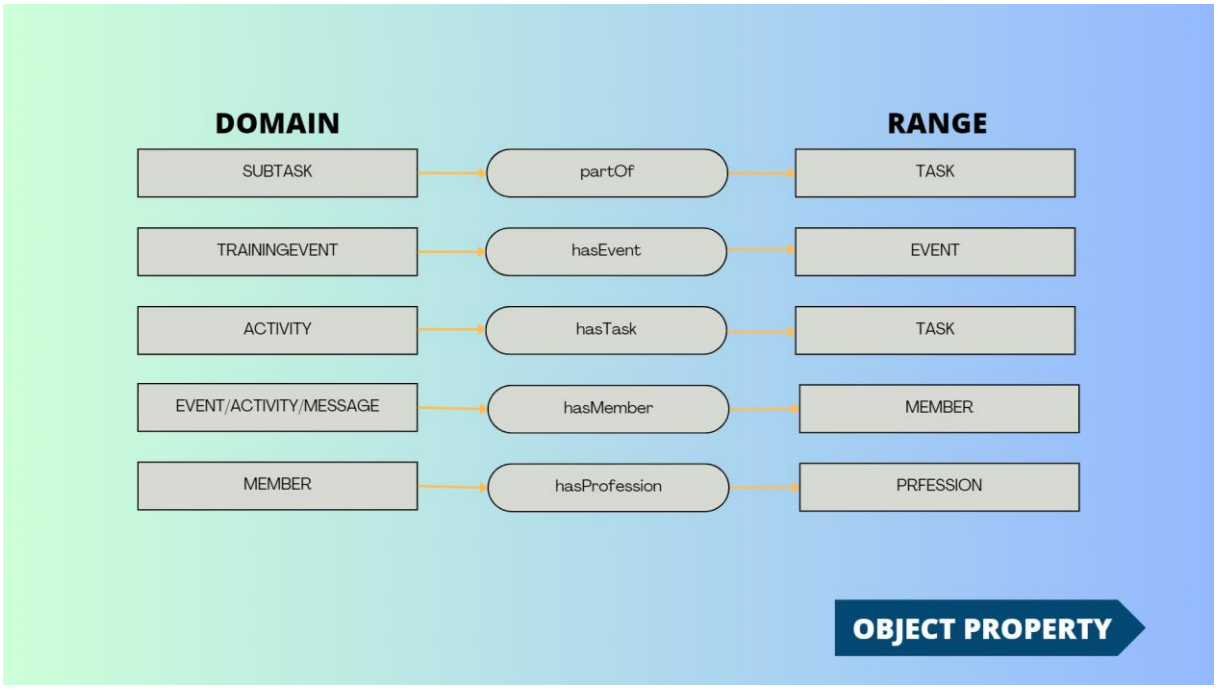

owl: Thing
ActivityType
Fundraisingn
Represents the fundraising activity through the writing of announcements, discussions with investors, etc.
Email
Represent all activities aimed to comunicates with internal staff

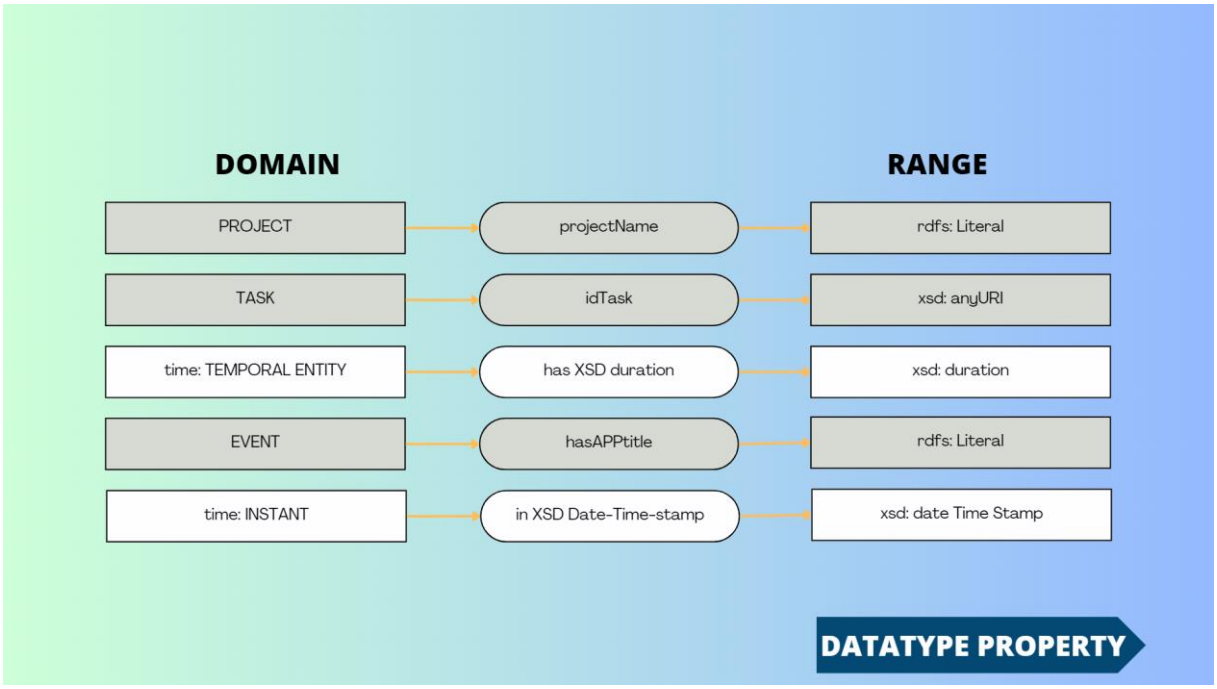

owl: Thing
Profession
Developer
Mechanical Designer
Manager
CEO
In this class i want to represent al the professions as instance of profession's class. I Build a new class to improve modulability and scalability
I defined the profession using the isDefinedBy label and referring to the professions present in the ESCO ontology
CLASSES

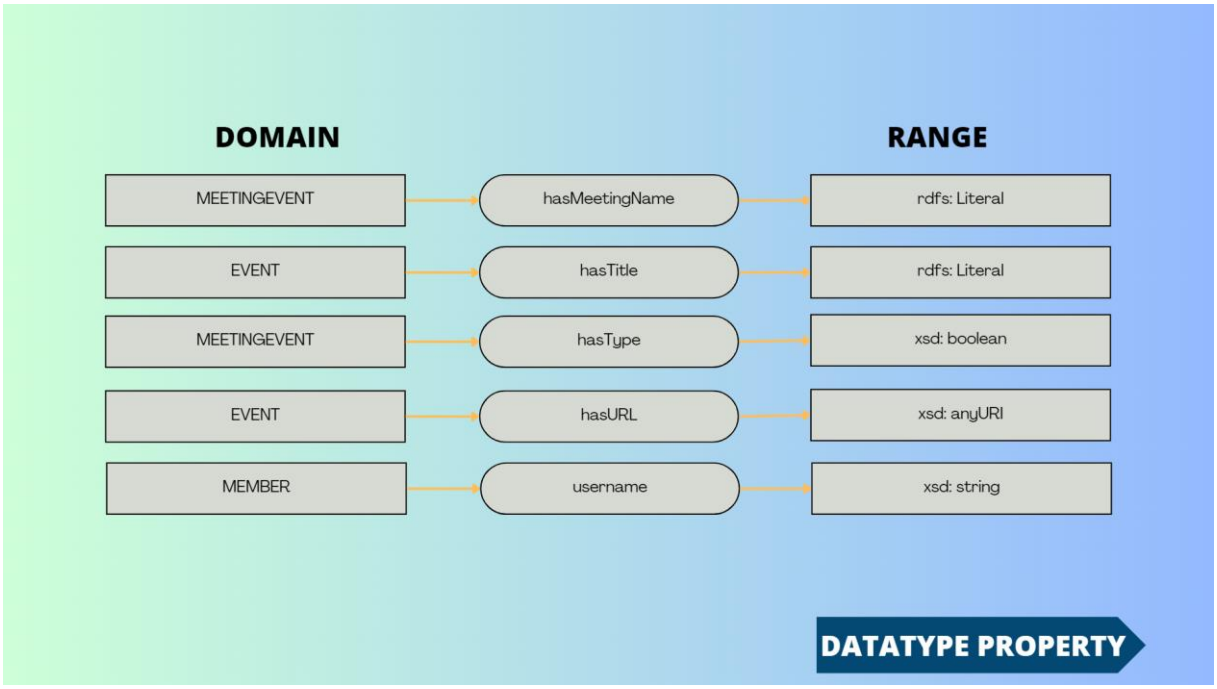

owl: Thing
time:TemporalEntity
Activity
Event
LogTime
Message
time:Instant
TrainingEvent
SubClassof:
TrainingData, Event,
time:TemporalEntity

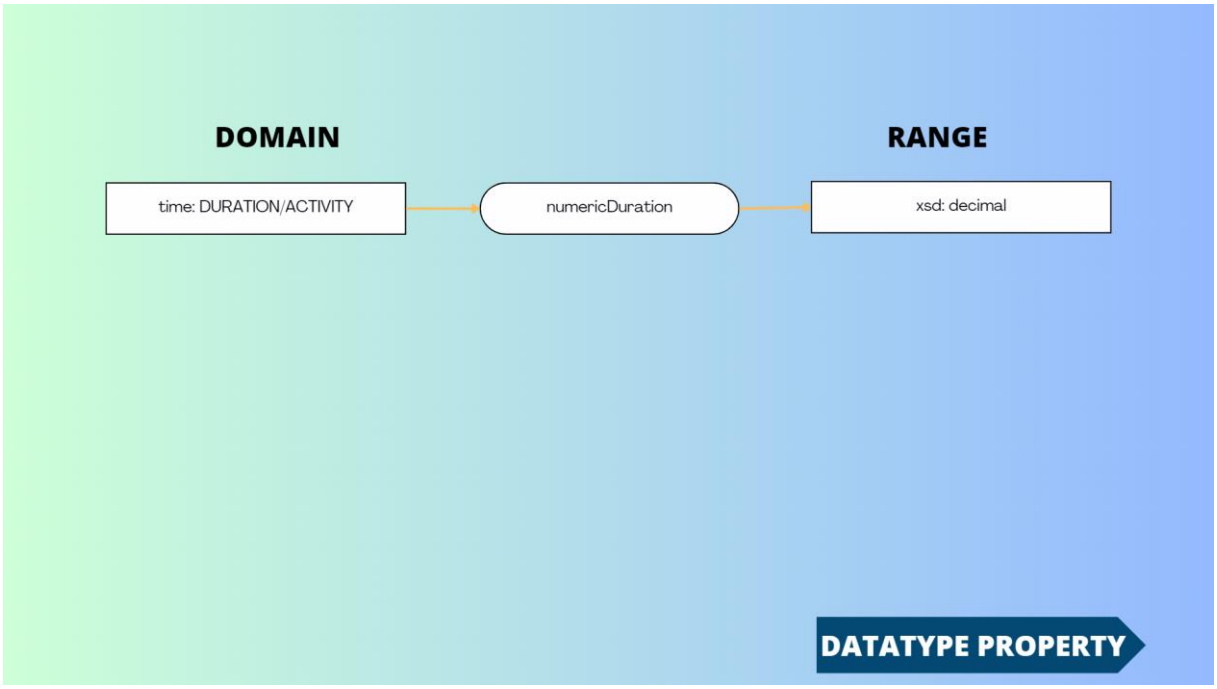

owl: Thing
time:TemporalEntity
Activity
Event
AWEvent
Message
TrainingEvent
ControlEvent
AfkEvent
MeetingEvent
WindowsEvent
NotShutdown
StartEvent
StopEVent
UploadEvent
equivalentTo:
timeInstant

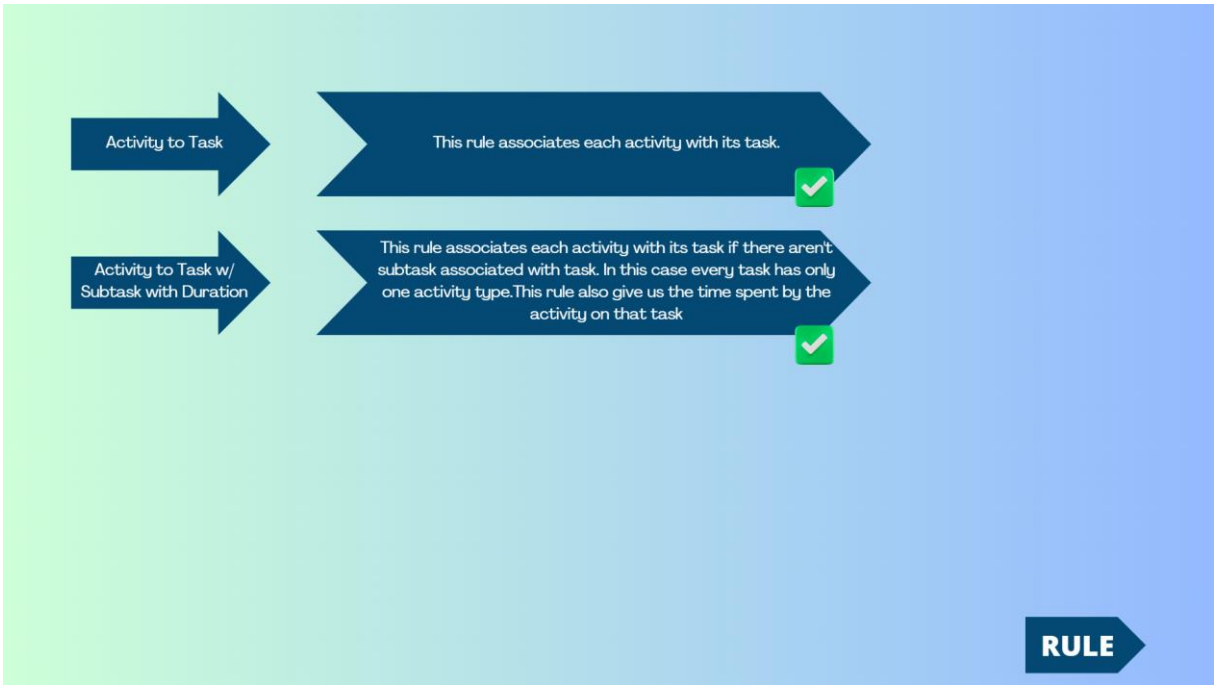

DOMAIN
RANGE
TASK
attachedTo
MEMBER
PROJECT
composedBy
TASK
TRAINING EVENT/ACTIVITY/SUBTASK/TASK/PROFESSION
hasActivityType
ACTIVITY TYPE
LOGTIME
hasActivity
ACTIVITY
ACTIVITY
hasLogTime
LOGTIME
OBJECT PROPERTY


DOMAIN
RANGE
SUBTASK
partOf
TASK
TRAININGEVENT
hasEvent
EVENT
ACTIVITY
hasTask
TASK
EVENT/ACTIVITY/MESSAGE
hasMember
MEMBER
MEMBER
hasProfession
PRFESSION
OBJECT PROPERTY


DOMAIN
RANGE
PROJECT
projectName
rdfs: Literal
TASK
idTask
xsd: anyURI
time: TEMPORAL ENTITY
has XSD duration
xsd: duration
EVENT
hasAPPtitle
rdfs: Literal
time: INSTANT
in XSD Date-Time-stamp
xsd: date Time Stamp
DATATYPE PROPERTY


DOMAIN
RANGE
MEETINGEVENT
hasMeetingName
rdfs: Literal
EVENT
hasTitle
rdfs: Literal
MEETINGEVENT
hasType
xsd: boolean
EVENT
hasURL
xsd: anyURI
MEMBER
username
xsd: string
DATATYPE PROPERTY


DOMAIN
RANGE
time: DURATION/ACTIVITY
numericDuration
xsd: decimal
DATATYPE PROPERTY


Activity to Task
This rule associates each activity with its task.
Activity to Task w/ Subtask with Duration
This rule associates each activity with its task if there aren't subtask associated with task. In this case every task has only one activity type.This rule also give us the time spent by the activity on that task
RULE

ALLEGATO IV

In questo allegato riporto il template utilizzato in fase di profilazione per analizzare i dati raccolti in azienda.

PROFILAZIONE MY2SEC

PROFILAZIONE DELLE FIGURE PROFESSIONALI COINVOLTE NEL PROCESSO DI REALIZZAZIONE DI APPARECCHIATURE BIOMEDICALI NELL'AZIENDA SIMAD

LAVORATORE	
CONTATTO	
PROFESSIONE	

DESCRIZIONE DELL'ATTIVITÀ LAVORATIVA

Nell'esercizio della sua professione, il lavoratore svolge differenti mansioni che possiamo sintetizzare come segue:

MANSIONE	DESCRIZIONE DELLE ATTIVITÀ CARATTERISTICHE DELLA MANSIONE

CATALOGAZIONE DEI DISPOSITIVI SW UTILIZZATI PER TIPO DI ATTIVITÀ

APPLICATIVO	MANSIONE 1	MANSIONE 2	MANSIONE 3	MANSIONE 4	MANSIONE 5	MANSIONE 6	MANSIONE 7	MANSIONE 8	MANSIONE 9	TEMPO DI UTILIZZO

BIBLIOGRAFIA

[1] «My2Sec - The Ultimate PM & HR Suite By Smart Workers for Smart Workers». VAIMEE, 13 marzo 2023. Consultato: 31 maggio 2023. [Online]. Disponibile su: https://github.com/vaimee/my2sec

[2] «Discorso della Presidente von der Leyen sullo stato dell'Unione», *European Commission - European Commission*. https://ec.europa.eu/commission/presscorner/detail/it/speech_22_5493 (consultato 1 luglio 2023).

[3] «Il 2023 è l'Anno europeo delle competenze», *Dipartimento per le Politiche Europee*. http://www.politicheeuropee.gov.it/it/comunicazione/notizie/anno-ue-competenze/ (consultato 1 luglio 2023).

[4] «World Wide Web», *Wikipedia*. 4 giugno 2023. Consultato: 7 giugno 2023. [Online]. Disponibile su: https://it.wikipedia.org/w/index.php?title=World_Wide_Web&oldid=133819995

[5] «Scientific American: The Semantic Web», *Sci. Am.*.

[6] «Spinning the Semantic Web : Bringing the World Wide Web to Its Full Potential». https://web-s-ebscohost-com.ezproxy.unibo.it/ehost/ebookviewer/ebook/bmxlYmtfXzgxMTI0X19BTg2?sid=1389149a-2760-44e5-b593-b1a4ad689f2f@redis&vid=0&format=EB&lpid=lp_xi&rid=0 (consultato 7 giugno 2023).

[7] «Uniform Resource Identifier», *Wikipedia*. 21 marzo 2023. Consultato: 7 giugno 2023. [Online]. Disponibile su: https://it.wikipedia.org/w/index.php?title=Uniform_Resource_Identifier&oldid=132609067

[8] «Resource Description Framework», *Wikipedia*. 10 marzo 2022. Consultato: 7 giugno 2023. [Online]. Disponibile su: https://it.wikipedia.org/w/index.php?title=Resource_Description_Framework&oldid=126198709

[9] «OWL - Standard del Web Semantico». https://www.w3.org/OWL/ (consultato 7 giugno 2023).

[10] «Filosofia dell'informazione/Ontologia - Wikibooks, manuali e libri di testo liberi». https://it.wikibooks.org/wiki/Filosofia_dell%27informazione/Ontologia (consultato 8 giugno 2023).

[11] «Ontologia», *Wikipedia*. 30 dicembre 2022. Consultato: 8 giugno 2023. [Online]. Disponibile su: https://it.wikipedia.org/w/index.php?title=Ontologia&oldid=131237305

[12] «arces-wot/SEPA». Dynamic Linked Data & Web of Things - University of Bologna, 13 giugno 2023. Consultato: 17 giugno 2023. [Online]. Disponibile su: https://github.com/arces-wot/SEPA

[13] A. Ferrari, «SPARQL Event Processing Architecture: analisi e ottimizzazione delle prestazioni», 2021.

[14] «Apache Jena - Welcome to Apache Jena». https://jena.apache.org/about_jena/ (consultato 20 giugno 2023).

[15] «Bigdata - Semantic Web Standards». https://www.w3.org/2001/sw/wiki/Bigdata (consultato 20 giugno 2023).

[16] A. Ferrari, E. Riforgiato, e L. Roffia, «JSON SPARQL Application Profile for Linked Data», in *2022 31st Conference of Open Innovations Association (FRUCT)*, Helsinki, Finland: IEEE, apr. 2022, pp. 380–384. doi: 10.23919/FRUCT54823.2022.9770884.

[17] «Regola delle 5 W», *Wikipedia*. 13 maggio 2023. Consultato: 17 giugno 2023. [Online]. Disponibile su: https://it.wikipedia.org/w/index.php?title=Regola_delle_5_W&oldid=133481437

[18] «protégé». https://protege.stanford.edu/ (consultato 18 giugno 2023).

[19] «OntoGraf - Protege Wiki». https://protegewiki.stanford.edu/wiki/OntoGraf (consultato 18 giugno 2023).

[20] «SWRLTab - Protege Wiki». https://protegewiki.stanford.edu/wiki/SWRLTab (consultato 18 giugno 2023).

[21] J. R. Hobbs e F. Pan, «An ontology of time for the semantic web», *ACM Trans. Asian Lang. Inf. Process.*, vol. 3, fasc. 1, pp. 66–85, mar. 2004, doi: 10.1145/1017068.1017073.

[22] «FOAF», *Wikipedia*. 29 dicembre 2021. Consultato: 18 giugno 2023. [Online]. Disponibile su: https://it.wikipedia.org/w/index.php?title=FOAF&oldid=124768878

[23] «SWRL: A Semantic Web Rule Language Combining OWL and RuleML». https://www.w3.org/Submission/SWRL/#8 (consultato 18 giugno 2023).

[24] «xquery cover page - W3C». https://www.w3.org/TR/xquery/ (consultato 18 giugno 2023).

[25] «xpath copertina - W3C». https://www.w3.org/TR/xpath/ (consultato 18 giugno 2023).

[26] «Open-world assumption», *Wikipedia*. 30 dicembre 2022. Consultato: 19 giugno 2023. [Online]. Disponibile su: https://en.wikipedia.org/w/index.php?title=Open-world_assumption&oldid=1130571733

[27] «Drools», *Wikipedia*. 13 dicembre 2022. Consultato: 19 giugno 2023. [Online]. Disponibile su: https://en.wikipedia.org/w/index.php?title=Drools&oldid=1127295644

[28] «protegeproject/swrlapi-drools-engine». Protégé Project, 15 febbraio 2023. Consultato: 19 giugno 2023. [Online]. Disponibile su: https://github.com/protegeproject/swrlapi-drools-engine

[29] «Expert system», *Wikipedia*. 1 aprile 2023. Consultato: 18 giugno 2023. [Online]. Disponibile su: https://en.wikipedia.org/w/index.php?title=Expert_system&oldid=1147678708

[30] «Inference engine», *Wikipedia*. 5 giugno 2023. Consultato: 18 giugno 2023. [Online]. Disponibile su: https://en.wikipedia.org/w/index.php?title=Inference_engine&oldid=1158598049

[31] «Semantic reasoner», *Wikipedia*. 12 febbraio 2023. Consultato: 19 giugno 2023. [Online]. Disponibile su: https://en.wikipedia.org/w/index.php?title=Semantic_reasoner&oldid=1138959512

[32] «What is Semantic Reasoning? | Fundamentals with Oxford Semantic Technologies». https://www.oxfordsemantic.tech/fundamentals/what-is-semantic-reasoning (consultato 19 giugno 2023).

[33] «Backward chaining», *Wikipedia*. 26 marzo 2023. Consultato: 19 giugno 2023. [Online]. Disponibile su: https://en.wikipedia.org/w/index.php?title=Backward_chaining&oldid=1146694660

[34] «Forward chaining», *Wikipedia*. 5 giugno 2023. Consultato: 19 giugno 2023. [Online]. Disponibile su: https://en.wikipedia.org/w/index.php?title=Forward_chaining&oldid=1158598217

[35] «Rule Based Systems».

[36] «0004370282900200.htm».

[37] «L'Azienda», *SIMAD*. https://simad-xray.com/azienda/ (consultato 19 giugno 2023).

[38] «What is ESCO?» https://esco.ec.europa.eu/en/about-esco/what-esco (consultato 20 giugno 2023).

[39] «Java Tutorial». https://www.w3schools.com/java/default.asp (consultato 2 luglio 2023).

[40] «OWL 2 Web Ontology Language Primer (Second Edition)».

https://www.w3.org/TR/owl2-primer/ (consultato 2 luglio 2023).

[41] «Java HashMap». https://www.w3schools.com/java/java_hashmap.asp (consultato 2 luglio 2023).

[42] «Java HashSet». https://www.w3schools.com/java/java_hashset.asp (consultato 2 luglio 2023).

[43] «Grafana Cloud | Observability platform overview», *Grafana Labs*. https://grafana.com/products/cloud/ (consultato 6 luglio 2023).

[44] «Mastering JBoss Drools 6: Discover the Power of Drools 6 and Business Rules for Developing Complex Scenarios in Your Applications - ProQuest». https://www.proquest.com/docview/2148288440/bookReader?accountid=9652 (consultato 8 luglio 2023).