



Ricerca di Sistema elettrico

Ontologia per la Governance dei Dati Urbani Energetici

Michela Milano, Federico Chesani, Paola Mello, Marco Patella

Ontologia per la Governance dei Dati Urbani Energetici

Michela Milano(Università di Bologna), Federico Chesani(Università di Bologna), Paola Mello (Università di Bologna), Marco Patella (Università di Bologna)

Aprile 2021

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA

Piano Triennale di Realizzazione 2019-2021 - II annualità

Obiettivo: Tecnologie

Progetto: Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali

Work package: Local Energy District

Linea di attività: WP1 – LA22

Responsabile del Progetto: Claudia Meloni, ENEA

Responsabile del Work package: Claudia Meloni, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione *"Ontologia del Framework per la Governance dei Dati Urbani Energetici"*.

Responsabile scientifico ENEA: Angelo Frascella

Responsabile scientifico Unibo : Michela Milano

Indice

SOMMARIO.....	4
1 INTRODUZIONE.....	6
2 DESCRIZIONE DELLE ATTIVITÀ SVOLTE E RISULTATI	9
2.1 ONTOLOGIE, VOCABOLARI, E MODELLI DEI DATI PER SMART CITIES	9
2.1.1 <i>Smart Appliances REference (SAREF)</i>	9
2.1.2 <i>FIWARE data model</i>	13
2.2 EVOLUZIONE DEL MODELLO ONTOLOGICO	17
2.2.1 <i>Proprietà opzionali</i>	18
2.2.2 <i>Unità di misura</i>	18
2.2.3 <i>Aggregazione temporale e spaziale</i>	21
2.2.4 <i>Versionamento</i>	22
2.3 MODIFICHE SOFTWARE.....	23
2.3.1 <i>Allineamento</i>	23
2.3.2 <i>Nuove funzionalità</i>	27
2.3.3 <i>Mantenimento</i>	28
3 CONCLUSIONI.....	32

Sommario

Il presente report documenta il lavoro fatto nel corso della Linea di Attività LA22 del WP Local Energy District, relativo seconda annualità del piano triennale di realizzazione 2019-2021. Tale lavoro riprende il lavoro del precedente triennio relativo allo sviluppo di strumenti semantici per l'interoperabilità della Smart City. Questi strumenti sono entrati a far parte delle specifiche SCPS (Smart City Platform Specification), prodotti da ENEA nello scorso triennio¹.

L'idea centrale delle SCPS è la realizzazione di una piattaforma software in grado di raccogliere dati e dataset di un distretto urbano al fine di creare servizi che, sfruttando queste informazioni, possano supportare le municipalità nella scelta e pianificazione di interventi volti a incrementare l'efficienza energetica e la qualità della vita di una Smart City.

Una delle azioni chiave per raggiungere tale risultato è la descrizione delle informazioni provenienti dai diversi ambiti organizzativi di un distretto come un edificio, un singolo appartamento o una strada. Per facilitare e uniformare lo scambio di informazioni è stata ipotizzata la realizzazione di uno strumento in grado di descrivere le informazioni da scambiare secondo una semantica condivisa. A tal fine negli anni precedenti è stata realizzata un'ontologia sulla base delle tecnologie del web semantico, che permetta una uniformità di ricerca dei dati e una organizzazione uniforme delle conoscenze al fine di semplificare lo scambio degli stessi.

Intorno a questa ontologia, sviluppata in linguaggio OWL, è stata poi costruita una suite di strumenti software in grado permettere l'accesso alle informazioni presenti nell'ontologia, di generare schemi di messaggi per lo scambio di informazioni e verificare che i messaggi da scambiare sulla piattaforma siano stati realizzati in maniera conforme alle definizioni delle informazioni. Infine, è stato realizzato un servizio web che in grado di fornire un facile accesso alle precedenti applicazioni.

Dopo che il modello semantico è stato sviluppato nel precedente triennio, si è passati da una fase prototipale delle SCPS all'uso in contesti reali con le città. Questa interazione con gli stakeholder ha messo in evidenza la necessità di aggiornare tutto il framework e, all'interno di esso, gli strumenti semantici.

In particolare sono venute fuori le seguenti esigenze:

- Aggiornare lo stato dell'arte per verificare se, rispetto al precedente triennio ci siano nuove iniziative di cui tener conto
- Far evolvere l'ontologia in modo da tener conto di una serie di esigenze provenienti dal mondo reale. Il risultato di questa attività è stato un modello ontologico esteso e migliorato, alla luce delle nuove specifiche semantiche, diventando così maggiormente performante e rappresentativo delle esigenze delle città.
- Far evolvere di conseguenza gli strumenti software semantici. Questo ha implicato l'estensione del software per gestire le nuove caratteristiche dell'ontologia e l'aggiornamento dell'interfaccia di visualizzazione e generazione dei template.

¹ Per i dettagli si vedano i report RdS/PAR2015/019, RdS/PAR2016/002 e RdS/PAR2017/040

Quest'ultimo aspetto, in particolare, si è rivelato particolarmente impegnativo, ma anche importante in quanto propedeutico alla creazione di una interfaccia di editing degli UrbanDataset (che non richieda di passare dall'OWL per definire nuovi UD) che sarà oggetto della LA 23.

Oltre alle precedenti attività sono stati risolti i bug dei servizi che riscontravano problemi nel loro funzionamento online ed è stata semplificata e arricchita l'interfaccia utente. Infine, il look&feel di tutte le applicazioni sviluppate in precedenza è stato allineato e uniformato con il resto delle pagine web di progetto.

1 Introduzione

La keyword “Smart City” viene spesso usata per indicare una delle misure prioritarie per affrontare la problematica energetico-ambientale propria di una città, ovvero il luogo in cui si concentra il maggiore consumo di risorse energetiche poiché vi si concentra l’attività insediativa, produttiva e di massimo impatto sull’ambiente.

L’approccio Smart City mira, quindi, al raggiungimento di traguardi di abbattimento dei consumi energetici (in primo luogo elettrici) molto più consistenti di quelli ottenuti finora attraverso strategie basate essenzialmente sulla sostituzione di componenti con altri “a maggiore efficienza energetica”. Il principio organizzativo da utilizzare è quello del “resource on demand”. Tale approccio richiede però una tecnologia di sistema avanzata che coinvolge e integra: i) una sensoristica urbana e sistemi di interazione per comprendere esattamente la necessità dell’utente, ii) sistemi di trasmissione e raccolta integrata dei dati (cloud urbani), iii) sistemi a elevata intelligenza, per analisi, diagnostica, elaborazione e ottimizzazione che fondono dati provenienti da diversi canali informativi, e infine iv) servizi urbani capaci di adattare il proprio comportamento in base ai dati ricevuto da altri ambiti.

Il principale obiettivo di una Smart City sta nella capacità di mettere insieme gli elementi energetico-ambientali e quelli di carattere sociale (come la consapevolezza energetica, la partecipazione e coesione sociale e la qualità della vita) attraverso l’uso di tecnologie e di applicazioni e sfruttando l’interconnessione tra reti, ottenendo lo sviluppo di “servizi innovativi multifunzionali” partendo dalle conoscenze messe a disposizione degli enti. Ovvero attraverso l’elaborazione di dataset contenenti informazioni utili allo sviluppo di servizi grazie all’integrazione di dati precedentemente separati e non pubblici, e quindi allo sviluppo di nuovi servizi dalla creazione di nuova conoscenza derivante dall’integrazione delle diverse sorgenti.

Il presente lavoro si inserisce nel Work Package WP1 “Local Energy District” del progetto 1.7 Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali. Il principale obiettivo del WP1 consiste nello sviluppo di un modello di “distretto urbano intelligente” che coniughi aspetti tecnologici e aspetti sociali, finalizzati al miglioramento dei servizi erogabili ai cittadini in quanto più efficienti dal punto di vista energetico e funzionale. L’attività si focalizza sullo sviluppo integrato di infrastrutture pubbliche urbane, sistemi per la modellazione e gestione della rete energetica del distretto (smart district), sistemi centralizzati per l’analisi dei dati provenienti dalle abitazioni con interfaccia dialogativa utente (smart homes service) e sistemi di supporto alle decisioni per la valutazione del rischio del patrimonio edilizio e delle infrastrutture.

In particolare uno degli obiettivi è la realizzazione di **Servizi Urbani Smart**, cioè la digitalizzazione, ottimizzazione ed integrazione in ottica smart di tutte le infrastrutture del distretto che hanno un impatto sulla efficienza, qualità ed innovazione del servizio elettrico.

All’interno di questo task si sviluppa il lavoro condotto dal Dipartimento di Informatica – Scienza e Ingegneria (DISI) dell’Università di Bologna.

Il lavoro svolto consiste nell’implementazione dei nuovi requisiti emersi dall’analisi dello stato dell’arte e dall’applicazione delle Smart City Platform Specification (SCPS) in casi reali per poter far evolvere lo stato di interoperabilità semantica implementato nel precedente triennio. Si ricorda che lo scopo di tale stato semantico sia quello di facilitare la selezione e aggregazione dei dati provenienti dai diversi ambiti applicativi e di abilitare l’implementazione di nuovi servizi basati su

algoritmi intelligenti, ma anche di favorire lo scambio di informazioni tra applicazioni eterogenee. Infatti, i dati che confluiscono sulla piattaforma cittadina di raccolta dati possono provenire da fornitori diversi che gestiscono parti diverse dell'infrastruttura e raccolgono informazioni provenienti da vari contesti della città o del distretto e mantenuti secondo formati o modalità di memorizzazione eterogenei.

Lo scopo dell'ontologia è proprio quello di armonizzare tali differenze al fine di facilitare l'analisi delle informazioni permettendo una più facile elaborazione automatica. Il fine è di progettare un'infrastruttura che limiti i tempi di sviluppo e, al tempo stesso, faciliti il lavoro di estensione dell'ontologia per favorire una implementazione futura di nuovi servizi da integrare nella piattaforma della Smart City.

L'ontologia definita nello scorso triennio ha mostrato diversi limiti nell'applicazione nei casi reali. L'ontologia infatti mancava di alcune caratteristiche importanti quali, per esempio:

- La possibilità di gestire unità di misura alternative per una stessa proprietà. L'ontologia definita nel precedente triennio, infatti, prevedeva una sola unità di misura associata in modo fisso alla proprietà
- La possibilità di gestire unità, multipli e sottomultipli non presenti nell'ontologia di riferimento. L'ontologia definita alla fine dello scorso triennio infatti poteva utilizzare esclusivamente quelle presenti nell'ontologia OM2²
- La possibilità di avere all'interno di uno specifico UrbanDataset proprietà (cioè tipologie di dati) opzionali, e quindi la conseguente capacità di definire dei veri profili di UrbanDataset. L'ontologia definita nello scorso triennio infatti vincolava il set di proprietà associate a un UrbanDataset (per esempio l'UrbanDataset relativo alle condizioni meteo) impedendo da una parte la sua espansione nel caso una città dovesse aver bisogno di altre proprietà non presenti (per esempio i mm di pioggia caduti) o obbligando a inviare campi vuoti nel caso le proprietà non fossero rilevate dai suoi sensori

I requisiti mancanti dall'ontologia necessitavano di essere inseriti in essa in maniera da non stravolgerla e sfruttando le caratteristiche presenti.

Oltre all'ontologia, sono necessari anche strumenti volti a favorire l'utilizzo delle informazioni presenti. Per questo motivo, già nello scorso triennio, erano state sviluppate applicazioni che forniscono:

- template di riferimento per l'implementazione dei messaggi in formato elettronico (JSON o XML) usati per lo scambio di informazioni tra gli attori che operano sulla piattaforma
- artefatti per validare la correttezza formale (sintassi) e semantica dei messaggi
- un'applicazione web in grado di fornire accesso a questi strumenti, oltre che visualizzare il contenuto dell'ontologia e navigare i vari concetti in modalità human-friendly.

Come si intuisce, ciascuna dei requisiti aggiunti nell'ontologia impatta fortemente sull'insieme degli strumenti software, richiedendo un gran numero di sviluppi aggiuntivi

Nel seguito del documento sono descritte:

² [Link all'ontologia OM2](#)

- lo studio dello stato dell'arte confrontato con l'ontologia dei Dati Urbani (nel capitolo 2.1)
- l'evoluzione del modello ontologico effettuata nel corso del progetto (nel capitolo 2.2). in seguito al confronto e da nuove specifiche.
- gli sviluppi effettuati per aggiornare il set degli applicativi software in modo da rendere utilizzabile la nuova versione dell'ontologia (capitolo 2.3). Tali sviluppi sono propedeutici allo sviluppo di un software di editing dell'ontologia che sarà effettuata nella prossima annualità
- le conclusioni del lavoro svolto (capitolo 3).

2 Descrizione delle attività svolte e risultati

Il primo step dell'attività svolta è stato uno studio dello stato dell'arte ragionato focalizzato sulle ontologie e semantiche esistenti per la governance di dati urbani, a cui è seguita un'analisi volta a confrontare le ontologie individuate e quella sviluppata nell'ambito delle specifiche SCPS nel precedente triennio. Inoltre si è analizzata la possibilità di interagire con ontologie descrittive unità di misura e si è definito un possibile approccio per poterle integrare. Da queste analisi e dai feedback ricevuti durante il precedente triennio è stato stilato un piano di modifica dell'ontologia al fine di armonizzarla e semplificarne l'estensione. Tali modifiche hanno richiesto necessariamente la modifica e l'aggiornamento dell'applicativo web per il consulto dell'ontologia al fine di poter mostrare le nuove funzionalità. Queste operazioni sono state giudicate necessarie, oltre per il fine ultimo di preservare il servizio di consulto, anche per il futuro sviluppo di un tool di editing online dell'ontologia stessa. Di fatto il tool di editing gioverà delle modifiche apportate proprio perché l'operazione di modifica dipende logicamente dalla possibilità di poter visualizzare lo stato corrente del modello dei dati.

Di seguito nel capitolo 2.1 verrà riportato lo stato dell'arte comparato focalizzato su due soluzioni comuni in ambito smart cities: Fiware e SAREF. Nel capitolo 2.2, invece, verranno mostrate le modifiche apportate all'ontologia ed infine, nel capitolo 2.3 le modifiche software apportate all'applicazione Web.

2.1 Ontologie, vocabolari, e modelli dei dati per Smart Cities

Nell'ambito delle Smart Cities l'integrazione di diverse fonti di dati permette la valorizzazione delle strutture hardware e software garantendo applicazioni sempre più smart e immersive. Tale processo è spesso minato dalla mancanza di interoperabilità tra i diversi servizi e formati dati. È il problema dei *data silos* dove le informazioni sono intrappolate all'interno di un particolare dominio applicativo e sono di fatto inaccessibili ad altri servizi. Per esempio, le informazioni sulla salute strutturale di un ponte non sono accessibili a servizi di monitoraggio traffico che potrebbero avvertire gli automobilisti di un pericolo imminente di crollo. Per questo è importante stabilire un modello dati aperto, semanticamente descritto, e interpretabile da degli agenti software.

In questo documento analizzeremo diverse proposte ontologiche al fine di valutare miglioramenti nella modello degli UrbanDataset. Lungi da essere un'analisi omnicomprensiva delle ontologie e vocabolari presenti, questa analisi si focalizza nello studio dei vocabolari più diffusi confrontandone l'approccio con la l'ontologia delle Smart City Platform Specification. In primis ci focalizzeremo sull'ontologia Smart Appliances REference (SAREF) e FIWARE data model.

2.1.1 Smart Appliances REference (SAREF)

L'ontologia SAREF(<https://w3id.org/saref#>), come ci suggerisce il suo nome, si specializza nella descrizione di apparecchi domestici intelligenti. Nasce da un consenso distribuito tra vari produttori di dispositivi, come:

- Sensori domotici (temperatura, umidità, misuratori di consumo energetico, etc.)
- Attuatori domotici (finestre, porte, etc.)
- Elettrodomestici vari (lavatrici, frigoriferi, caldaie, etc.)

- Prodotti per la pulizia (aspirapolveri robot)
- Prodotti per la cucina automatica
- Sistemi di ventilazione e di aria condizionata
- Sistemi di illuminazione
- Impianti per la produzione di micro energia rinnovabile (i.e., pannelli solari)

Oltre agli obiettivi di interoperabilità tra i dispositivi presentati qui sopra, il progetto SAREF vuole proporsi anche come framework per la gestione dei consumi energetici. Grazie a questa funzionalità può anche essere usato come modello dati per l'ottimizzazione dell'efficienza energetica di case private e di edifici pubblici.

Il concetto centrale nell'ontologia in esame è il Dispositivo (saref:Device): un oggetto fisico progettato per soddisfazione di una o più funzionalità domestiche. Tali funzionalità sono concretizzate nel concetto di Funzione (saref:Function) il quale descrive, per l'appunto, una funzionalità necessaria per raggiungere l'obiettivo per cui dispositivo è stato creato. SAREF offre nativamente alcune funzionalità generali che possono essere poi specializzate per i propri bisogni. Tra di esse troviamo: Funzionalità di acquisizione (saref:Sensing_function), Funzionalità di misura (saref:Metering_function) e Funzionalità di notifica o allarme (saref:Event_function). Ogni funzionalità può essere attivata o raggiunta attraverso l'esecuzione di uno o più Comandi (saref:Command) i quali possono influire sullo Stato (saref:State) di un apparecchio oppure lasciarlo invariato. Per esempio, un comando On attiva un elettrodomestico agendo sul suo stato On/Off mentre il comando Acquisisci semplicemente esegue la direttiva per leggere un valore da un sensore.

Oltre a questi concetti "core" l'ontologia SAREF definisce altre classi satellite utili per poter descrivere e ottimizzare i consumi di un dispositivo. Tra essi citiamo il Profilo (saref:Profile) e le Proprietà (saref:Property: qualsiasi valore che possa essere acquisito, misurato o controllato in un edificio domestico) Energia (saref:Energy) e Potenza (saref:Power). Le istanze di tali proprietà possono essere assegnate ad un apparecchio per comunicare il suo consumo medio previsto.

Per quanto riguarda invece la classe Profilo è utile per profilare nel tempo il consumo da parte di un dispositivo di una particolare Proprietà. Come accennato in precedenza solitamente questa riguarda l'Energia ma la classe rende possibile anche tenere traccia di qualsiasi fornitura casalinga (saref:Commodity) come acqua e gas. Infatti in un'istanza di Profilo potremmo serializzare quale saref:Property o saref:Commodity è stata analizzata, per quanto tempo è stata osservata e qual è il suo prezzo stimato.

Infine SAREF introduce anche il concetto di Service, il quale ha lo scopo di fare da ponte tra i comandi fisici (i.e., cliccare un bottone per accendere un dispositivo) e quelli digitali (i.e., accendere un dispositivo da remoto usando un remote procedure call).

Per concludere la descrizione di SAREF la Figura 1 riassume i concetti mostrati fin ora e le loro relazioni. Per un quadro completo del modello dei dati fare riferimento alla pagina di documentazione³.

³ <https://saref.etsi.org/core/v3.1.1/>

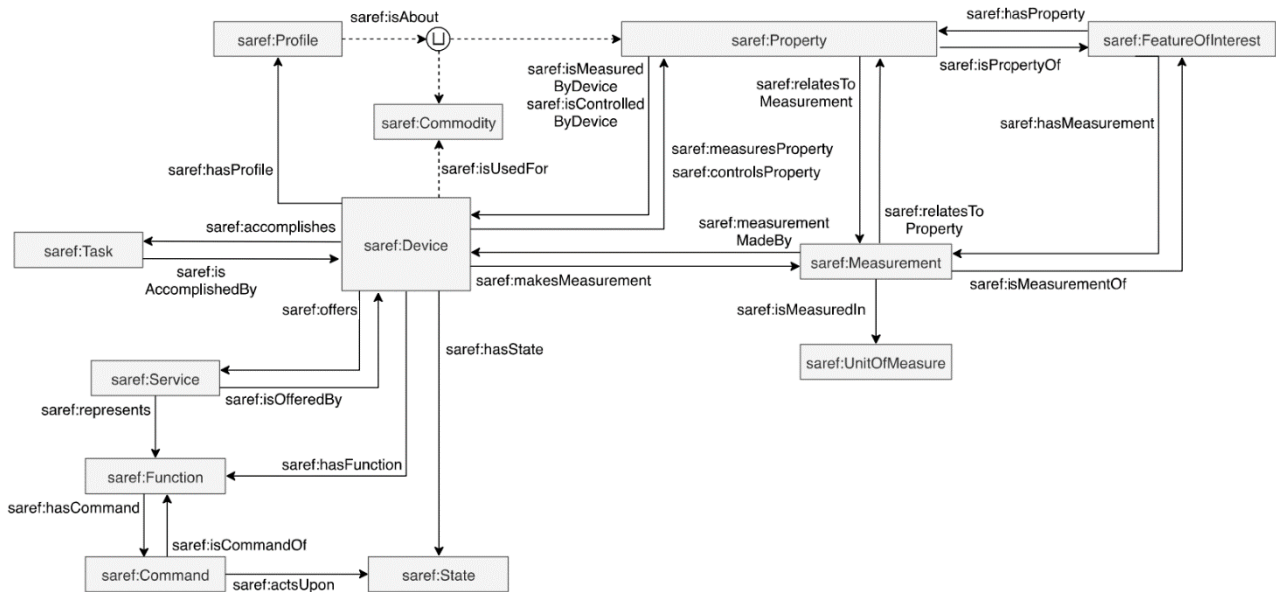


Figura 1 - Schema delle classi e delle relazioni dell'ontologia SAREF

Confrontando SAREF con l'ontologia SCPS possiamo notare come i domini applicativi di riferimento possano intersecarsi in alcuni casi d'uso. Per esempio nella descrizione del consumo energetico di un edificio smart. Possiamo quindi definire un possibile piano d'integrazione dell'ontologie che tenga conto dei vari punti di forza e mantenga quanto più possibile intatte le definizioni esistenti. Il primo step di questo lavoro potrebbe essere quello di valutare SAREF come ontologia per la descrizione dei dispositivi in gioco in una smart city. Nell'ontologia ci sono vari concetti atti a relazionare un data set con il dispositivo che l'ha generato. Tali proprietà sono:

- *smartcityplatform:enea:semantic:ontology:urbandataset#SensorID*
- *smartcityplatform:enea:semantic:ontology:urbandataset#SensorName*
- *smartcityplatform:enea:semantic:ontology:urbandataset#SensorTypeCode*
- *smartcityplatform:enea:semantic:ontology:urbandataset#ApplianceCode*
- *smartcityplatform:enea:semantic:ontology:urbandataset#DeviceIDn*

Un'idea per allineare le due ontologie potrebbe essere quella di aggiungere una proprietà tipo *SAREFDevice* allo scopo di coprire lo stesso significato semantico di *SensorID*, *DeviceID*, *SensorName* e *SensorTypeCode* e un'altra per *ApplianceCode* (i.e., *SAREFAppliance*). Queste due proprietà permetterebbero di integrare descrizioni più dettagliate di dispositivi facendo uso del ricco vocabolario definito da SAREF. Sarebbe consigliabile introdurre queste proprietà come informazioni aggiuntive rispetto a quelle già definite nell'ontologia, onde evitare problemi di retrocompatibilità. Infatti se i software legacy utilizzassero *SAREFDevice* in alternativa a *SensorID*, non sarebbero in grado di interpretare la nuova istanza di UD e richiederebbero degli aggiornamenti.

Un altro punto di integrazione si potrebbe trovare nell'allineamento di alcuni concetti dell'ontologia SCPS con la classe SAREF Profile. Questa, come menzionato in precedenza, serve proprio a tenere traccia di consumi e costi relativi ad un determinato dispositivo. Nell'ontologia SCPS troviamo il dataset *ApplianceElectricConsumption* che segue lo stesso scopo. Per rafforzare l'integrazione tra le due ontologie sarebbe bene definire *ApplianceElectricConsumption* come

sottoclasse di Profile permettendo così a software basati su SAREF l'interpretazione semantica di tale dato.

Continuando con i punti in comune tra le due ontologie troviamo la definizione delle unità di misura. Entrambi le ontologie supportano l'associazione di una unità di misura ad un valore. In SAREF questo viene fatto tramite una proprietà chiamata *isMeasuredIn* che ha come range istanze della classe *UnitOfMeasure*, mentre in SCPS viene riutilizzata la proprietà d'annotazione di *Ontology of Measure (OM)*⁴ chiamata *hasUnit*. Nonostante le differenze di nomenclatura e modellazione, le due proprietà sono simili in quanto SAREF *UnitOfMeasure* è equivalente per definizione (*isDefinedBy*) ad unità di misura OM. Per cui sia SAREF che SCPS si avvalgono della unità di misura in OM e ne utilizzano i concetti in modo da poter contestualizzare i valori misurati. Al contempo, SAREF preferisce categorizzare le unità di misura OM in sottoclassi utili per poter descrivere brevemente misurazioni contestualizzate nel suo campo applicativo. Infatti definisce: *EnergyUnit*, *IlluminanceUnit*, *Currency*, *TemperatureUnit*, *Commodity*, *PressureUnit*, *PowerUnit*, *TemporalUnit*.

Infine è bene ricordare che, anche se vi è una sovrapposizione di vari concetti, le due ontologie hanno strutture diverse. Infatti SAREF sceglie un approccio Dispositivo centrico, mentre l'ontologia SCPS mette in primo piano i dati acquisiti e loro struttura lasciando in secondo piano la descrizione l'oggetto fisico che li ha generati. Per cui una completa interoperabilità tra i due modelli è difficilmente raggiungibile, ma come accennato in precedenza si potrebbe iniziare un lavoro di allineamento dei due mondi attraverso un modulo ontologico specifico e dichiarazioni complementari. Tale *ontologia di allineamento* andrebbe a contenere le proprietà sopra definite e degli statement owl che ne definiscano l'utilizzo all'interno di dataset. Sarebbe per cui possibile utilizzare questo modulo a seconda delle necessità per poter integrare anche dati presi da SAREF.

⁴ <https://www.ebi.ac.uk/ols/ontologies/om>

2.1.2 FIWARE data model

FIWARE è un'iniziativa open source atta a definire un insieme di standard universali per la gestione di dati contestuali allo scopo di facilitare lo sviluppo di soluzioni smart per diversi domini applicativi come Smart Cities, Smart Industry, Smart Agrifood e Smart Energy. In ogni piattaforma smart c'è necessità di collezionare e gestire informazioni sul contesto dei dati, processarle e notificare attori esterni abilitandoli ad agire con lo scopo di arricchire a loro volta la base di dati. Focalizzandosi su questi obiettivi FIWARE propone un agente software fondamentale chiamato "Context Broker" il quale si occupa dell'aggiornamento e smistamento dello stato contestuale condiviso tra i vari "FIWARE agents".

L'architettura di FIWARE abilita la comunicazione asincrona fra le diverse componenti della soluzione software che la sfrutta. Tuttavia spesso questo non è sufficiente a garantire un'interoperabilità completa tra le parti. Infatti è importante definire un modello di dati condiviso al fine di permettere la mutua comprensione tra applicazione diverse. Per questo la fondazione FIWARE definisce anche il FIWARE data model, un insieme di specifiche descrivente la struttura e il significato dei dati trasmessi nella piattaforma.

Il FIWARE data model è contenuto in una serie di repository GitHub raggruppati nell'organizzazione Smart Data model⁵. Il data model è suddiviso in Topic a cui è assegnato un singolo repository. Un Topic descrive un particolare insieme di dati "verticali" che può essere riutilizzato in un contesto più ampio. Correntemente in FIWARE data model sono definiti i seguenti Topic:

- Allarmi
- Ambiente
- Punto di interesse
- Tracking di problemi civili
- Illuminazione stradale
- Dispositivi
- Trasporti
- Indicatori
- Gestione rifiuti
- Parcheggio
- Meteo

I vari Topic poi possono essere aggregati in altri repository di "dominio" i quali costituiscono una guida per gli sviluppatori che vogliono creare applicazioni od utility per tale ambito. Un esempio è il repository di Ambiente Smart dove sono raggruppati i Topic Ambiente, Gestione rifiuti e Meteo. Ovviamente i vari Topic possono essere utilizzati contemporaneamente in più repository di dominio, aumentando quindi l'interoperabilità tra i domini orizzontali utilizzando concetti comuni e ben definiti. Un Topic è definito da una serie di documenti e templates organizzati in cartelle. In Figura 2 è riportato un esempio di definizione di un Topic. La struttura sintattica del dato viene descritta attraverso un JSONSchema serializzato poi in YAML o in JSON a seconda dei casi. Inoltre vengono forniti dei file di markdown come documentazione per gli utilizzatori del tipo di dato. Infine vengono aggiunti dei file di esempio in JSON standard oppure in JSON-LD nel caso gli sviluppatori volessero optare per una serializzazione basata sui LinkedData.

⁵ <https://github.com/smart-data-models/>

- specs/
 - NewModel/
 - doc/
 - `spec.md`: A data model description based on the data model template, e.g. `spec.md` of WeatherObserved.
 - `README.md`: A summary file (as an extract from the spec file), e.g. `README.md` of WeatherObserved
 - `schema.json`: The JSON Schema definition, e.g. `schema.json` of WeatherObserved
 - `example.json`: One or more JSON example file, e.g. `example.json` of WeatherObserved
 - `example-normalized.json`: One or more JSON example file in NGSi v2 normalized format, e.g. `example-normalized.json` of WeatherObserved
 - `example-normalized-ld.jsonld`: One or more JSON example file in NGSi-LD normalized format, e.g. `example-normalized-ld.jsonld` of WeatherObserved

Figura 2 - Struttura di file e cartella per la creazione di un nuovo Fiware Data Model Topic

Seppure FIWARE data model supporti concetti basati sul Semantic Web, non è del tutto chiaro dove venga definita l'ontologia di riferimento in formato owl o similari. Quello che è presente, ad oggi, è un file di contesto⁶ JSON-LD come sostituto per tale mancanza. Tuttavia tale file contiene in maniera indiscriminata tutti i concetti definiti nei vari Topic. A tal proposito, vista l'organizzazione dei file di definizione, sarebbe più consono dividere questo file di contesto nei repository di appartenenza in modo da aver una descrizione più modulare e contestualizzata.

Una caratteristica interessante dell'approccio dei FIWARE data model è la possibilità di estendere il modello modificando o aggiungendo nuovi Topic. Per farlo, vengono definite delle guidelines⁷ che, in breve, propongono un processo basato su GitHub flow⁸. In pratica, si crea una copia del repository che si intende modificare, si applicano le modifiche (e.g., si aggiunge un nuovo JSONSchema o si modifica quello esistente) e si crea una richiesta di pull al repository padre. In questa richiesta poi si potrà partecipare ad una discussione pubblica dove poter valutare l'annessione della modifica nella documentazione ufficiale.

Comparando il FIWARE data model e l'ontologia SCPS è possibile fare un parallelismo tra il concetto di Topic e UD. Entrambi rappresentano l'unità minima per lo scambio di informazioni tra i due sistemi e possono essere specializzati in varie sotto classi. Tuttavia, da un'analisi più approfondita, si può notare come i Topic di FIWARE vogliono descrivere un insieme più ampio di concetti. Infatti, se UD si focalizza più sui dati di tipo statistico e sensoristico, i Topic possono anche essere specializzati per la descrizione di apparati e oggetti fisici. Un esempio è il Topic Dispositivi dove all'interno sono riutilizzati i concetti di SAREF per la delimitazione delle funzionalità e della struttura di apparecchi smart. Per tale motivo la struttura di base di un Topic è in generale molto minimale, infatti si rifà alla nozione NGSi di Entità. Al contrario un UD ha una struttura lievemente più rigida, la quale non ne permette l'uso al di fuori dell'ambito definito in precedenza.

Un'altra differenza tra Topic e UD si trova nella definizione delle loro sotto classi. Infatti FIWARE propone una soluzione "più" lasca fermandosi alla definizione di una struttura sintattica e lasciando il livello semantico all'interpretazione umana (anche se un livello basilare di semantica viene introdotto tramite descrizione dei concetti, unità di misura, ecc.). SCPS invece segue un approccio più rigoroso definendo i vari concetti presentati in un'ontologia di riferimento

⁶ <https://schema.lab.fiware.org/ld/context>

⁷ <https://fiware-datamodels.readthedocs.io/en/latest/guidelines/index.html#how-to-contribute>

⁸ <https://guides.github.com/introduction/flow/>

formalizzata in linguaggio OWL e, parallelamente, fornendo della documentazione “human-readable”. Potrebbe essere comunque interessante valutare a fondo il modello contributivo di FIWARE e adattarlo ad SCPS. In questo modo si otterrebbe un certo livello di flessibilità e trasparenza per gli utilizzatori della piattaforma.

Per quanto riguarda i concetti comuni definiti nei due vocabolari si possono individuare le voci descritte in Tabella 1. La tabella serve anche come mapping tra Topics e UD evidenziandone differenze e punti comuni. Si tratta comunque di una comparazione che fotografa lo stato attuale, visto che sia l’insieme degli UD che quello dei topic sono dinamici e pensati per crescere nel tempo

Tabella 1 Comparazione tra modello FIWARE e Ontologia SCPS

Topic	UD	Differenze	Similitudini
Weather Forecast	Weather Forecast	<ul style="list-style-type: none"> - Formato - posizione - Temperatura percepita - Direzione vento - Massimali e minimali giornalieri - UV index - Data di creazione - Data di pubblicazione 	<ul style="list-style-type: none"> - Temperature - Pressure - Periodo
Weather Observed	Weather Condition	<ul style="list-style-type: none"> - Riferimento Device vs Nome stazione - Tendenza pressione - Livello luce vs Radiazione solare - Neve 	
Device	-	<p>Concetto non esistente come entità a se in SCPS. Alcune sue proprietà sono “sparse” in diverse UDS Properties.</p>	<ul style="list-style-type: none"> - ID Sensore - Nome dispositivo
Three-phase alternating current measurement	Counter Reading	<ul style="list-style-type: none"> - Classe generica per ogni misura di sistemi trifase - Misurazioni in entrata ed uscita 	<ul style="list-style-type: none"> - Reactive power - Active power - Apparent power - Voltage - Current

Topic	UD	Differenze	Similitudini
		<ul style="list-style-type: none"> - Frequenza - Total harmonic distortion of electrical current - Misurazioni totali ma non su singola linea 	
Streetlight cabinet	control Counter Reading	<p>Rappresenta un oggetto fisico non la misurazione. Inoltre fornisce informazioni anche per:</p> <ul style="list-style-type: none"> - Total harmonic distortion of electrical current - CosPhi - Energia consumata 	<p>Allo stesso tempo fanno riferimento a quantità comuni quali:</p> <ul style="list-style-type: none"> - Reactive power - Active power - Apparent power - Voltage
Building	Building Records	<ul style="list-style-type: none"> - Piani - Descrizione generica del posto in cui si trova - Proprietario - Ore di apertura 	<ul style="list-style-type: none"> - Indirizzo - Posizione geografica - Tipologia

Volendo approfondire la comparazione tra i due vocabolari, la seguente tabella mostra un possibile “porting” di un istanza del Topic “Three-phase alternating current measurement” ad un istanza del UrbanDataset “Counter Reading”. L’idea è quella di mostrare come sia possibile importare i dati prelevati da un sistema bastato su Fiware ad uno basato sull’ontologia sviluppata negli scorsi anni.

Tabella 2 - confronto fra campi del Topic Fiware e UrbanDataset

Campo del topic	Campo UrbanDataset
id	Id
type	Uri
name	Name
location	coordinates
address	PODID
dateModified	period?
dateCreated	period?
totalActivePower	TotalActivePower
totalReactivePower	TotalReactivePower
totalApparentPower	TotalApparentPower
activePower	Phase(1,2,3)ActivePower
apparentPower	Phase(1,2,3)ApparentPower
reactivePower	Phase(1,2,3)ReactivePower
powerFactor	Phase(1,2,3)PowerFactor
phaseVoltage	PhaseVoltage(1,2,3)
current	LineCurrent(1,2,3)

La tabella riporta solo concetti comuni alle due rappresentazioni facendo corrispondere proprietà semanticamente equivalente al loro corrispettivo. Notare come FIWARE utilizzi sotto proprietà per tenere traccia dei valori nelle varie fasi mentre l'ontologia di riferimento preferisca utilizzare un proprietà specifica per ogni fase (esempio: Phase1ActivePower).

2.2 Evoluzione del modello ontologico

L'evoluzione del modello ontologico ha toccato vari aspetti riguardanti la modellazione di un Urban Dataset. In particolare, l'operazione di aggiornamento si è focalizzata nei seguenti punti:

- Definizione di proprietà opzionali all'interno di un Urban Dataset
- Definizione di unità di misura personalizzate e aggiunta di unità di misura alternative di una singola proprietà
- Ridefinizione delle modalità di aggregazione di Urban Dataset

Nei prossimi sotto capitoli descriveremo nel particolare le modifiche apportate e per finire trarremo qualche considerazione sulla gestione delle versioni dell'ontologia.

2.2.1 Proprietà opzionali

Ricordiamo che l'ontologia definisce ciascun tipo di UrbanDataset come lista di proprietà e/o sotto-proprietà che determinano le informazioni che compongono quel tipo di dato e che la struttura dell'ontologia realizzata nel precedente triennio richiedeva che tutte le proprietà di un dato Urban Dataset dovessero essere riportate nei relativi messaggi.

Una delle prime modifiche del modello ontologico è stata quella riguardante la possibilità di poter dichiarare proprietà opzionali all'interno di un tipo UrbanDataset,

ovvero di poter dichiarare quali proprietà della lista debbano essere sempre presenti e quali possano essere eventualmente omesse. Per implementare questa funzionalità sono state analizzate due opzioni progettuali: inserire una DataProperty "isOptional" oppure introdurre una nuova ObjectProperty "hasOptionalUrbanDatasetProperty".

La prima soluzione è stata scartata poiché troppo verbosa: non essendo "l'opzionalità" una caratteristica di genere di una data proprietà. (una proprietà può essere allo stesso tempo opzionale per un UrbanDataset ma obbligatoria per un altro) si sarebbe reso necessario utilizzare una DataProperty per descrivere il comportamento desiderato e creare un individuo per ogni proprietà per poi poter annotare con "isOptional".

Al contrario, la seconda soluzione, ovvero quella adottata, ha permesso di riutilizzare la struttura presente (i.e., un UrbanDataset è descritto dalla ObjectProperty hasUrbanDatasetProperty con range UrbanDataset e domain UrbanDatasetProperty) ed estendere le query per catturare anche proprietà associate con "hasOptionalUrbanDatasetProperty".

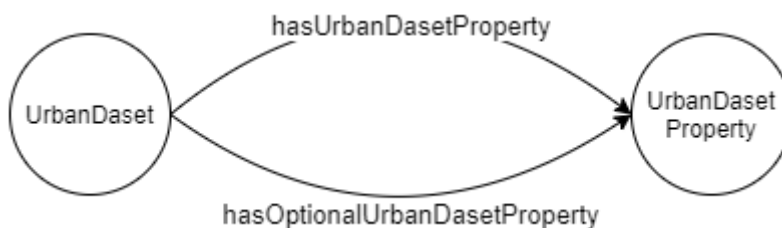


Figura 3 - Dettaglio delle nuove relazioni tra le classi UrbanDataset e UrbanDatasetProperty

2.2.2 Unità di misura

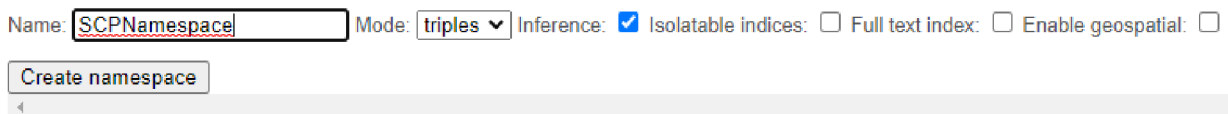
Un altro punto d'estensione discusso durante l'analisi dello stato dell'arte e sull'uso della piattaforma è quello sulla estensibilità dell'unità di misura.

L'ontologia SCP si affida ad un vocabolario esterno per la definizione dell'unità di misura: Ontology of units of Measure (OM); sostanzialmente l'obiettivo prefissato è stato quello di dare la possibilità agli operatori di aggiungere nell'ontologia SCPS unità di misura non disponibili in OM.

Il raggiungimento di questo obiettivo, quindi, si è concretizzato nella definizione del processo di estensione di OM con nuovi assiomi o istanze. In generale l'estensione di un'ontologia può avvenire in due modi in OWL: usando semplicemente la direttiva *prefix* o importando l'ontologia. Le due tecniche deferiscono rispetto all'utilizzo finale dell'ontologia risultante. Infatti se l'ontologia "da estendere" deve poi essere consultata e caricata in una Knowledge Base (KB), l'operazione più appropriata è quella di *import*. La direttiva **owl:import** dichiara che l'ontologia corrente conterrà anche tutti i concetti di quella importata. In sostanza, nel nostro caso d'uso

significa includere completamente OM nell'ontologia SCPS. Al contrario, se si ha bisogno solo di far riferimento ai concetti esterni senza però doverli interrogare o esplorare si può usare un prefisso che indica la provvidenza di certi vocaboli. Questo permette di non appesantire l'ontologia risultante ma comunque di poter utilizzare e estendere l'ontologia importata.

Nel caso d'uso in esame OM non solo deve poter essere estesa ma si richiede che sia anche consultabile nella KB poiché contiene informazioni utili per la descrizione finale dell'unità di misura. Per esempio se usassimo la direttiva **prefix** e dichiarassimo che una DataSetProperty ha come unità di misura **om:degreeCelsius** non sarebbe possibile recuperare il simbolo oppure la descrizione di tale istanza. Per cui per implementare l'aggiunta di una nuova unità di misura nell'ontologia SCPS si è deciso di optare per l'utilizzo della direttiva **owl:import**. È importante notare che per sfruttare tale funzionalità in uno sparql endpoint, esso deve avere un minimo supporto all'inferenza per poter riconoscere **owl:import** e automaticamente caricare l'ontologia OM nel database. In blazegraph ⁹(database RDF e SPARQL endpoint) il motore d'inferenza è attivabile alla creazione di un namespace come mostrato in Figura 1. Un altro punto d'attenzione quando si carica una ontologia con **owl:import** è che l'inclusione delle triple avviene scaricando il documento passato come argomento. Quindi all'inizializzazione della KB è necessario essere connessi alla rete.



Name: Mode: Inference: Isolatable indices: Full text index: Enable geospatial:

Figura 4 - Attivazione del motore di inferenza in Blazegraph. Notare la checkbox in blu

Una volta importata l'ontologia per aggiungere una nuova unità è sufficiente eseguire i seguenti passi:

1. Individuare tra le sotto classi di **om:Unit** la categoria appropriata
2. Se esiste una categoria adatta:
 - a. Creare un individual che ha come tipo la categoria selezionata
 - b. Aggiungere annotazioni come: comment, label, e om:symbol
 - c. Opzionalmente completare anche le object properties secondo le regole definite da OM
 - d. Se l'unità di misura ammette prefissi aggiungere una sotto classe a om:PrefixedUnit dove si descrivono i prefissi ammessi (?)
3. Creare una nuova categoria ed eseguire i passi 2.a, 2.b,2.c

⁹ <https://blazegraph.com/>

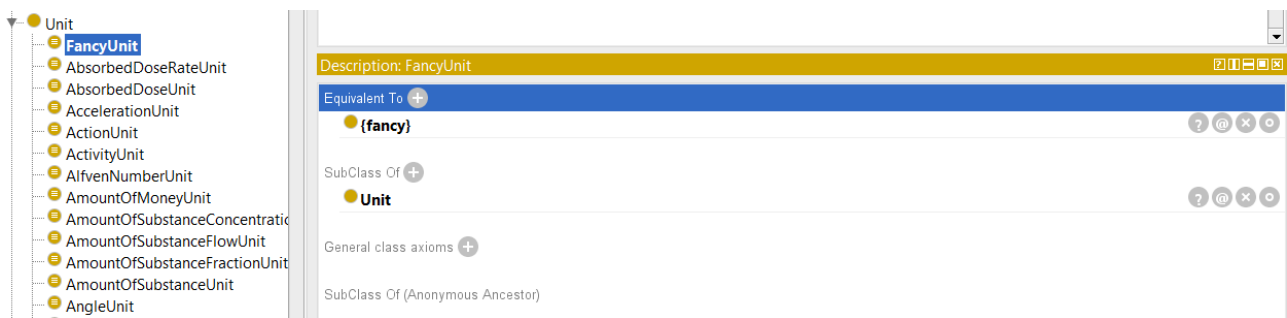


Figura 5 - Creazione di una nuova categoria

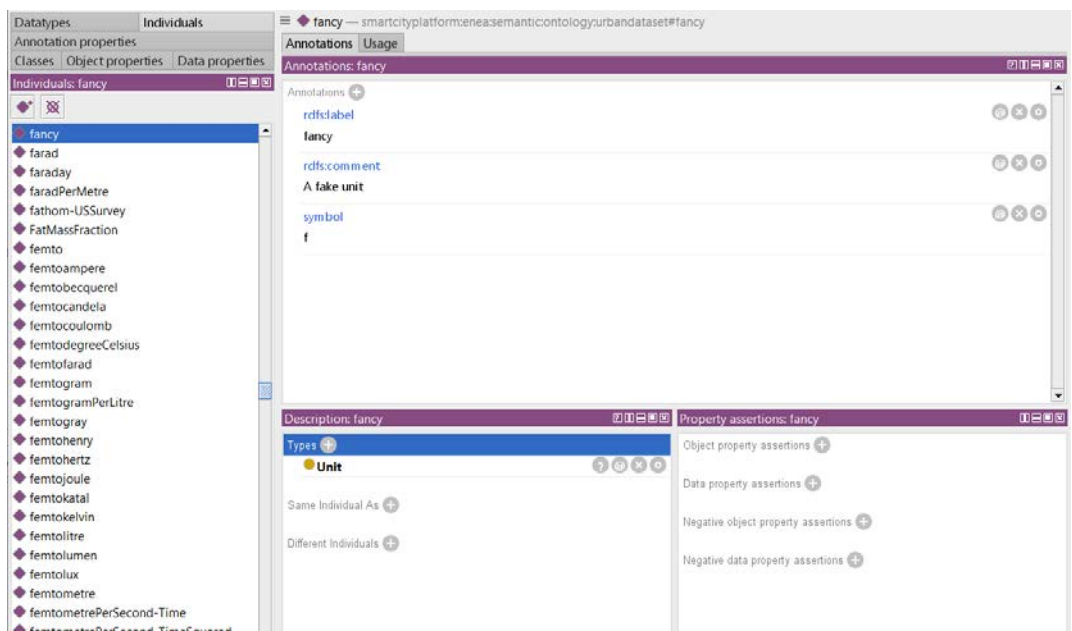


Figura 6 - Creare una nuova unità di misura

Per gli scopi dell’ontologia SPCS solo i passi 2.a e 2.b sono fondamentali per ottenere il corretto funzionamento delle funzionalità della Smart City PI mentre i restanti servono a mantenere una consistenza con le altre definizioni di unità di misura. Tale consistenza dipende molto da quale unità si sta definendo e da come essa si colloca all’interno delle sottocategorie definite da OM. Si suggerisce un’attenta valutazione delle altre istanze prima di aggiungere asserzioni al di fuori di quelle specificate in questo report.

Sempre riguardo all’unità di misura è stata rilevata la necessità di poter fornire una lista di unità di misura possibiliper una data proprietà. Per ottenere questa funzionalità è stata aggiunta la Data Property *hasAlternativeUnit* che ha lo scopo di dichiarare un’unità alternativa rispetto a quella fornita con *hasUnit*. Si è quindi deciso che tutte le unità di misura associate con *hasUnit* vanno indicate come default mentre quelle associate con *hasAlternativeUnit* sono da intendersi come possibili alternative.

```
<!-- smartcityplatform:enea:semantic:ontology:urbandataset#AirTemperature -->
<owl:NamedIndividual rdf:about="smartcityplatform:enea:semantic:ontology:urbandataset#AirTemperature">
  <rdf:type rdf:resource="smartcityplatform:enea:semantic:ontology:urbandataset#WeatherProperty"/>
  <hasDataType rdf:resource="smartcityplatform:enea:semantic:ontology:urbandataset#double"/>
  <om-2:hasUnit rdf:resource="http://www.ontology-of-units-of-measure.org/resource/om-2/degreeCelsius"/>
  <rdfs:comment>Temperatura dell&apos;aria</rdfs:comment>
  <hasAlternativeUnit rdf:resource="http://www.ontology-of-units-of-measure.org/resource/om-2/degreeFahrenheit"/>
</owl:NamedIndividual>
```

Codice 1 - Descrizione di una proprietà con unità di misura alternative

2.2.3 Aggregazione temporale e spaziale

Un'ulteriore modifica è stata quella relativa alla categorizzazione Spaziale e Temporale dei tipi di *UrbanDataset* definiti nell'ontologia.

Nella versione precedente, l'ontologia supportava la definizione del tipo di Aggregazione (temporale o spaziale) a livello di *UrbanDataset*: questo significa che per ciascun *UrbanDataset* veniva definito anche se era pensato per inviare dati, per es, a livello di un sensore, di una facility, di una città o magari di una regione e se i dati erano istantanei, medi, totali, ecc.

Dopo i feedback raccolti dagli utenti si è notato che questo portava a diversi limiti:

1. spesso un *UrbanDataset* richiede tipi di aggregazione diversi per singola proprietà (per esempio inviare un dato medio per una grandezza e uno istantaneo per un altro). L'approccio adottato rendeva ostico descrivere queste situazioni di questo tipo
2. la *SpaceAggregation* (indicante un aggregazioni di tipo spaziale) è stata trovata ridondante in quanto è possibile esprimerla con altre proprietà di contesto
3. la *TimeAggregation* (dato istantaneo, medio, ecc.) risultava non chiara, in quanto il campo "period" copriva lo stesso tipo di informazione
4. Si rischiava inoltre di dover replicare *UrbanDataset* con le stesse proprietà perché cambiava solo la copertura spaziale o temporale.

In questo percorso di sviluppo si è quindi deciso di eliminare questi concetti e di introdurre un nuovo sotto tipo della classe *UrbanDasetProperty* chiamata *MeasurementProperty*.

MeasurementProperty indica una proprietà per cui è possibile definire un *MeasurementType* attraverso la relazione *hasMeasurementType*. Correntemente sono stati definiti quattro *MeasurementType* con l'intenzione di estenderli in futuro qualora ci fosse la necessità:

- *Average*: indica che il valore della proprietà è una media
- *Istantaneous*: indica un valore misurato istantaneamente
- *Static*: un valore fisso
- *Total*: Il valore è il risultato di una sommatoria di valori misurati durante un periodo

Questo nuovo design permette quindi di associare alle proprietà per le quali ha senso (ad esempio *ElectricConsumption*) l'informazione su come è stato misurato il valore. Il principio ricorda quello di SAREF ma in questo caso non è stata definita una classe di individui bensì una proprietà OWL con cui poter riportare una misurazione. In SAREF quindi prima si deve relazionare un device ad una misura ed una proprietà per poi leggere il valore di tale misura, mentre in un *UrbanDataset* la relazione tra valore e individuo è diretta e data dalla relazione di tipo *hasUrbanDasetProperty*.

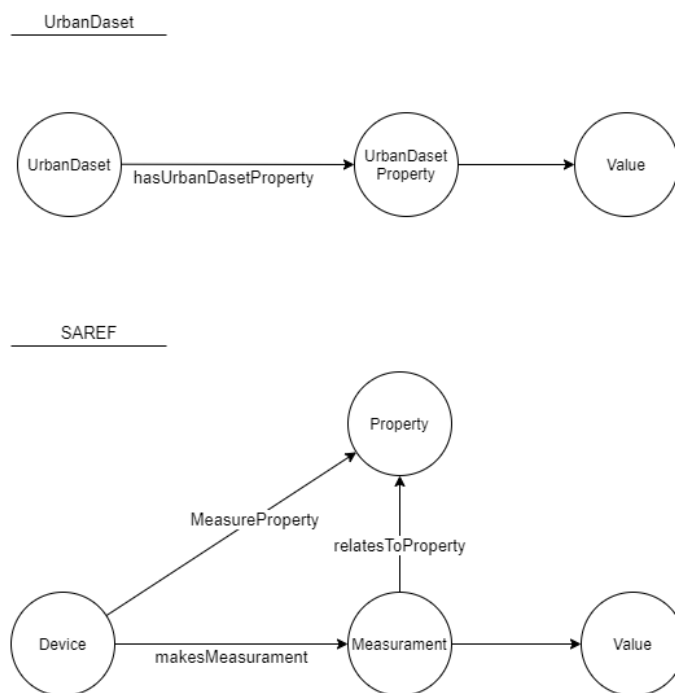


Figura 7 - In alto la struttura dell'ontologia in esame a confronto con SAREF in basso

2.2.4 Versionamento

Le modifiche apportate hanno portato alla creazione di una divergenza tra la soluzione precedente e quella corrente. Ci si è quindi interrogati come gestire questa problematica alla luce anche della prossima attività LA 23 dove le modifiche dell'ontologia saranno semplificate grazie ad un interfaccia di editing intuitiva. Il versionamento è un problema noto nella produzione software e tra la soluzioni più adottate c'è quello della notazione semantica¹⁰. Il versionamento semantico prevede di utilizzare una notazione basata su tre numeri separati da un punto: MAJOR.MINOR.PATCH . Nell'iterazione di sviluppo precedente la notazione utilizzata era basata su due numeri separati da un punto. Si è quindi deciso di mantenere tale dicitura ma allo stesso tempo di seguire le guide lines definite nella notazione semantica. Quindi ogni qualvolta la struttura del modello ontologico cambia in maniera non retro-compabile il primo numero viene incrementato di uno. Se invece vengono aggiunte proprietà o aggiornate descrizione di un Urban Dataset questo comporta un incremento del secondo numero (Minor). Modifiche di tipo *bug fixes* non vengono riportate nel numero di versione ma a seconda del loro impatto si decide se aumentare la Minor o continuare ad usare la stessa versione.

Inoltre si è deciso nel continuare a riportare il numero di versione anche a livello di singola istanza di Urban Dataset ma questo non va inteso come una versione di quel specifico modello bensì riporta la versione corrente dell'ontologia da cui è stato estratto. Stiamo ancora valutando se introdurre un valore informativo il quali riporti il numero di versione in cui un Urban Dataset sia stato introdotto.

¹⁰ <https://semver.org/lang/it/>

2.3 Modifiche software

Le modifiche software apportante possono essere categorizzate in tre tipi: allineamento con il nuovo modello ontologico, nuove funzionalità e mantenimento. Tali modifiche hanno richiesto particolare cura poiché è stato necessario uno studio approfondito della precedente soluzione software, capirne le scelte architetture, le tecniche di query e di configurazione, per poi agire in maniera chirurgica per ottenere l'effetto desiderato. Inoltre è stato necessario anche riprodurre fedelmente l'ambiente di produzione in locale in quanto protetto da politiche di accesso ristrette. Questo ha portato delle difficoltà nella riproduzione di bug e condizioni particolari rinvenute nella versione pubblicata rispetto a quella in sviluppo. Correntemente il software è pubblicato presso: <https://smartcityplatform.enea.it/SCPSWebLibrary/ontologyinfo>.

2.3.1 Allineamento

Il nuovo modello ontologico ha portato cambiamenti strutturali al grafo della conoscenza i quali hanno richiesto un aggiornamento delle funzionalità di caricamento dati nell'applicazione web. Tra le prime modifiche apportate c'è quella nell'interrogazione della base di conoscenza per poter gestire le proprietà opzionali. Riprendendo la Figura 3 nel nuovo modello per ottenere la lista della Proprietà di un tipo di Urban Dataset è necessario tenere conto anche di quelle Proprietà collegate con "hasOptionalUrbanDatasetProperty". A tal fine è stata riutilizzata ed estesa l'interfaccia di caricamento dati precedente che permetteva di ottenere la lista di coppie proprietà e oggetto RDF collegati ad un dato Urban Dataset. Per sapere quali coppie di questa lista fossero delle proprietà opzionali essa è stata filtrata con selezionando le coppie collegate con "hasOptionalUrbanDatasetProperty". Dopo di che si è proceduto con l'aggiornamento dell'interfaccia HTML generata attraverso JSP. Il design finale (Figura 8) prevede l'utilizzo di checkbox per mostrare se una data proprietà è richiesta o no (i.e., è opzionale).

Specifiche

Nome:	Home Aggregated Electric Consumption
URN:	smartcityplatform:enea:semantic:ontology:urbandataset#HomeAggregatedElectricConsumption
Versione:	2.0
Descrizione:	Fornisce i dati relativi al consumo di energia elettrica di uno o più gruppi di abitazioni. Ogni riga dell'UD riporta il consumo totale di un determinato gruppo, identificato dalle coordinate indicate nella riga

Lista delle proprietà :

Nome	Descrizione	Obbligatoria
ElectricConsumption	Energia elettrica totale consumata dal gruppo di abitazioni identificato dalle coordinate indicate nella riga	<input checked="" type="checkbox"/>
FloorArea	Superficie totale del gruppo di abitazioni a cui fa riferimento il consumo energetico	<input checked="" type="checkbox"/>
OccupantCount	Numero di occupanti totale del gruppo di abitazioni a cui fa riferimento il consumo energetico	<input checked="" type="checkbox"/>
coordinates	Riferimento geografico	<input checked="" type="checkbox"/>
period	Periodo durante il quale sono stati rilevati i dati riportati nella riga	<input checked="" type="checkbox"/>
BuildingID	Identificatore dell'edificio	<input type="checkbox"/>

Figura 8 - Nuova visualizzazione della lista di proprietà di un Urban Dataset

Inoltre, come descritto nel capitolo 2.3.2, la pagina di descrizione di Proprietà ora contiene una sezione per mostrare i tipi di Urban Dataset in cui è utilizzata. Per cui il codice è stato aggiornato per tenere conto anche di tipi nei quali tali proprietà era usata solo in via opzionale. Infatti, se in precedenza la pagina caricava solo Urban Daset Property collegate attraverso "hasUrbanDatasetProperty" ora vengono cercate e unite alla lista anche quelle collegate con "hasOptionalUrbanDatasetProperty".

Allo stesso modo per quanto riguarda le unità di misura è stato aggiunto un filtro per ricercare anche unità collegate con *hasAlternativeUnit* per poi mostrale come lista separata da virgole nella tabella di descrizione. Inoltre per sottolineare l'unità di misura di default si è optato per l'utilizzo del grassetto come mostrato in Figura 9.

Specifiche

Nome:	AirTemperature
URI:	smartcityplatform:enea:semantic:ontology:urbandataset#AirTemperature
Descrizione:	Temperatura dell'aria
Data Type:	double
Unità di misura:	degreeCelsius (default) , degreeFahrenheit
Code list:	Nessun code list disponibile

Figura 9 Una esempio di proprietà con più di un unità di misura

In aggiunta, a causa dell'introduzione di *hasMeasurementType* è stato necessario introdurre una nuova query SPARQL dedicata al calcolo delle proprietà derivate. Una proprietà derivata è una proprietà RDF definita in una super classe. Questo ha reso possibile definire una sotto classe di *UrbanDataSetProperty* che indicasse che le sue sottoclassi potessero utilizzare *hasMeasurementType* come associazione: *MeasurementProperty*. La query risultante quindi segue il grafo delle classi collezionando le possibili proprietà in ogni tipo trovato (Codice 2). La versione corrente della query è il risultato finale di varie sperimentazioni che hanno vagliato il supporto di particolari funzionalità SPARQL in due diverse implementazioni di database RDF (Blazegraph e Virtuoso). In particolare, si è verificato che l'utilizzo di costrutti come UNION e Property Patterns restituiscono risultati diversi a seconda del database scelto. Al fine quindi di mantenere l'applicazione agnostica rispetto al tipo di DB utilizzato, si è quindi sviluppata una query in grado di funzionare in entrambi (Blazegraph e Virtuoso), senza sacrificarne però la specificità (i.e., l'insieme delle risposte è lo stesso di quello della versione precedente).

Per quanto riguarda la visualizzazione delle proprietà derivate si è dapprima studiato un sistema per rappresentarle sottoforma tabellare allo stesso modo delle altre proprietà. Il supporto di questa funzionalità avrebbe richiesto un completo ripensamento del layout della pagina di descrizione proprietà. Infatti se la lista di attributi fosse variabile bisognerebbe prevedere la possibilità di introdurre una scrollbar oppure nascondere quei attributi meno importanti a favore di quelli più critici. Inoltre poiché la lista dei valori ammessi per attributo poteva essere qualsiasi, si avrebbe dovuto anche sviluppare una metodologia per definire sotto-tipi di dato specifici. Ad esempio, assumiamo l'introduzione di un nuovo attributo chiamato "isMeasurableBy" il quale definisce che una data Proprietà è misurabile da una particolare *entità*. Per mostrare correttamente questo modello sarebbe necessario dedicare una pagina di specifica e documentazione dedicata a tale *entità*. Allo stesso tempo però le specifiche dell'applicazione e del modello ontologico non comprendono la possibilità di definire entità terze ad UrbanDataset. Inoltre nello specifico il bisogno primario era semplicemente esprimere il fatto che una data classe di *UrbanDataSetProperties* potesse avere l'attributo *hasMeasurementType* (nota bene non un qualunque attributo). Per tali motivazioni si è deciso di semplificare la visualizzazione di proprietà

derivate e di mostrare solo la presenza o no di *hasMeasurementType* e non mostrare altre proprietà derivate. Il risultato è quindi mostrato in Figura 10. Di contro però il codice di back-end è estensibile e permetterebbe in futuro di definire altri attributi.

Infine, un'altra conseguenza delle modifiche apportate è stata quella di riadattare il meccanismo della generazione template di messaggio, schematron e documentazione. Nell'applicazione in esame i template d'esempio per un tipo di UrbanDataset vengono generati a partire dallo schema di validazione. Questo rende possibile la generazione di file sempre validi e mantenere allineata la logica validazione sintattica con quella di generazione. Dopo lo studio di tali funzionalità si è notato come la logica di generazione potesse rimanere disallineata rispetto allo schema di validazione, invalidando fortemente i vantaggi sopra elencati. Nella pratica l'applicazione web della smart city platform si avvale di due tool esterni per generare automatica delle classi java che ricalcano il modello descritto nel file di validazione: *jsonschema2pojo*¹¹ e *jaxb*¹². In particolare il problema riscontrato non era tanto nel funzionamento dei tool o nella loro configurazione ma bensì ne fatto che essi dovevano essere manualmente avviati dal developer per ottenere la sincronizzazione tra i file di schema e il codice. Durante lo sviluppo si è quindi riscontrato in varie occasioni la mancanza del corretto allineamento, per cui si è avviato al problema automatizzando il processo. Quindi, si è proceduto con la configurazione di due gradle tasks (vedi il lavoro di mantenimento riguardo al cambio del sistema di compilazione) che avviassero i tool al momento della compilazione. In definitiva, nella versione corrente i file nei package *jaxb.model* e *jsonschema.model* vengono generati automaticamente ogni qualvolta si crea il pacchetto dell'applicativo senza nessuna operazione manuale da parte dello sviluppatore.

Con l'implementazione di tale modifica ci si è adoperati nell'aggiornamento dei file di schema per poter esprimere e validare anche i nuovi concetti introdotti. In Codice 2 si può trovare un estratto del nuovo schema il quale comprende la nuova proprietà *measurementType* derivata per l'appunto dall'introduzione della proprietà derivata *hasMeasurementType*. Ovviamente le modifiche sono state apportate di conseguenza anche allo schema di validazione xml che, similmente a quello json, riporta il nuovo numero di versione nel nome del file: *scps-urbandataset-schema-2.0.xsd*. Infine, è stato necessario aggiornare la logica di generazione per poter configurare il motore di processamento con i nuovi dati estratti dall'ontologia. Infatti, è bene ricordare, che i tool automatici non si occupano di provvedere anche alla logica di configurazione ma bensì creare un modello 1 a 1 dei tipi riportati nel file di schema. Per esempio, se nello schema X viene definito un oggetto che ha come proprietà NOME, COGNOME ed ETÀ, *jsonschema2pojo* crea solamente una classe java con nome il nome dell'oggetto e metodi i getter e setters per le proprietà NOME, COGNOME ed ETÀ. Sta quindi all'applicazione prendere il modello e riempire tali proprietà con valori coerenti. Per cui la precedente logica è stata aggiornata con chiamate per estrarre:

- La lista completa delle proprietà (aggiunta alla lista anche quelle opzionali)
- Aggiungere *measurementType* se definito
- Indicare l'unità di misura di default

¹¹ <https://www.jsonschema2pojo.org/>

¹² <https://docs.oracle.com/javase/8/docs/technotes/guides/xml/jaxb/index.htm>

```

"type": "object",
"additionalProperties": false,
"properties": {
  "propertyName": {"type": "string"},
  "propertyDescription": {"type": "string"},
  "dataType": {"type": "string"},
  "codeList": {"type": "string"},
  "unitOfMeasure": {"type": "string"},
  "measurementType": {"type": "string"},
  "subProperties": {
    "type": "object",
    "properties": {
      "propertyName": {
        "items": {"type": "string"},
        "type": "array",
        "minItems": 2
      }
    },
    "required": ["propertyName"],
    "additionalProperties": false
  }
}

```

Codice 2 - Estratto del nuovo Json schema per la validazione di UrbanDataset

Similmente la generazione degli schematron (file per la verifica della consistenza semantica dei messaggi di Urban Dataset) ha necessitato un aggiornamento consistente. Il file di template da cui un particolare file di validazione viene creato è stato modificato in modo da tenere conto la lunghezza variabile delle proprietà definite. Infatti se prima il numero di UrbanDataset proprietà era fissato ora l'utente può decidere di non riportare uno o più proprietà opzionali. Inoltre anche la verifica dell'unità di misura ora tiene conto della possibilità dell'utilizzo di valori alternativi oltre a quello di default. Per finire, il codice per la configurazione del template è stato aggiornato in modo da estrarre le informazioni necessarie dall'ontologia. Per esempio, ora il SchematronBuilder.java ricava tutti i valori ammessi per *hasMeasurementType* e la lista delle unità di misura alternative dal modello ontologico e crea in accordo il file schematron. Infine, nella documentazione sono stati aggiornati i link riportando la versione corretta dell'ontologia ed è stato adottato un approccio basato su template simile a quello utilizzato per la creazione di schematron.

Da questa esperienza d'allineamento si può definire una lista dei punti d'intervento nel caso di modifiche future al modello ontologico in esame:

- Modulo *scpslib* per l'accesso al modello ontologico. Va modificato se si prevedono modifica sostanziali, come l'introduzione di *hasMeasurementType*
- Modulo *SCPSWebApplication* per l'interfaccia utente:
- Modificare i controller dedicati alla logica di presentazione (esempio *UrbanDatasetController*).

Una volta caricati i dati grazie alle utilità fornite da *scpslib* nel *ViewModel* modificare la pagina JSP dedicata al fine di mostrare i dati.

Modulo *templateGeneration* per la generazione dei template. In particolare, le modifiche si sono focalizzate nell'aggiornamento dello schema e della classe *MessageBuilder* e classi associate.

Specifiche

Nome:	ElectricConsumption
URI:	smartcityplatform:enea:semantic:ontology:urbandataset#ElectricConsumption
Descrizione:	Consumo energia elettrica
Data Type:	double
Unità di misura:	kilowattHour (default)
Code list:	Nessun code list disponibile
hasMeasurementType	[average, instantaneous, static, total]

Figura 10 - Una proprietà con *hasMeasurementType* come proprietà derivata

2.3.2 Nuove funzionalità

Oltre al lavoro di miglioramento del modello ontologico, sono state introdotte nuove funzionalità in vista di LA 23.

In prima battuta si è migliorata la visualizzazione della pagina riportante informazioni sugli Urban Dataset e sulla pagina dedicata alle Urban DataSet property. In particolare, ora la pagina al link `./urbandataset` mostra nella tabella delle proprietà una descrizione specifica e contestualizzata al tipo di Urban Dataset in esame. Per esempio `smartcityplatform:enea:semantic:ontology:urbandataset#ApplianceElectricConsumption` riporta tra le sue proprietà `smartcityplatform:enea:semantic:ontology:urbandataset#ElectricConsumption` e come descrizione la seguente: "Energia elettrica consumata dall'appliance nel periodo indicato". Se si va a confrontare con la vecchia versione invece leggeremo la descrizione standard: "Consumo energia elettrica". Come si può notare dell'esempio la descrizione dedicata aiuta l'utilizzatore a capire il senso specifico di una data proprietà e lo rende consapevole di come utilizzarla al meglio nel contesto d'uso. Se si pensa all'attività LA 23 avere questo tipo d'informazione è fondamentale per modificare in maniera consapevole l'ontologia ed inoltre per fornire la giusta documentazione quando si introdurranno nuove Urban Dataset Property. Nella pagina dedicata alle proprietà, invece, ora viene riportata la lista dei tipi di Urban Dataset in questa è utilizzata con la relativa descrizione specifica (Figura 11). Questa modifica migliora la navigabilità del sito inserendo dei link circolari tra Urban Dataset e le loro proprietà.

Usato negli UrbanDataset:

Nome	Descrizione
<code>BuildingElectricConsumption</code>	Energia elettrica consumata dall'edificio monitorato nel periodo indicato
<code>HomeAggregatedElectricConsumption</code>	Energia elettrica totale consumata dal gruppo di abitazioni identificato dalle coordinate indicate nella riga
<code>TreatmentPlantWasteWaterPerformance</code>	Energia elettrica consumata mediamente per metro cubo di acqua depurato nel periodo indicato
<code>ApplianceElectricConsumption</code>	Energia elettrica consumata dall'appliance nel periodo indicato

Figura 11 - Lista d'uso della proprietà *ElectricConsumption*

Una seconda miglioria ha riguardato la generazione dei messaggi di template con proprietà opzionali (vedi capitolo 2.3.1). Infatti, poiché ora è possibile definire quali proprietà siano opzionali in un tipo di Urban Dataset, è stata introdotta la possibilità di scegliere quali tra queste debbano essere inserite nel template generato. La selezione avviene tramite le checkbox indicate nella colonna “Obbligatoria” e una volta cliccato il link dedicato al servizio di generazione, viene inviata al server la lista delle proprietà selezionate. Il servizio, quindi genera il template secondo quanto specificato dall’utente. È bene ricordare che questa modifica è in linea a quanto previsto in LA 23, infatti va verso una personalizzazione della piattaforma configurabile secondo le esigenze specifiche di un singolo utente.

2.3.3 Manutenimento

Le modifiche di mantenimento sono state necessarie al fine di migliorare l’estensibilità della piattaforma, lo sviluppo e l’usabilità. In primis è stato uniformato il sistema di building del progetto, utilizzando in tutti gli applicativi e librerie il build tool Gradle. Inoltre, tutto il progetto è ora supervisionato da un sistema di controllo versione (Git) al fine di tenere traccia dei cambiamenti apportati e di gestire una modifica collaborativa.

Limitatamente all’applicativo web sono state apportate migliorie nel caricamento dei dati e nel processo di presentazione dati. In aggiunta, il look&feel è stato allineato con quello della pagina principale del progetto disponibile a questo indirizzo: <https://smartcityplatform.enea.it/>.

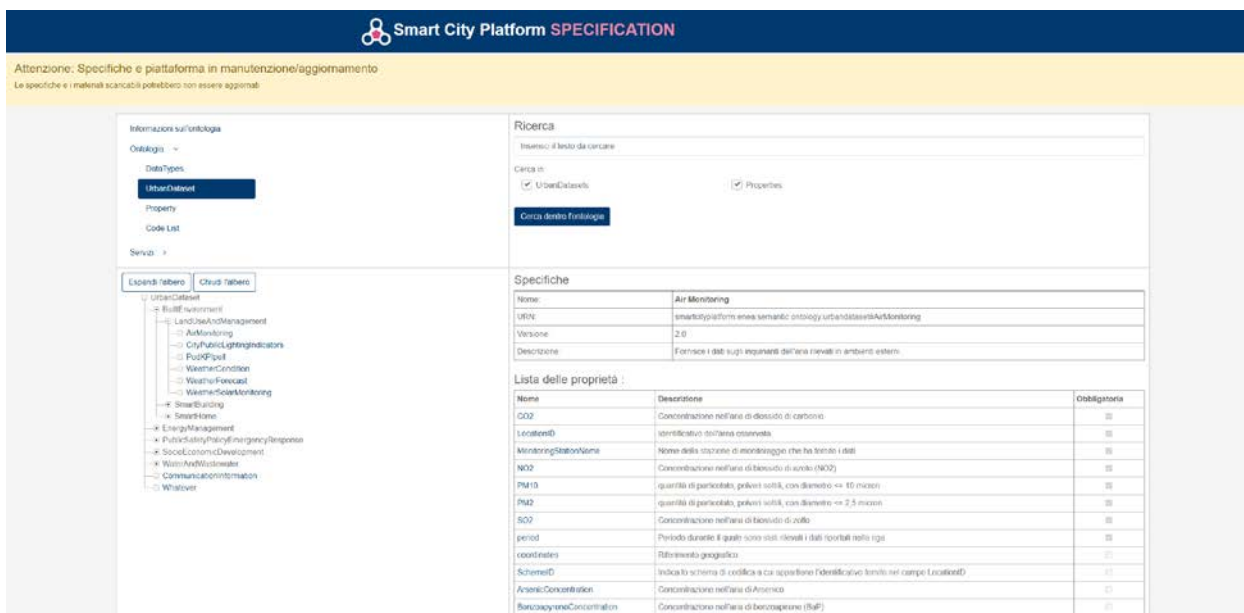
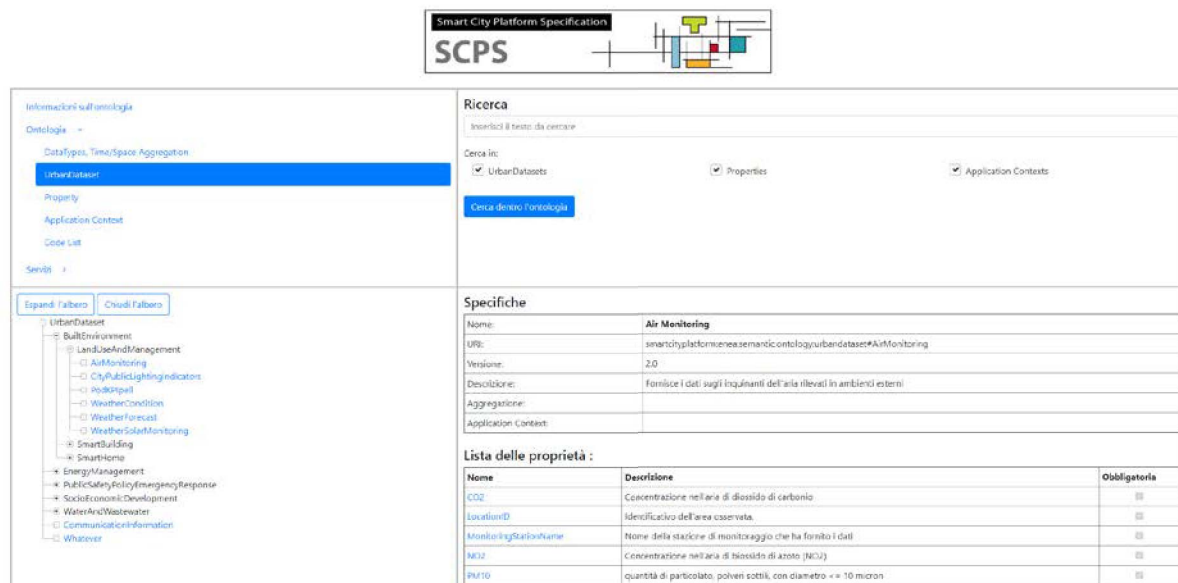


Figura 12 -Nuovo tema



Nome	Descrizione	Obbligatoria
CO2	Concentrazione nell'aria di diossido di carbonio	☐
LocationID	Identificativo dell'area censuata	☐
MonitoringStationName	Nome della stazione di monitoraggio che ha fornito i dati	☐
PM10	Concentrazione nell'aria di biossido di azoto (NO2)	☐
PM10	quantità di particolato: polveri sottili, con diametro <= 10 micron	☐

Figura 13 - Vecchio stile

Relativamente all'implementazione del nuovo tema si è prestata particolare cura nel definire un sistema compatibile con la libreria Bootstrap. Infatti, non si è semplicemente optato per utilizzare regole CSS ad-hoc ma si è creato un sistema di compilazione SCSS che importasse e sovrascrivesse le regole definite in Bootstrap¹³. Quindi nell'applicazione allo stato attuale vi è un file .scss dedicato alla generazione del tema, il quale viene automaticamente compilato in CSS assieme ai sorgenti Java ed inserito tra le risorse statiche disponibili tramite richieste http.

Inoltre, sempre per quanto riguarda l'applicativo web sono stati aggiornati e ripristinati i servizi di Validazione e Trasformazione i quali non erano funzionati nella versione online. Durante il processo di risoluzione si è anche migliorata l'organizzazione logica del codice relativamente all'aspetto di validazione. Infatti sia il servizio di Validazione che di Trasformazione necessitano, per l'appunto, di validare un file contenente un Urban Dataset. Se nella versione precedente questa funzionalità era stata implementata in due Spring controllers separati, ora essa è offerta da uno Spring Service dedicato. Questo permette di evitare l'inutile duplicazione del codice e, quindi, migliorare la manutenibilità dell'applicazione. Inoltre, invece di confinare la logica di validazione all'interno dell'applicativo web essa è stata estratta in una libreria esterna così da facilitare in futuro l'implementazione di altre applicazioni che utilizzino la tipologia di messaggi della Smart City Platform. In aggiunta, come è intuibile, questa logica condivisa tra i due servizi di validazione e trasformazione si riflette anche nell'implementazione del lato visuale. Per questo motivo, intera interfaccia è stata ripensata e migliorata. In primis, si è definito un componente JSP per la visualizzazione di un risultato d'output generico. Questo componente è diviso in due sezioni: la prima mostra il risultato della validazione con schema, mentre la seconda quella con schematron. Come mostrato in Figura 14 si nota che la validazione con schema è andata a buon fine mentre quella con schematron è risultata in un warning. Una volta implementato questa logica di visualizzazione il componente è stato riutilizzato nella JSP dedicata alla validazione nel pannello di destra (Figura 14). Inoltre, prima di mostrare l'output la pagina JSP ora mostra istruzioni valide a guidare l'utente nell'uso del servizio. Stessa procedura è stata riservata al file *transform.jsp* dedicato al servizio di trasformazione. In questo caso la particolarità è stata che nel servizio citato il file viene validato

¹³ <https://getbootstrap.com/>

due volte: prima e dopo la trasformazione. Quindi nella pagina sono presenti due istanze del componente descritto prima. Ovviamente il risultato della validazione dopo la trasformazione viene mostrato solo dopo che la prima è stata eseguita con successo.



Figura 14 - Nuovo pannello uniformato di presentazione risultati di validazione

Oltre alle modifiche per aggiornare e migliorare le funzionalità esistenti sono stati risolti vari errori. Tra essi citiamo quelli che hanno richiesto uno sforzo significativo. Il primo errore si presentava solo nella versione online dell'applicativo web. Nella pratica le pagine non venivano generate nella loro interezza ed erano mancati delle parti di descrizione per Urban Dataset e Urban Dataset Property. La differenza principale tra l'ambiente di produzione e quello di sviluppo è l'utilizzo di un database RDF diverso (Virtuoso vs Blazegraph). Dopo un'indagine accurata si è riscontrato un problema nella comunicazione tra applicativo e database. Nella pratica, la libreria scpslib apriva varie connessioni in parallelo senza poi però liberare le risorse. Questo creava un deadlock e ulteriori richieste non venivano soddisfatte, da qui il risultato di pagine non complete. Una volta chiuse le connessioni si sono ripristinate la funzionalità base. Si sospetta che tale errore non era stato rinvenuto in precedenza poiché nella nuova versione del modello ontologico il numero di query per pagina è aumentato raggiungendo il limite di connessioni massime gestibile dal thread pool predisposto dalla libreria. Un'altro problema riguardava la gestione dei file generati (documentazione e schematron). Infatti, essi venivano salvati su disco per ridurre i tempi di attesa di richieste successive e in rapida sequenza. L'applicativo web creava quindi una sorta di cache permanente dei file generati dai servizi di documentazione e schematron. In previsione dell'attività LA 23 però questo comportamento, seppure efficiente, limita la modificabilità "online" del modello poiché i file vengono generati solo la prima volta ed è impossibile rigenerarli mentre l'applicativo è in esecuzione. Di conseguenza si è prevista una revisione completa del sistema di caching utilizzando le funzionalità fornite dal framework spring boot¹⁴. Grazie a tale operazione è stato possibile gestire un tempo di validità della cache. In futuro, si potrebbe prevedere un'estensione del sistema anche alle query utilizzate per la costruzione della pagine e dei metodi per invalidare immediatamente la cache (e.g., per esempio non appena il modello è stato aggiornato).

Infine sono state apportate altre modifiche minori:

- Risoluzione di errori nella ricerca
- Migliorate le pagine di descrizione di Data Types
- Aggiornata la pagine di documentazione con un nuovo template

¹⁴ <https://docs.spring.io/spring-boot/docs/2.1.6.RELEASE/reference/html/boot-features-caching.html#boot-features-caching>

- Sia gli schematron che i file di documentazione ora sono disponibili ad un perma-link dedicato:
(es. <http://smartcityplatform.enea.it/SCPSWebLibrary/schematrons/AirMonitoring>)
- Migliorata la descrizione della pagina di validazione e trasformazione
- Aggiunto feedback visivo all'avvenuto caricamento del file da validare/trasformare

3 Conclusioni

In questo documento si è descritto il lavoro svolto per la definizione, la realizzazione ed il miglioramento di diverse applicazioni con l'obiettivo di estendere e di aggiornare l'ontologia costruita nei precedenti anni di progetto. Lo scopo di queste applicazioni era di migliorare la gestione dell'ontologia e delle specifiche dei documenti di scambio delle informazioni in formato XML e JSON all'interno di una piattaforma ICT per una Smart City. Lo scopo di tale piattaforma è la condivisione di informazioni tra i vari ambiti applicativi di una città. La raccolta e la condivisione di informazioni tra i diversi ambiti può essere utile a creare sinergie tali da favorire nuovi e più importanti risultati nell'ambito del risparmio energetico e dell'efficienza di una città.

Il lavoro dello scorso triennio per lo sviluppo concettuale dell'ontologia e dei relativi strumenti software è stato molto importante per avviare un nuovo approccio al problema dell'interoperabilità nella smart city, ma ha mostrato alcuni limiti nel passaggio nel mondo reale, con l'utilizzo da parte degli Stakeholder. Questo ha reso necessari diversi interventi importanti sia a livello di aggiornamenti che a livello di nuove idee e nuovi sviluppi.

Il lavoro svolto durante questo periodo di estensione, sfruttando quanto sviluppato precedentemente, è continuato rivisitando e migliorando alcune applicazioni sviluppate in precedenza, la gestione e la consultazione dell'ontologia degli UrbanDataset.

In particolare è stato volto a:

- allineare il modello ontologico con altre alternative correntemente usate nello stesso ambito applicativo e nella manutenzione della piattaforma
- rispondere alle esigenze emerse testando l'uso delle specifiche in contesti reali

Le modifiche apportate formano una base solida da cui costruire tool o piattaforme online atte allo sviluppo collaborativo della base di conoscenza. Queste attività verranno realizzate all'interno di LA 23 con lo scopo di adattare sempre più la piattaforma dei dati urbani all'esigenze particolari di ogni realtà cittadina.

Curriculum Vitae Michela Milano

Laureata in Ingegneria Elettronica presso l'Università degli Studi di Bologna riportando la votazione di 100/100 e lode, il 16 Marzo 1994. Ha ottenuto il titolo di **Dottore di Ricerca** in Ingegneria Elettronica e Informatica il 30 Giugno 1998.

Posizione Attuale

Dal 1 Aprile 2016 ricopre il ruolo di **Professore Ordinario nel settore concorsuale 9/H1 (ING-INF/05) – Sistemi di Elaborazione delle Informazioni** presso la Facoltà di Ingegneria di Bologna afferendo al Dipartimento di Informatica – Scienza e Ingegneria presso cui svolge attività didattica e di ricerca, partecipando attivamente a convenzioni e progetti di ricerca.

Attività di Ricerca

L'attività di ricerca di Michela Milano riguarda i sistemi di supporto alle decisioni basati sulla Programmazione a Vincoli e la sua integrazione con tecniche di Programmazione Intera: in particolare, sono stati investigati sia aspetti metodologici sia aspetti applicativi con riferimento a numerose applicazioni quali scheduling, allocazione, cutting e packing, routing, aste combinatorie e recentemente per problemi decisionale e di ottimizzazioni legati allo sviluppo sostenibile e al processo di policy making. In questo settore Michela Milano ha raggiunto visibilità internazionale e ha collaborazioni con diversi gruppi di ricerca, universitari e industriali. Infine le è stato assegnato il Google Focused Grant Program on Mathematical Optimization and Combinatorial Optimization in Europe nel 2012 e il Google Faculty Research Award nel 2016 su integrazione di reti neurali profonde in modelli combinatori.

Curriculum Vitae Federico Chesani

Laureato in Ingegneria Informatica presso l'Università degli Studi di Bologna riportando la votazione di 98/100, il 17 Luglio 2002.

Ha ottenuto il titolo di **Dottore di Ricerca** in Ingegneria Elettronica, Informatica e delle Telecomunicazioni il 12 Aprile 2007.

Posizione Attuale

Dal 1 Aprile 2012 svolge attività didattica e di ricerca presso la Scuola di Ingegneria e Architettura dell'Università di Bologna, e afferisce attualmente al Dipartimento di Informatica – Scienza e Ingegneria DISI, nel ruolo di Ricercatore Universitario a tempo indeterminato, partecipando attivamente sia a progetti di ricerca, che a progetti di trasferimento tecnologico.

Attività di Ricerca

Federico Chesani ha svolto la sua attività di ricerca prevalentemente nell'ambito dei sistemi esperti e di supporto alle decisioni basati su approcci a regole: in particolare, si è occupato sia di aspetti teorici legati ai sistemi a regole in logiche abduitive per gestire l'assenza di conoscenza e l'integrazione di conoscenza ontologica, sia ad aspetti maggiormente pratici legati all'applicazione di sistemi in presenza di conoscenza incerta e/o probabilistica. In particolare, nell'ambito dei numerosi progetti a cui ha contribuito, ha applicato sistemi a regole per il supporto alle decisioni in ambito sanitario (sia a livello italiano che europeo), "policy making", e manifatturiero/industriale. Nell'ambito di tale attività di ricerca, è co-autore di oltre 60 pubblicazioni, ed è stato invitato a tenere seminari e tutorial nell'ambito di conferenze internazionali.

Curriculum Vitae di Paola Mello

Paola Mello si è laureata in Ingegneria Elettronica presso l'Università degli Studi di Bologna nel dicembre 1982, riportando la votazione 100/100 e lode. Nel 1989 ha conseguito il titolo di Dottore di Ricerca in Ingegneria Elettronica e Informatica (Il Ciclo di Dottorato). Dal 1990 ha svolto la sua attività didattica e di ricerca presso il Dipartimento di Elettronica, Informatica e Sistemistica dell'Università di Bologna prima come ricercatrice e poi (dal 1992) quale professoressa associata.

Posizione Attuale

È attualmente in servizio come professore universitario di ruolo di I fascia (settore scientifico-disciplinare: ING-INF/05) presso il Dipartimento di Informatica - Scienza e Ingegneria dell'Università di Bologna, dove tiene, quale titolare, gli insegnamenti di Fondamenti di Intelligenza Artificiale e Fondamenti di Informatica per il Corso di Laurea in Ingegneria Informatica.

Attività Scientifica

L'attività scientifica di Paola Mello si è sviluppata principalmente nell'ambito dell'Intelligenza Artificiale. Ha partecipato e coordinato, quale responsabile scientifico, numerosi progetti di ricerca nazionali ed internazionali sui temi dell'Intelligenza Artificiale e della Logica Computazionale e ha collaborato e collabora con i principali centri di ricerca internazionali del settore. L'attività di ricerca ha riguardato tematiche distinte, ma correlate, fra cui citiamo Sistemi Basati sulla Conoscenza, Linguaggi Logici, Sistemi ad agenti. È socia della Associazione Italiana per il Calcolo Automatico (AICA), della Association for Logic Programming (ALP), socia fondatrice della Associazione Italiana per l'Intelligenza Artificiale (AI*IA), del Gruppo Utenti Logic Programming (GULP) e della Società Italiana di Informatica Biomedica (SIBIM). Dal 2010 al 2014 è stata Presidente dell'Associazione Italiana di Intelligenza Artificiale. Dal 2016 fa parte dell'Accademia delle Scienze. Nel 2017 è stata nominata fellow dell'Associazione Europea dell'Intelligenza Artificiale.

Curriculum Vitae di Marco Patella

Marco Patella si è laureato in Ingegneria Elettronica il 9/12/1993 presso l'Università degli Studi di Bologna con 100/100 e lode e nel 1999 ha conseguito, presso lo stesso ateneo, il titolo di dottore di ricerca in Ingegneria Elettronica ed Informatica.

Posizione Attuale

Dal 2019 ricopre il ruolo di professore ordinario per il settore scientifico-disciplinare ING-INF/05 – Sistemi di elaborazione delle informazioni – all'interno del Dipartimento di Informatica - Scienza e Ingegneria (DISI) dell'Università degli Studi di Bologna, presso la Scuola di Ingegneria.

Attività Scientifica

La sua attività scientifica si è sviluppata prevalentemente con riferimento a problematiche di interrogazione nell'ambito delle basi di dati multimediali. In tale contesto sono stati ottenuti risultati di rilievo nella progettazione ed analisi teorica dei metodi di accesso e nell'ideazione di algoritmi per l'elaborazione delle interrogazioni. Più recentemente, la sua ricerca si è anche concentrata sull'interrogazione nelle basi di dati di immagini, sul Data Mining e su interrogazioni semantiche in reti peer-to-peer. Marco Patella è uno degli ideatori dell'M-tree che, come riconosciuto da tutti i maggiori esperti del settore, rappresenta una pietra miliare nel campo dell'indicizzazione di spazi metrici. La significatività dei risultati ottenuti nel campo della ricerca è provata dall'intensa produzione di articoli pubblicati sulle più prestigiose riviste internazionali, quali IEEE Transactions on Pattern Analysis and Machine Intelligence e ACM Transactions on Database Systems, e presentati nei maggiori congressi internazionali del settore. Marco Patella ha infine svolto con efficacia un'intensa attività didattica nell'ambito dei Corsi di Laurea in Ingegneria Informatica, Meccanica e Gestionale dell'Università di Bologna, tenendo, con grande soddisfazione degli studenti, corsi di Fondamenti di Informatica, Sistemi Informativi Avanzati, Sistemi Informativi per le Decisioni LS, Tecnologie delle Basi di Dati M e Ingegneria del Software.