Distributed Systems and Internet Technologies Lab
Distributed Data Intelligence and Technologies Lab
Department of Information Engineering (DINFO)
University of Florence
http://www.disit.dinfo.unifi.it

# Km4City - The Knowledge Model 4 the City

# Smart City Ontology

Since 2018, the Km4City is maintained by Snap4City: https://www.snap4city.org also operated by DISIT lab and community.

Authors: Pierfrancesco Bellini, Paolo Nesi, Mirco Soderi

Referent coordinator: paolo.nesi@unifi.it

Knowledge base accessible as SPARQL entry point as shown from https://log.disit.org

OWL version accessible from: https://www.disit.org/6506 **[the download link is for version 1.6.7]**

https://www.disit.org/km4city/schema/ **[the download link is for version 1.6.7]**

https://www.disit.dinfo.unifi.it

Info from info@disit.org , snap4city@disit.org



Version 5.1, of this document

Referring to version 1.6.7 of the ontology

Date 26-08-2020



Questo documento in Italiano: https://www.disit.org/6461 **[the link is for version 5.1 of this document]**

This document in English: https://www.disit.org/5606 **[the link is for version 5.1 of this document]**

## Scopo

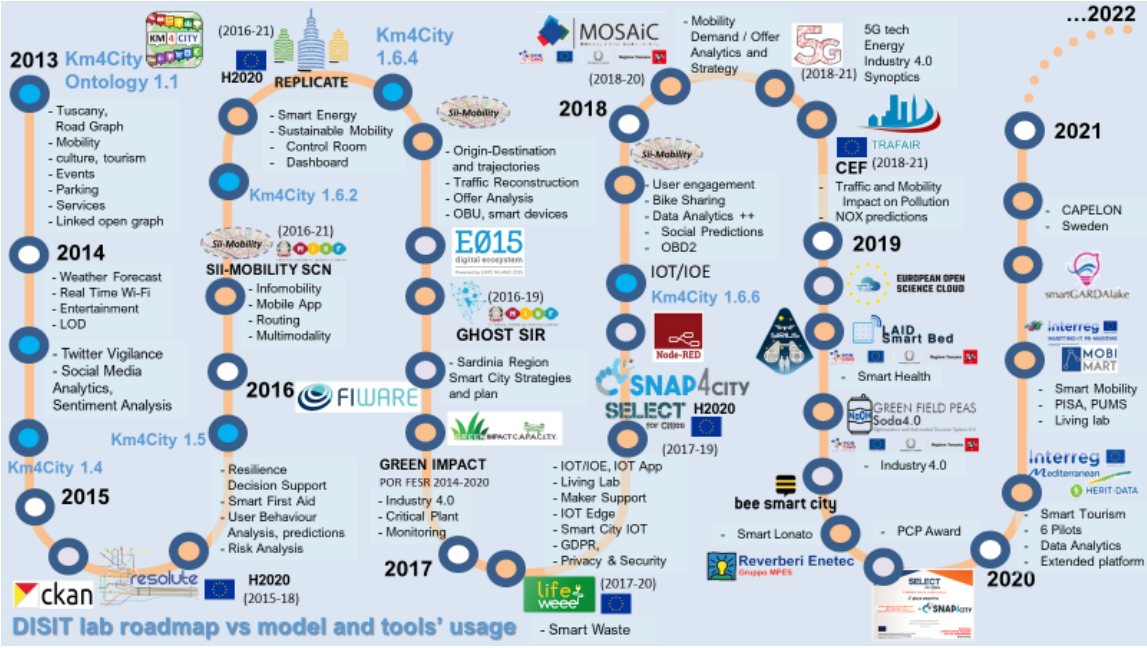
Lo sviluppo di un ontologia integrata ed unificata per Smart City che includa trasporti, infomobilità' e grandi set di altri Open Data.

- **Fornire un punto di accesso unificato** a dati e strumenti integrati e aggregati per:
  - Utenti qualificati: pubbliche amministrazioni → sviluppatori
  - Operatori: mobilità, energia, SME, negozi, … → sviluppatori
  - Utenti finali → cittadini, studenti, pendolari, turisti
- **Problemi**:
  - Support for semantic queries enabling the development of smart solutions:
    - Bike sharing
    - Smart parking
    - Resilience decision support system
    - What-if analysis
    - Routing
    - Pollutant prediction models
    - IOT integration
    - Anomaly detection
    - Etc.
  - Dati aggregati non sono disponibili:
    - Non interoperabili semanticamente, eterogenei rispetto a: formato, vocabolario, struttura, velocità, volume, proprietà/controllo, accesso/licenza, …
    - As OD, LD, LOD, dati privati, …
  - Carenza di servizi e strumenti per rendere l'adozione ***semplice***

Km4City è utilizzata nei progetti Sii-Mobility SCN e RESOLUTE H2020. Km4City è raccomandata da Ready4SmartCities FP7 CSA http://smartcity.linkeddata.es. Va anche detto che tutti gli strumenti sviluppati dal DISIT per le smart city non sono vincolati all'adozione di uno specifico modello dei dati.

**Km4City fornisce un insieme di modelli e strumenti per gli sviluppatori e gli amministratori delle smart city.** Questo lavoro è stato utilizzato in numerosi progetti relative alle smart city, tra cui:

- SMART CITY nel DISIT Lab: http://www.disit.org/6056
- Linked Open Graph: http://log.disit.org
- Service Map: http://servicemap.disit.org
- Slide sullo stato di Km4City, tendenze e strumenti: http://www.disit.org/6669
- Processi di importazione e relativa documentazione: http://www.disit.org/6058
-




A number of projects are using and/or contributing on Km4City technologies and solutions:

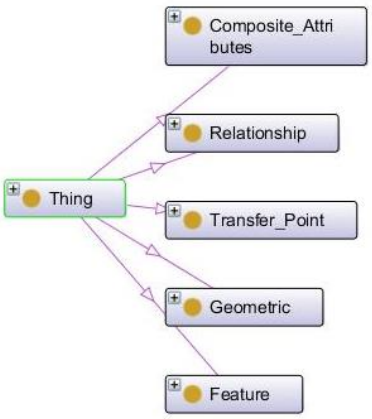
- Snap4city:
  - https://www.snap4city.org
- Sii-Mobility → mobility and transport, sustainability
  - http://www.sii-mobility.org/
- REPLICATE → ICT, smart City Control room, Energy, IOT
  - http://www.resolute-eu.org/
- RESOLUTE → Resilience, ICT, Big Data
  - http://replicate-project.eu/
- GHOST → Strategies, smart city
  - http://sites.unica.it/ghost/home/
- TRAFAIR → Environment & transport
  - https://trafair.eu/
- MOSAIC → mobility and transport
- WEEE Life → Smart waste, environment
  - https://www.lifeweee.eu/lifeWeee_it
- Smart Garda Lake → Castelnuovo del Garda
- 5G → Industry 4.0 vs SmartCity
  - https://www.corrierecomunicazioni.it/telco/5g/wind-tre-e-open-fiber-protagonisti-a-prato-del-debutto-italiano-del-5g/
- Green Impact → Industry 4.0, Chemical Plant
- SmartBed (laid) → smart health
- Green Field Peas (soda) → Industry 4.0, Chemical plant
- PISA MobiMart and Agreement → data aggregation, Living Lab
  - http://interreg-maritime.eu/web/mobimart
- Lonato del Garda → smart parking, environment
  - https://smartgardalake.it/progetto/

## Stato dell'Arte

Per interconnettere tra di loro i dati forniti dalla Regione Toscana, gli Open Data del comune di Firenze e gli altri dataset statici e Real Time che ci sono stati forniti, abbiamo iniziato a sviluppare uno Knowledge Model che permetta di raccogliere tutti i dati provenienti dalla città relativi a mobilità, statistiche, grafo strade etc.

L'unica ontologia presentata come un aiuto per trasformare le città in Smart City, è SCRIBE, realizzata da IBM, della quale però non sono disponibili gratuitamente online molte informazioni.

Tra le altre ontologie che potrebbero essere in qualche modo legate alle Smart City, citiamo la OTN, Ontology of Transportation Networks.




Questa ontologia al suo interno definisce l'intero sistema di trasporto in tutte le sue parti, dalla singola strada/ferrovia, fino al tipo di manovra che è possibile effettuare su un segmento stradale o i percorsi del trasporto pubblico. Come è possibile osservare dalla figura a fianco, la OTN raggruppa i concetti che esprime sotto cinque principali macro-classi: quella degli attributi compositi (dove troviamo classi come TimeTable, Accident, House_Number_Range, Validity_Period, Maximum_Height_Allowed), delle relazioni (all'interno della quale troviamo le Manoeuvre), dei punti di trasferimento (macro-classe che comprende classi come Road, Road_Element, Building e altre), delle geometrie (cioè le classi Edge, Node e Face) e infine delle caratteristiche (che raccoglie classi come Railways, Service, Road_and_Ferry_Feature, Public_Transport).

Sulla rete ci sono anche molte altre ontologie relative alle reti di sensori, come la versione più recente della Semantic Sensor Network Ontology, che fornisce elementi per la descrizione dei sensori e delle loro osservazioni, e FIPA Ontology che è più focalizzata sulla descrizione dei dispositivi e delle loro proprietà sia hardware che software. Ad oggi, l'ontologia SSN viene riutilizzata nell'Ontologia Km4City per la modellazione dei dispositivi Internet of Things, la loro organizzazione e le loro rilevazioni.

Analizzando poi i dati messi a disposizione dalle PA (principalmente dalla Regione Toscana) è stato riscontrato che, in relazione al Grafo Strade, i dati forniti potevano essere facilmente mappati in ampie parti della OTN, relative allo stradario; tenendo conto di queste similitudini riscontrate, abbiamo ritenuto che potesse essere utile, allo scopo di associare una semantica più precisa alle varie entità della nostra ontologia, fare riferimento esplicito alla OTN, in modo tale anche da rendere i concetti espressi più chiari e più facilmente linkabili ad altri, volendo seguire le linee guida per la realizzazione di Linked Open Data. A questo principio si ispira il modello della conoscenza di Km4City.

Un'altra interessante ontologia, rivolta però maggiormente verso l'e-commerce, è GoodRelations, cioè un vocabolario standardizzato che permette di descrivere dati relativi a prodotti, prezzi, negozi ed aziende in modo tale che possano essere inclusi all'interno di pagine Web esistenti e possano essere interpretati da altri computer, in modo tale da incrementare anche la visibilità di prodotti e servizi offerti nei motori di ricerca di ultima generazione, sistemi per raccomandazioni e applicazioni simili.

La possibile integrazione di tra l'ontologia Km4City e la GoodRelations è realizzabile a livello di classe: le due classi km4c:Entry e goodrelations:Locality possono essere connesse in modo da collegare il servizio al punto di accesso (attraverso la proprietà km4c:hasAccess).

Anche l'ontologia Schema.org, che integra già il progetto GoodRelation, è stata fondamentale per la definizione degli eventi: la classe creata eredita infatti la struttura della schema:Event ed è stata poi espansa in base alle informazioni in nostro possesso; la stessa ontologia è stata sfruttata per la definizione della schede di contatto delle aziende.

Un'altra ontologia che è stata utilizzata per la definizione delle forme geometriche, è la geoSPARQL, la cui integrazione proietta Km4City verso l'utilizzo del linguaggio di query geoSPARQL.

## L'ontologia

Km4City dovrà permettere l'interconnessione, la memorizzazione e la successiva interrogazione, di molti dati provenienti da differenti fonti, quali i vari portali della regione toscana (MIIC, muoversi in toscana, osservatorio dei trasporti) o gli stessi Open Data e Linked Data messi a disposizione dai singoli comuni (principalmente Firenze). È evidente quindi che l'ontologia che si andrà a realizzare non sarà di piccole dimensioni, e quindi potrebbe essere utile vederla come costituita da varie macro-classi, e per la precisione, attualmente, sono state individuate le seguenti macro-classi:

1. Amministrazione: la prima macro-classe che è possibile individuare, le cui classi principali sono PA, Municipality, Province, Region, Resolution.
2. Stradario: formata dalle classi Road, Node, RoadElement, AdministrativeRoad, Milestone, StreetNumber, RoadLink, Junction, Entry, EntryRule e Maneuver.
3. Punti di Interesse: comprende tutti i servizi e le attività che possono essere utili al cittadino e che quindi quest'ultimo può aver necessità di ricercare e di raggiungere. La classificazione dei singoli servizi e attività si baserà sulla classificazione precedentemente adottata dalla Regione Toscana. Sono inoltre inclusi in questa macro-classe le Digital Location e gli eventi programmati (dati Real Time) del Comune di Firenze
4. Trasporto Pubblico Locale: attualmente disponiamo dell'accesso ai dati relativi agli orari programmati delle principali aziende TPL, al grafo ferroviario, e ai dati relativi al tempo reale dell'ATAF. Tale macro-classe è quindi formata dalle classi TPLLine, Ride, Route, AVMRecord, RouteSection, BusStopForeast, Lot, BusStop, RouteLink, TPLJunction.
5. Sensori: l'Ontologia Km4City è idonea a rappresentare i sensori del traffico, i rilevamenti del traffico, i posti liberi nei parcheggi, le rilevazioni in tempo reale di eventi critici ed emergenze, le condizioni meteorologiche, le previsioni meteo e così via.
6. Tempo reale: macro-classe che punta all'inserimento di concetti legati al tempo (istanti di tempo e intervalli di tempo) all'interno dell'ontologia, in modo da poter associare una linea temporale agli avvenimenti registrati e poter riuscire a fare previsioni.
7. Metadati: macro-classe di triple (metadati) associate al context di ciascun dataset; tali triple raccolgono le informazioni relative a licenza del dataset, se il processo di ingestion è automatizzato completamente, il formato della risorsa, una breve descrizione della risorsa ed altre info sempre legate alla risorsa stessa e al suo processo di ingestion.
8. Internet of Things: una nuova sezione è stata inclusa nell'Ontologia Km4City dalla versione 1.6.5, che mira a modellare sensori generici, attuatori, i loro broker e i tipi di misure che sono in grado di eseguire.

Analizziamo adesso una ad una le differenti macro-classi individuate.

La prima macro-classe, Amministrazione, si compone come riportato in Figura 1.

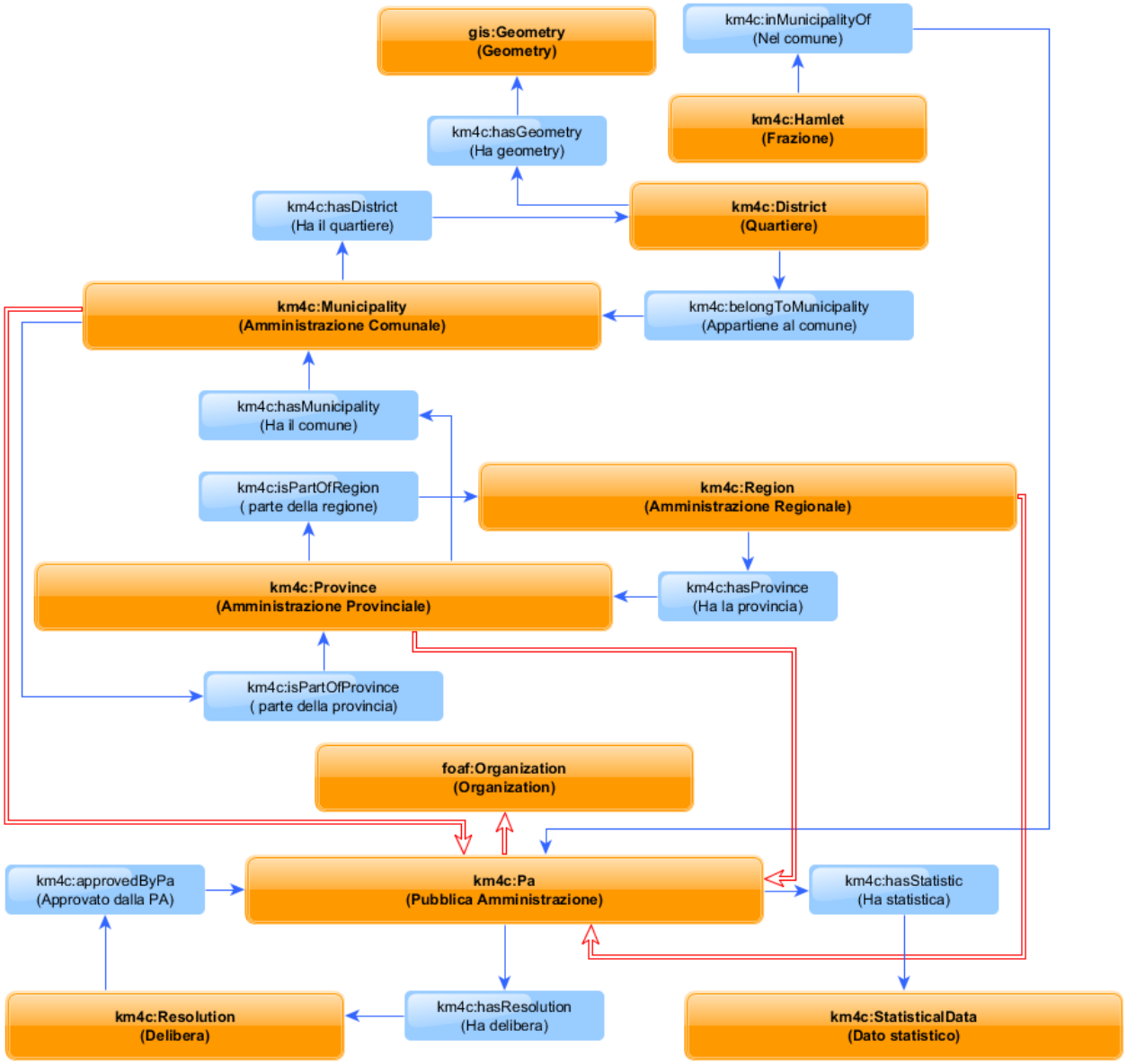



*Figura 1 Pubbliche Amministrazioni*

La classe principale è km4c:PA, che è stata definita come sottoclasse della foaf:Organization, collegamento che ci aiuta ad assegnare un più chiaro significato alla nostra classe. Le tre sottoclassi di km4c:PA sono automaticamente definite in base alla restrizione sulle object property (rappresentate in figura in azzurro). Ad esempio, la classe km4c:Region è definita come restrizione della classe km4c:PA sulla object property km4c:hasProvince, in modo tale che solo le PA che posseggono provincie possano essere classificate come regioni. Altro esempio: per definire gli elementi di PA che compongono la classe km4c:Municipality è stata invece utilizzata una restrizione sull' object property km4c:isPartOfProvince, quindi se una PA non è assegnata ad una provincia, non può essere considerata un comune.

Nella costituzione della gerarchia all'interno della classe PA, per ciascun passaggio, sono state definite coppie di object property inverse: km4c:hasProvince e km4c:isPartOfRegion, km4c:hasMunicipality e km4c:isPartOfProvince.

Collegata alla classe km4c:PA attraverso l' object property km4c:hasResolution, troviamo la classe km4c:Resolution, le cui istanze sono rappresentate dalle delibere approvate appunto dalle varie PA. km4c:hasResolution ha la sua object property inversa, cioè km4c:approvedByPa.

I Quartieri in cui una città può essere amministrativamente divisa sono rappresentati dalla classe km4c:District; questa classe è direttamente connessa alla classe km4c:Municipality attraverso una coppia di proprietà inverse, km4c:belongToMunicipality e km4c:hasDistrict. Semanticamente simile al km4c:District è la classe km4c:Hamlet, che rappresenta ogni porzione di territorio all'interno di un comune a cui sia stato assegnato un nome, quindi non soltanto i Quartieri che sono ufficialmente istituiti nelle grandi città. La classe km4c:Hamlet è collegata al comune dove si trova, attraverso la proprietà km4c:inMunicipalityOf.

L'ultima classe presente in questa macroclasse è km4c:StatisticalData: vista la grande quantità di dati statistici legati sia ai vari comuni, alla regione, ma anche alle singole strade, è stata inserita questa classe, condivisa sia dalla macro-classe Amministrazione che dallo Stradario. Come vedremo poi anche nella descrizione della prossima macro-classe, km4c:StatisticalData è collegata sia a km4c:Pa che a km4c:Road, attraverso l'object property km4c:hasStatistic.

La macro-classe Stradario è invece riportata in Figura 2.

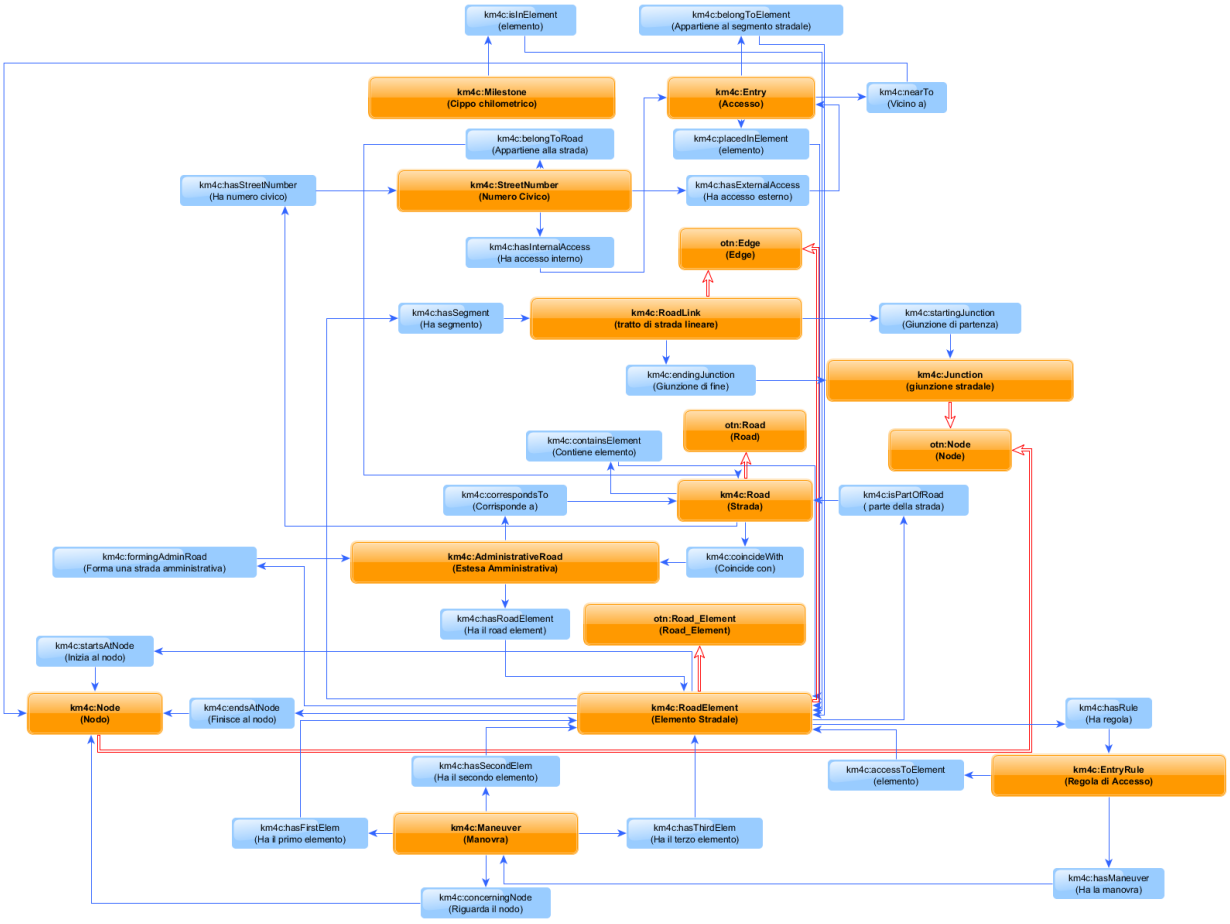



*Figura 2 Grafo stradale*

È evidente che tale macro-classe è la più complessa, in quanto rappresenta tutto il grafo strade, i vari numeri civici, gli accessi a questi ultimi, ma anche le regole di accesso alle varie strade e le manovre consentite.

La classe principale, al centro della macro-classe Stradario, è il km4c:RoadElement, definito come sottoclasse del corrispondente elemento all'interno dell'ontologia OTN, cioè Road_Element. Ciascun km4c:RoadElement è delimitato da un nodo iniziale ed un nodo finale, individuabili tramite le object property km4c:startsAtNode ed km4c:endsAtNode, che collegano la classe in oggetto alla classe km4c:Node. Nella definizione della classe km4c:RoadElement sono state specificate alcune restrizioni, legate appunto alla classe km4c:Node: un elemento stradale deve avere definite entrambe le object property km4c:startsAtNode e km4c:endsAtNode, entrambe con una cardinalità precisamente pari ad 1. Uno o più elementi stradali formano una strada: la classe Road è infatti definita come sottoclasse della corrispondente classe nella OTN, cioè l'omonima Road, e con restrizione sulla cardinalità dell'object property km4c:containsElement (che ha proprietà inversa km4c:isPartOfRoad), che deve essere minimo pari ad 1, cioè non può esistere una strada che non contiene almeno un elemento stradale. Anche la classe km4c:AdministrativeRoad, che rappresenta le estese amministrative, è collegata alla classe km4c:RoadElement tramite le due object property inverse km4c:hasRoadElement e km4c:formAdminRoad, mentre è collegata con una sola object property alla classe km4c:Road (km4c:coincideWith). Chiariamo bene la relazione che c'è tra km4c:Road, km4c:AdministrativeRoad e km4c:RoadElement: un'istanza di Road può essere collegata a più istanze di km4c:AdministrativeRoad (ad esempio se una strada attraversa il confine tra due provincie), ma vale anche il contrario (ad esempio quando una strada provinciale attraversa un centro città e assume differenti nomi), esiste cioè tra le due classi una relazione N:M. Entrambe però hanno una relazione 1:N con i km4c:RoadElement, cioè sia una istanza di Road che di km4c:AdministrativeRoad, sono collegate a più istanze di km4c:RoadElement.

Su ciascun elemento stradale è possibile definire delle restrizioni di accesso, individuate dalla classe km4c:EntryRule, che è connessa alla classe km4c:RoadElement attraverso due object property inverse, cioè km4c:hasRule e km4c:accessToElement. La classe km4c:EntryRule è definita con una restrizione sulla minima cardinalità della object property km4c:accessToElement (imposta pari a 0), che nella maggior parte dei casi ha associato un 1 solo elemento stradale, ma in alcuni casi eccezionali tale associazione viene a mancare. Le regole di accesso permettono di definire in modo univoco un permesso o limitazione di accesso sia sugli elementi stradali (per esempio a causa della presenza di una ZTL) come appena visto, ma anche sulle manovre; per questo motivo la classe km4c:Maneuver e la classe km4c:EntryRule sono collegate dall'object property km4c:hasManeuver. Per manovra si intendono principalmente le manovre di svolta obbligatorie, prioritarie o proibite, che sono descritte indicando l'ordine degli elementi stradali che interessano. Analizzando i dati provenienti dal Grafo Strade è stato verificato che in taluni casi le manovre interessano tre differenti elementi stradali e quindi, per rappresentare la relazione che intercorre tra le classi km4c:Maneuver ed km4c:RoadElement sono state definite tre object property: km4c:hasFirstElem, km4c:hasSecondElem ed km4c:hasThirdElem, oltre all'object property che lega una manovra alla giunzione che interessa, cioè km4c:concerningNode (perché una manovra avviene sempre in prossimità di un nodo). Nella definizione della classe km4c:Maneuver sono state inoltre aggiunte restrizioni di cardinalità, fissata obbligatoriamente ad 1 per km4c:hasFirstElem e km4c:hasSecondElem e cardinalità massima ad 1 per km4c:hasThirdElem, in quanto per le manovre che interessano solo due elementi stradali, questa object property non è definita.

Come precedentemente accennato, ciascun elemento stradale è delimitato da due nodi (o giunzioni), uno di inizio e uno di fine. È stata quindi definita la classe km4c:Node, sottoclasse della omonima classe Node appartenente all'ontologia OTN. La classe km4c:Node è stata definita con una restrizione sulle data property geo:lat e geo:long, due proprietà ereditate dal fatto che la classe è stata definita come sottoclasse

della geo:SpatialThing appartenente all'ontologia Geo wgs84: ciascun nodo infatti può essere associato ad una ed una sola coppia di coordinate geografiche.

La classe Milestone rappresenta i cippi chilometrici che sono posizionati lungo le estese amministrative, cioè degli elementi puntuali che identificano il valore della chilometrica in quel preciso punto, ovvero la progressiva del tracciato rispetto al punto d'inizio. Un milestone può essere associato ad un'unica km4c:AdministrativeRoad, ed è quindi stata definita una restrizione sulla cardinalità associata alla object property km4c:isInElement, imposta esattamente pari ad 1. Anche la classe km4c:Milestone è definita come sottoclasse della geo:SpatialThing, ma stavolta la presenza delle coordinate non è obbligatoria (restrizione sulla cardinalità massima che deve essere pari ad uno, ma che comunque non esclude la possibile mancanza del valore).

Il numero civico serve a definire un indirizzo, ed è sempre logicamente relazionato ad almeno un accesso, infatti ad ogni civico corrisponde sempre un solo accesso esterno, che può essere diretto o indiretto; a volte può essere presente anche un accesso interno. Vedendo tale relazione dal punto di vista dell'accesso, si può invece affermare che ciascuno di questi è connesso logicamente ad almeno un numero civico. Sono quindi state definite le classi km4c:StreetNumber ed km4c:Entry.

Con i dati posseduti è possibile collegare la classe km4c:StreetNumber sia alla classe km4c:RoadElement che alla classe km4c:Road, rispettivamente attraverso le object property km4c:placedInElement e km4c:belongToRoad. Tale informazione risulta effettivamente ridondante e nel caso si ritenga opportuno potrebbe essere eliminato il collegamento alla classe km4c:Road a favore di quello verso km4c:RoadElement, più facilmente ricavabile dai dati; è infatti per questo motivo che per l'object property km4c:belongToRoad è stata definita anche l'inversa km4c:hasStreetNumber.

All'interno dell'ontologia quindi, la classe km4c:StreetNumber è definita come avente una restrizione sulla cardinalità della object property km4c:belongToRoad, che deve essere pari ad 1.

Anche la classe km4c:Entry può essere collegata sia al numero civico che all'elemento stradale in cui si trova. La relazione tra km4c:Entry e km4c:StreetNumber, è definita da due object property, km4c:hasInternalAccess e km4c:hasExternalAccess, sulle quali sono state definite delle restrizioni di cardinalità, in quanto, come già detto in precedenza, un numero civico avrà sempre un solo accesso esterno, mentre potrebbe avere anche un accesso interno (quest'ultima restrizione è infatti stata definita impostando la massima cardinalità pari ad 1, cioè sono ammessi i valori 0 ed 1). Anche la classe km4c:Entry è definita come sottoclasse della geo:SpatialThing, ed è quindi possibile associare massimo una coppia di coordinate geo:lat e geo:long a ciascuna istanza (restrizione sulla cardinalità massima delle due data property della Geo ontology, imposta a 1, così che possano assumere o un valore, o nessuno).

La macro-classe Stradario è collegata alla macro-classe Amministrazione attraverso due differenti object property, km4c:ownerAuthority e km4c:managingAuthority, che rappresentano chiaramente come suggerisce il nome, rispettivamente la pubblica amministrazione che possiede l'estesa amministrativa, o la pubblica amministrazione che gestisce l'elemento stradale.

Da un punto di vista cartografico però, ciascun elemento stradale non è una retta, ma una spezzata, che seguirà l'effettivo andamento della strada. Per poter rappresentare questa situazione, sono state quindi aggiunte le classi km4c:RoadLink e km4c:Junction: grazie all'interpretazione dei file KMZ, abbiamo recuperato la serie di coordinate che definiscono ciascun road element, e ciascuno di questi punti, verrà inserito nell'ontologia come una istanza di Junction (definita come sottoclasse di geo:SpatialThing, con

obbligatoriamente un'unica coppia di coordinate). Ciascun segmentino tra due Junction è invece un'istanza della classe km4c:RoadLink, che è definita con una restrizione sulle object property km4c:endJunction e km4c:startJunction (entrambe devono obbligatoriamente avere cardinalità pari ad 1), che connettono le due classi.

La modellazione dell'infrastruttura stradale è stata migliorata nella versione 1.6.5 con l'inclusione di concetti volti a rappresentare le corsie e le restrizioni del traffico. Una panoramica dei miglioramenti è fornita in Figura 3.

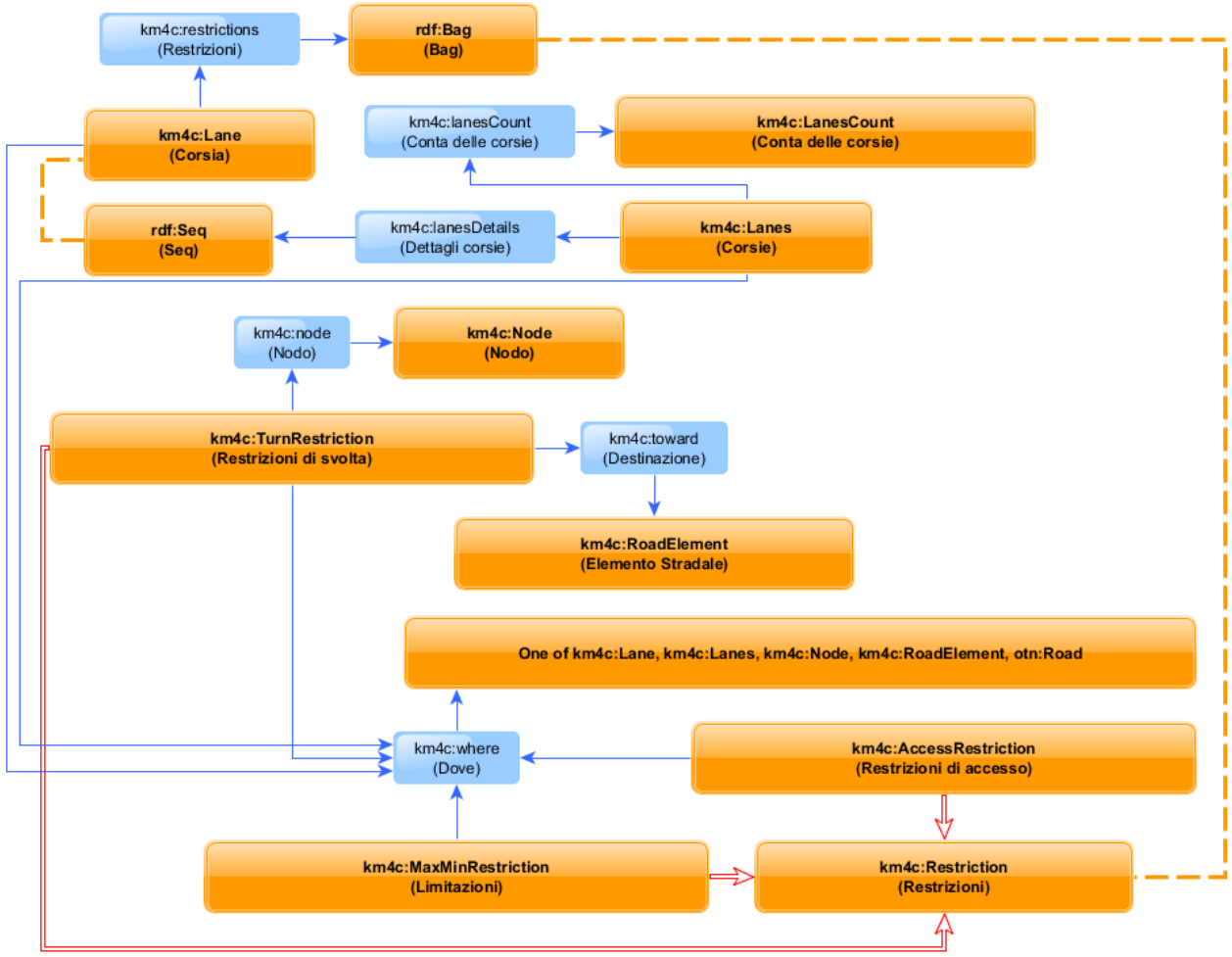



*Figura 3 Corsie e restrizioni*

Per quanto riguarda la modellazione delle corsie, il concetto centrale è km4c:Lanes. Un'unica istanza del concetto km4c:lanes rappresenta le corsie che possono essere trovate segnate su una strada o un segmento di strada, in una direzione di traffico specifica che viene espressa attraverso la proprietà km4c:direction. La proprietà km4c:where indica dove sono segnate le corsie. Può trattarsi di una strada (km4c:Road), o di un elemento stradale (km4c:RoadElement). La proprietà inversa di km4c:where è km4c:lanes. Poiché alcune delle corsie potrebbero essere riservate a specifiche tipologie di veicoli, e dobbiamo rappresentare questi vincoli, ogni istanza di km4c:Lanes ha una proprietà km4c:lanesCount il cui valore è un'istanza del concetto km4c:LanesCount, le cui proprietà ricalcano i loro nomi dalle diverse tipologie di veicoli che potrebbero avere corsie dedicate, ed assumono valori numerici (interi) che indicano quante delle corsie sono dedicate alle diverse tipologie di veicoli. La proprietà km4c:undesignated viene tipicamente impostata per un'istanza di km4c:LanesCount, e indica quante delle corsie non sono riservate a

una specifica tipologia di veicoli. A volte può capitare che una o più corsie abbiano restrizioni particolari, a parte il fatto di essere riservate a specifiche tipologie di veicoli. Ad esempio, alcune corsie potrebbero avere un limite di velocità, mentre altre potrebbero avere un limite di velocità diverso o non avere un limite di velocità. Poiché dobbiamo rappresentare queste restrizioni, la proprietà km4c:lanesDetails è stata introdotta e può essere trovata impostata per un'istanza di km4c:Lanes. Il suo valore è una rdf:Seq di istanze della classe km4c:Lane, ognuna delle quali rappresenta una delle corsie. Una rdf:Seq è un insieme ordinato di istanze di un qualche tipo. Le corsie compaiono al suo interno da sinistra a destra rispetto alla direzione di guida. Ogni istanza del concetto km4c:Lane dovrebbe avere una proprietà km4c:where che indica l'istanza di km4c:Lanes di cui la km4c:Lane è parte. Inoltre, ogni istanza di km4c:Lane dovrebbe avere una proprietà km4c:position, con valore numerico (intero) che indica la posizione della corsia da sinistra a destra rispetto alla direzione di guida. Inoltre, un'istanza di km4c:Lane potrebbe avere una proprietà km4c:turn, una stringa di testo che indica la manovra obbligatoria che i conducenti dovranno eseguire una volta raggiunta la fine della corsia. Infine, un'istanza di km4c:Lane potrebbe avere una proprietà km4c:restrictions, il cui valore è un rdf:Bag di istanze della classe km4c:Restriction. Un rdf:Bag è un insieme non ordinato di istanze di un qualche tipo. A km4c:Restriction rappresenta una limitazione del traffico. Le restrizioni del traffico possono anche essere applicate a istanze di km4c:Road e km4c:RoadElement. In questi casi, la proprietà km4c:where della km4c:Restriction indica la km4c:Road o il km4c:RoadElement soggetto alla restrizione. Esistono tre tipi di restrizioni: restrizioni di svolta, restrizioni di accesso e restrizioni di raggio. Le restrizioni di svolta sono modellate attraverso il concetto km4c:TurnRestriction, e rappresentano le manovre obbligatorie e proibite. Per ogni istanza di km4c:TurnRestriction, una proprietà km4c:where è prevista per indicare l'origine, una proprietà km4c:toward è prevista per indicare la destinazione, e una proprietà km4c:restriction è prevista per indicare se la manovra è obbligatoria o proibita. È possibile trovare altre proprietà se la limitazione si applica solo in giorni e orari specifici o se si applica solo a categorie specifiche di veicoli e così via. Le restrizioni di accesso sono rappresentate dal concetto km4c:AccessRestriction e rappresentano il divieto di accesso a una strada o ad un segmento di strada. Spesso si applicano a una o più categorie di veicoli, nel qual caso la proprietà km4c:who indica le categorie di veicoli a cui si applica la restrizione. Inoltre, si applicano spesso a una direzione specifica del traffico, nel qual caso la proprietà km4c:direction è impostata per indicare la direzione del traffico a cui si applica la restrizione. Tutte le istanze di km4c:AccessRestriction dovrebbero avere una proprietà km4c:access che fornisce l'interpretazione corretta della restrizione, indicando se le tipologie di veicoli indicate nella proprietà km4c:who sono autorizzate ad accedere in via esclusiva, o se sono invece escluse. Per quelle restrizioni di accesso che si applicano solo in condizioni particolari, come ad esempio in alcuni giorni e/o orari soltanto, viene specificata anche la proprietà km4c:condition che descrive la condizione. Esistono infine anche restrizioni legate al peso o alle dimensioni dei veicoli, oppure alla loro velocità. Sono modellate attraverso il concetto km4c:MaxMinRestriction. Si prevede che le istanze di questo concetto abbiano una proprietà km4c:where che indica la strada, il segmento di strada o corsia a cui si applica la restrizione. Ci si aspetta anche che abbiano una proprietà km4c:what, insieme di proprietà descrive ciò a cui si riferisce la restrizione di intervallo (velocità, larghezza, altezza, peso o altro) e una proprietà km4c:limit dove è indicato l'intervallo stesso. Se la limitazione si applica solo a una delle direzioni del traffico, la proprietà km4c:direction viene impostata ed indica la direzione del traffico a cui si applica la restrizione. Alla fine, se la restrizione si applica solo in condizioni particolari, tali condizioni sono rappresentate attraverso la proprietà km4c:condition.

Per la terza macro classe, i punti di interesse detti anche servizi (PoI), illustrati nella Figura 4, è stato definita una generica classe km4c:Service ed è stato definito un insieme di classi e sottoclassi che traggono ispirazione dalla classificazione ATECO (la classificazione del codice ISTAT delle attività economiche). In una

prima versione dell'Ontologia, le classi e le sottoclassi erano state invece definite sulla base di una classificazione regionale delle attività commerciali. Quei servizi che erano stati importati nella Knowledge Base in quel momento, sono stati riclassificati.

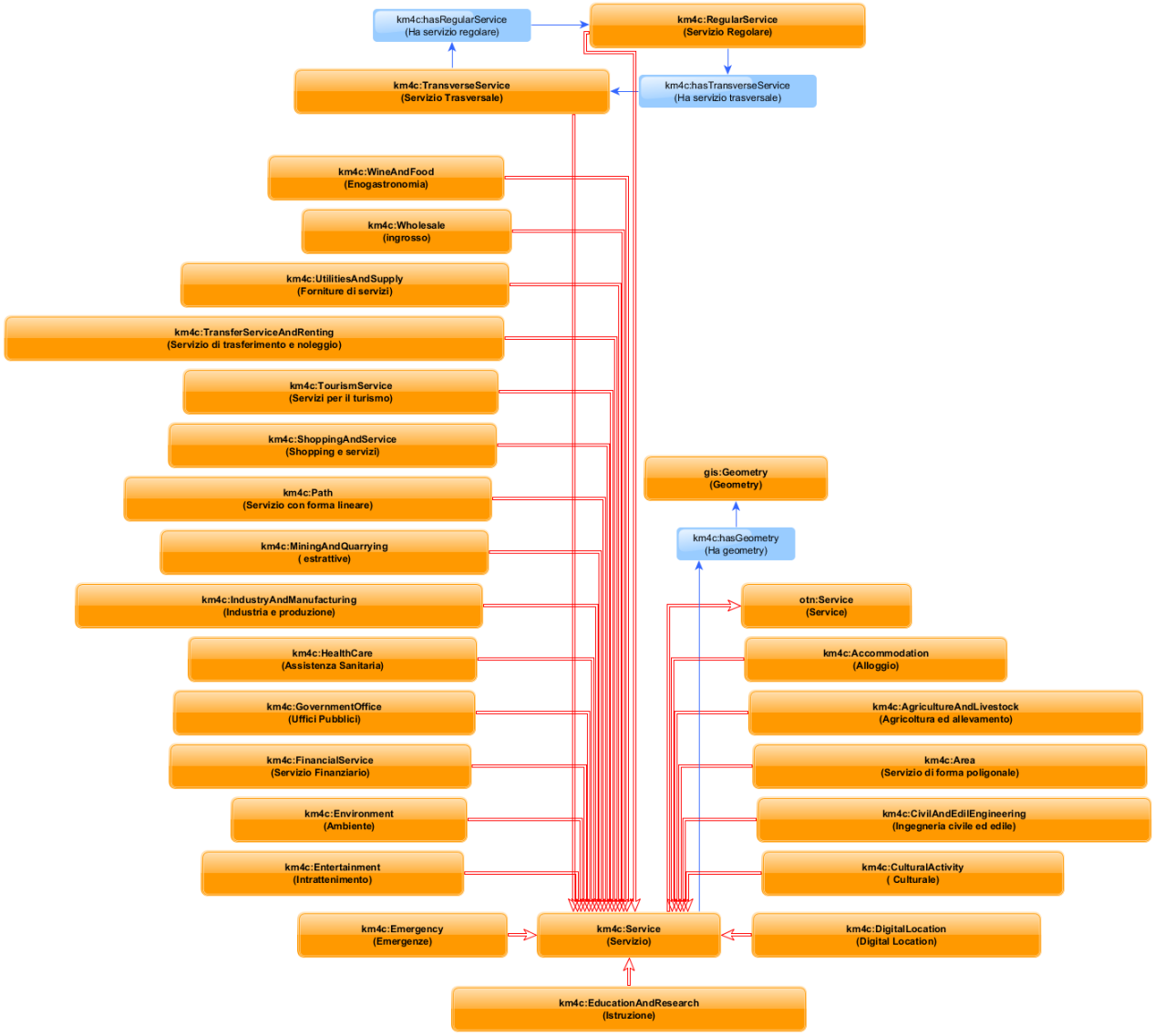



*Figura 4 Servizi*

Tuttavia, la nuova categorizzazione non è stata creata con una corrispondenza totale con l'intero elenco dei codici ATECO. Invece, è stata creata allo scopo di fornire maggiori dettagli per quelle sottoclassi che sono di maggiore interesse per i nostri scopi (fornitori di servizi, vendita al dettaglio vendite, ecc.), rimanendo più generale per quelle categorie che sono meno rilevanti per i nostri scopi, come il commercio all'ingrosso, l'industria e così via.

Attualmente quindi sono state individuate le classi Accommodation, GovernmentOffice, TourismService, TransferServiceAndRenting, CulturalActivity, FinantialService, ShoppingAndService, HealtCare, EducationAndResearch, Entertainment, Emergency, WineAndFood, IndustryAndManufacturing, AgricultureAndLivestock, UtilitiesAndSupply, CivilAndEdilEngineering, Wholesale, Advertising, MiningAndQuarrying and Environment. Ciascuna di esse presenta una serie di sottoclassi che definiscono quali tipi di servizi appartengono a tale classe: ad esempio per la classe Accommodation, le sottoclassi

definite sono Holiday_village, Hotel, Summer_residence, Rest_home, Hostel, Farm_house, Beach_resort, Agritourism, Vacation_resort, Day_care_centre, Camping, Boarding_house, Other_Accommodation, Mountain_shelter, Religiuos_guest_house, Bed_and_breakfast, Historic_residence e Summer_camp. Degna di menzione è anche la classe Noise_level_sensor, introdotta a fine 2018 come sottoclasse di Environment ed immediatamente utilizzata per rappresentare i rilevatori di rumore installati nella città di Helsinki, e la classe People_counter introdotta nel 2019 per rappresentare dispositivi come i pax counter (ESP32, …). Merita una menzione a parte anche la classe Digital_signage, specializzazione di TourismService, introdotta a fine 2019 per rappresentare alcuni pannelli informativi elettronici presenti nella città di Firenze e in alcune località limitrofe.

Attualmente all'interno di Km4City sono state definite 512 sottoclassi di servizi così divise:

| | |
|---|---|
| Accommodation | 18 |
| FinancialService | 10 |
| Environment | 12 |
| MiningAndQuarrying | 5 |
| Advertising | 2 |
| Wholesale | 10 |
| CivilAndEdilEngineering | 9 |
| UtilitiesAndSupply | 30 |
| AgricultureAndLivestock | 7 |
| IndustryAndManufacturing | 54 |
| EducationAndResearch | 33 |
| Entertainment | 27 |
| Emergency | 14 |
| TourismService | 15 |
| HealthCare | 25 |
| WineAndFood | 21 |
| CulturalActivity | 26 |
| ShoppingAndService | 140 |
| GovernmentOffice | 15 |
| TransferServiceAndRenting | 39 |

La classe Servizio ha anche una coppia di sottoclassi che consentono di identificare più rapidamente servizi non puntuali. Queste classi, ovvero km4c:Path e km4c:Area, rappresentano rispettivamente quei servizi che possono essere rappresentati attraverso una linea spezzata su una mappa e quei servizi che possono essere rappresentati come poligoni. Pertanto, ogni servizio non puntuale rientra necessariamente in una di queste due classi.

È stata anche definita un'ulteriore categorizzazione dei servizi, in servizi *regolari* (km4c:RegularService) e servizi *trasversali* (km4c:TransversalService). Questa categorizzazione si è rivelata necessaria poiché, con l'inclusione di nuovi servizi, alcuni servizi sono stati scoperti come correlati ad altri servizi. Ad esempio, un ristorante (servizio regolare) può offrire ai propri clienti un servizio WiFi gratuito (servizio trasversale). Per gestire questa situazione, è quindi necessario collegare i due servizi tra loro, ma dovrebbe anche essere possibile identificare solo il Wifi, come servizio a sé stante. È quindi chiaro che esistono due tipi di servizi, servizi primari (o regolari) e servizi secondari (o trasversali). I servizi trasversali sono connessi a servizi regolari attraverso la proprietà dell'oggetto km4c:hasRegularService, la cui inversa è la proprietà km4c:hasTransversalService.

Un'ulteriore parte della macro-area dei Punti di Interesse è rappresentata dall'integrazione delle DigitalLocation dal Comune di Firenze. Si tratta ad esempio di percorsi di jogging, aree verdi e giardini, monumenti e luoghi storici, per un totale di circa 3200 elementi diversi. Queste posizioni digitali sono state rappresentate come istanze della classe km4c:Service. Per quelle che erano già state importate nella Knowledge Base prima che venisse compilata la lista delle DigitalLocation, due istanze di km4c:Service possono essere trovate che rappresentano lo stesso servizio nel mondo reale. Le due istanze sono correlate tra loro tramite la proprietà owl:sameAs.

L'introduzione dell'Ontologia geoSPARQL, ha inoltre permesso di associare ai servizi una forma geometrica: un servizio infatti può essere individuato da un punto (nel caso di un negozio, un monumento, etc.), oppure da un percorso (una linea spezzata che connette vari punti di interesse, ad esempio un percorso turistico) oppure da un'area (un poligono che ad esempio corrisponde ad un giardino pubblico o un parco).

Un'altra classe che è stata aggiunta a questa macro-classe è la classe km4c:Event, la quale eredita la maggior parte delle sue proprietà dalla classe schema:Event, ma non solo. Tale classe è connessa tramite la object property schema:location, alla classe km4c:Service, la quale indica il luogo dell'evento

La quarta macro-classe, il trasporto pubblico locale, illustrato nella figura 5, presenta dati e classi ispirati alle linee di autobus metropolitane della rete tranviaria e ferroviaria di Firenze. Tuttavia, il modello si è dimostrato applicabile anche alle linee di autobus urbani in territori diversi dal Comune di Firenze, e anche alle linee di autobus suburbane.

Analizziamo prima la parte relativa al trasporto metropolitano fiorentino che comprende trasporto su gomma e tramvia.

Ciascun lotto TPL, rappresentato dalla classe km4c:Lot, è composto da un certo numero di Linee di autobus o tranvia (classe km4c:PublicTransportLine), e tale relazione è rappresentata dalla object property km4c:isPartOfLot, che collega ciascuna istanza di km4c:PublicTransportLine al rispettivo km4c:Lot. La classe km4c:PublicTransportLine è definita come sottoclasse della otn:Line. Ciascuna linea prevede almeno una corsa, individuata attraverso un codice fornito dall'azienda TPL; la classe km4c:PublicTransportLine è stata infatti collegata alla classe km4c:Ride attraverso l'object property km4c:scheduledOnLine, sulla quale è definita anche una limitazione di cardinalità pari esattamente ad 1, perché a ciascuna corsa può essere associata una sola linea.

Ciascuna corsa segue almeno un percorso, e i percorsi possono essere in numero variabile anche se riferiti ad una stessa linea: nella maggior parte dei casi sono 2, cioè percorso in avanti e percorso indietro, ma a volte arrivano ad essere anche 3 o 4, in base ad eventuali prolungamenti di percorsi o deviazioni magari effettuate solo in particolari orari. L'object property km4c:onRoute è utilizzata anche per collegare la classe km4c:Ride, rappresentante le singole corse definite dall'operatore TPL su appunto un certo percorso. Tramite l'object property gis:hasGeometry, le istanze della classe km4c:Route possono invece essere collegate all'istanza della classe gis:Geometry che contiene la line string rappresentante il percorso effettivo di quella corsa.

Ciascun percorso è considerato come costituito da una serie di tratti stradali delimitati da fermate successive: per modellare questa situazione, si è scelto di definire due object property che collegano le classi km4c:Route e km4c:RouteSection, km4c:hasFirstSection e km4c:hasSection, in quanto, da un punto di vista cartografico, volendo rappresentare il percorso che segue un certo autobus, definito quale sia il primo segmento e conosciuta la fermata di partenza, si è in grado di recuperare tutti gli altri segmenti che

costituiscono il percorso e partendo dalla seconda fermata individuata come fermata differente dalla prima, ma contenuta nel primo segmento, siamo in grado di ricostruire l'esatta sequenza delle fermate e quindi dei segmenti, che costituiscono l'intero percorso. A tale scopo è stata definita quindi anche l'object property km4c:hasFirstStop, che collega le classi km4c:Route e km4c:BusStop.

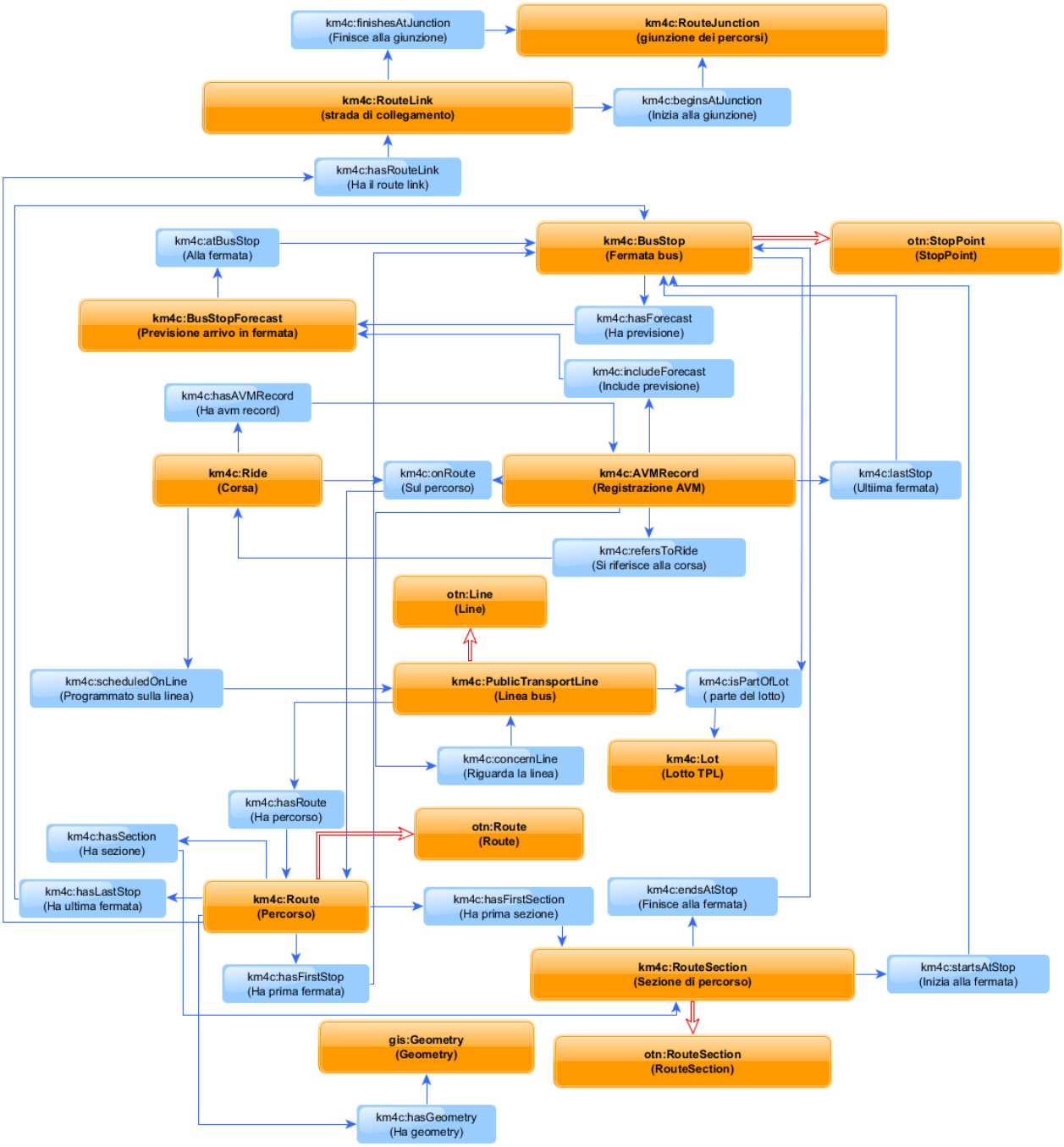



*Figura 5 Trasporto pubblico locale*

Applicando lo stesso tipo di modellazione utilizzato per gli elementi stradali, sono state definite due object property km4c:endsAtStop e km4c:startsAtStop, per collegare ogni istanza di km4c:RouteSection a due istanze della classe km4c:BusStop, classe a sua volta definita come sottoclasse della otn:StopPoint. Ciascuna fermata è inoltre connessa alla classe km4c:Lot, attraverso l'object property km4c:isPartOfLot,

relazione 1:N in quanto esistono fermate condivise da linee urbane e extraurbane che quindi appartengono a due differenti lotti.

Disponendo inoltre delle coordinate di ciascuna fermata, la classe km4c:BusStop è stata definita come sottoclasse della geo:SpatialThing, ed è stata inoltre definita una cardinalità pari ad 1 per le due data property geo:lat e geo:long.

Volendo poi da un punto di vista cartografico rappresentare il percorso di un autobus, quindi una istanza di km4c:Route, abbiamo bisogno di rappresentare la spezzata che compone ciascun tratto di strada attraversato dal mezzo stesso e per far ciò, è stata riutilizzata la modellazione precedentemente usata per gli elementi stradali: ciascun percorso lo possiamo vedere come un insieme di piccoli segmenti, ciascuno delimitato da due giunzioni: sono quindi state definite le classi km4c:RouteLink e km4c:RouteJunction, e le object property km4c:beginsAtJunction e km4c:finishesAtJunction. La classe km4c:RouteLink è stata definita come restrizione di cardinalità su entrambe le object property appena citate, imponendo che sia sempre pari ad 1. La classe km4c:Route è invece stata collegata alla classe km4c:RouteLink attraverso la object property km4c:hasRouteLink.

La classe km4c:BusStop è inoltre collegata alla classe km4c:Road della macro-classe Grafo Strade, tramite l'object property km4c:isInRoad, grazie alla quale risulta più immediata una localizzazione a livello di comune delle singole fermate.

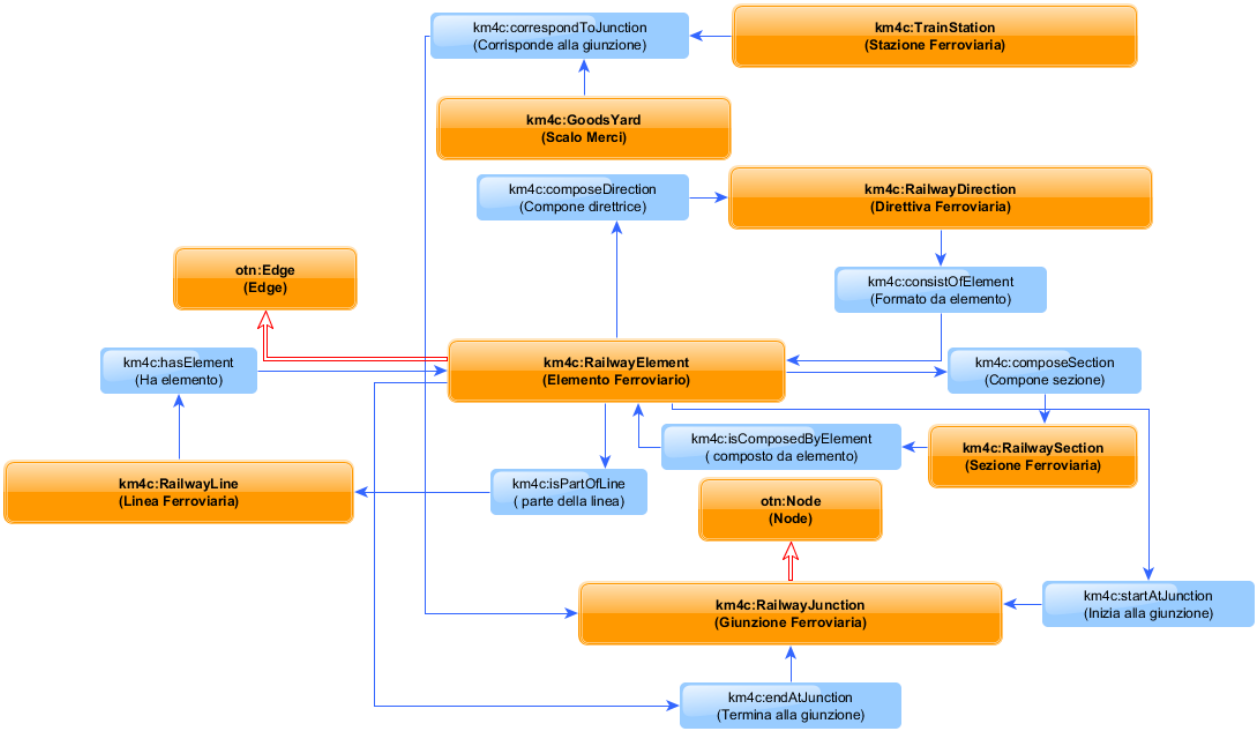



*Figura 6 Grafo ferroviario*

La parte relativa al Grafo Ferroviario è formata principalmente dalla classe km4c:RailwayElement (definita come sottoclasse della OTN:Edge), le cui istanze rappresentano un singolo elemento ferroviario; ciascun elemento ferroviario può comporre una direttrice ferroviaria, cioè una linea ferroviaria avente particolari caratteristiche di importanza per il volume dei traffici e le relazioni di trasporto che su di essa si svolgono, e che congiunge tra loro centri o nodi principali della rete ferroviaria, oppure una sezione ferroviaria (tratto

di linea in cui si può trovare solo un treno per volta che solitamente è preceduto da un segnale "di protezione" o "di blocco") oppure una linea ferroviaria (cioè l'infrastruttura atta a far viaggiare treni o altri convogli ferroviari tra due località di servizio). Inoltre ciascun elemento ferroviario, inizia e termina in una giunzione ferroviaria, cioè un istanza della classe km4c:RailwayJunction, definita come sottoclasse della OTN:Node. Si hanno inoltre le classi km4c:TrainStation che rappresenta appunto una stazione ferroviaria, e km4c:GoodsYard cioè uno scalo merci, e solitamente entrambi corrispondono ad un'unica istanza di km4c:RailwayJunction.

Anche la classe km4c:RailwayElement è connessa alla classe gis:Geometry, in quanto ciascun elemento ferroviario può essere visto come una line string che rappresenta il vero andamento della linea ferroviaria in quel tratto.

Dal febbraio 2019, l'ontologia include un ulteriore concetto che completa il quadro del trasporto pubblico, vale a dire la stazione della metropolitana, rappresentata attraverso la classe Subway_station, introdotta sulla spinta dei dati raccolti relativamente alla città di Helsinki ed in particolare dell'agenzia del trasporto pubblico HLS nell'ambito del progetto Snap4City. Legati al trasporto sono anche i km4c:Traffic_flasher, semafori lampeggianti che segnalano code e altre anomalie, così come i km4c:Variable_message_sign. Altri semafori che hanno un concetto dedicato nell'ontologia sono i km4c:Tram_traffic_light.

La Km4City Ontology include anche una serie completa di concetti volti a modellare gli eventi e le rilevazioni in tempo reale e i dispositivi che eseguono tali rilevamenti e attivano tali eventi, con la loro strutturazione e organizzazione. È la macro-area dei Sensori, che può essere a sua volta suddivisa in diverse sezioni: parcheggi, previsioni meteo, traffico, trasporto pubblico, e beacons.

Per quanto riguarda i parcheggi, la classe km4c:TransferService è collegata alla classe km4c:CarParkSensor, che rappresenta il sensore installato in un certo parcheggio e al quale saranno collegate le istanze della classe km4c:SituationRecord, che rappresentano lo stato di un certo parcheggio ad un certo istante. Il primo collegamento, cioè quello tra la classe km4c:TransferService e km4c:CarParkSensor, è realizzato attraverso due object property inverse, cioè km4c:observeCarPark e km4c:hasCarParkSensor, mentre il collegamento tra la classe km4c:CarParkSensor e km4c:SituationRecord è realizzato tramite le object property inverse km4c:relatedToSensor e km4c:hasRecord. La classe km4c:SituationRecord permette di memorizzare le informazioni relative a quanti posti liberi ed occupati ci sono in un determinato istante (anch'esso memorizzato) nei vari parcheggi della Regione Toscana.

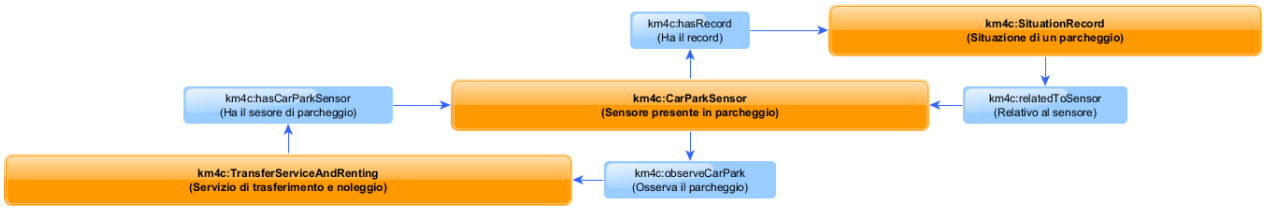



*Figura 7 Parcheggi*

Per quanto riguarda le previsioni meteorologiche, esse sono disponibili per i differenti comuni (e sono quindi legate alla classe km4c:Municipality), grazie ai dati forniti dal consorzio LAMMA. Il consorzio aggiorna i report di ciascun comune una o due volte al giorno ed ogni report al suo interno riporta le previsioni di 5 gg suddivise in fasce, che presentano una maggior precisione (ed un maggior numero) per i giorni più vicini fino ad arrivare ad un'unica previsione giornaliera per il quarto ed il quinto giorno. Tale situazione è stata infatti rappresentata dalla classe km4c:WeatherReport connessa tramite l'object

property km4c:hasPrediction alla classe km4c:WeatherPrediction. Il comune invece è connesso ad un report tramite due object property inverse, km4c:refersToMunicipality e km4c:hasWeatherReport.

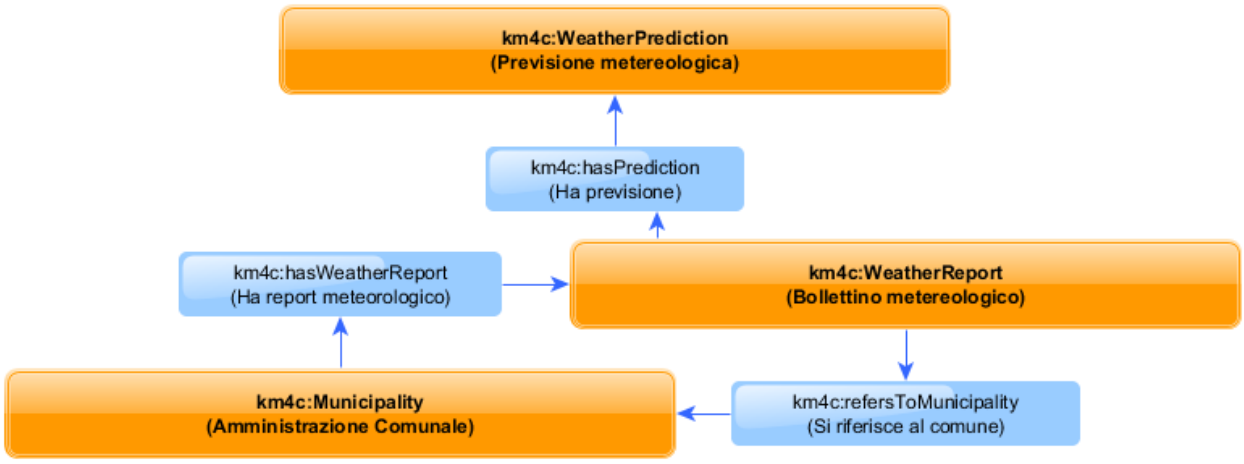



*Figura 8 Meteo*

Per quanto riguarda i sensori disposti lungo le strade della regione, essi permettono di fare differenti rilevazioni in relazione alla situazione del traffico. Grazie alle informazioni aggiuntive che sono fornite dall'Osservatorio dei Trasporti, è possibile geolocalizzare tutti i sensori con precisione. I sensori sono suddivisi in gruppi. Ciascun gruppo è rappresentato dalla classe km4c:SensorSiteTable, e ciascun sensore, cioè ciascuna istanza della classe km4c:SensorSite, si connette al proprio gruppo tramite la object property km4c:formsTable, ed è collegato anche alla strada su cui si trova, istanza della classe km4c:Road, attraverso la object property km4c:installedOnRoad. Ciascun sensore produce delle osservazioni, le quali sono rappresentate dalla classe km4c:Observation; tali osservazioni possono essere di 4 tipi cioè relative alla velocità media (sottoclasse km4c:TrafficSpeed), oppure relative al flusso di auto che passa davanti al sensore (sottoclasse TrafficFlow), alla concentrazione di traffico (sottoclasse km4c:TrafficConcentration) e infine relativi alla densità di traffico (sottoclasse km4c:TrafficHeadway). La classe km4c:Observation e la classe km4c:SensorSite sono collegate attraverso la proprietà km4c:hasObservation, e la sua inversa km4c:measuredBySensor. Le stesse proprietà sono utilizzate anche per i km4c:Weather_sensor.

Per quanto riguarda il trasporto pubblico in real-time, esso è legato prevalentemente ai sistemi AVM che si trovano installati su buona parte dei mezzi ATAF, ed è rappresentato principalmente attraverso due classi, km4c:AVMRecord e km4c:BusStopForecast: la prima classe rappresenta il record inviato dal sistema AVM, all'interno del quale, oltre alle informazioni relative all'ultima fermata effettuata (rappresentata tramite l'object property km4c:lastStop che collega km4c:AVMrecord a km4c:BusStop), alle coordinate GPS di localizzazione del mezzo, all'identificativo del mezzo e della linea, troviamo anche l'elenco delle prossime fermate con l'orario di passaggio previsto. Tale elenco risulta di lunghezza variabile e rappresenta le istanze della classe km4c:BusStopForecast. Quest'ultima classe è stata collegata alla classe km4c:BusStop attraverso l'object property km4c:atBusStop in modo tale da poter recuperare anche l'elenco delle eventuali linee previste in arrivo ad una certa fermata (la classe km4c:AVMRecord è stata infatti collegata anche alla classe km4c:Line tramite l'object property km4c:concernLine).

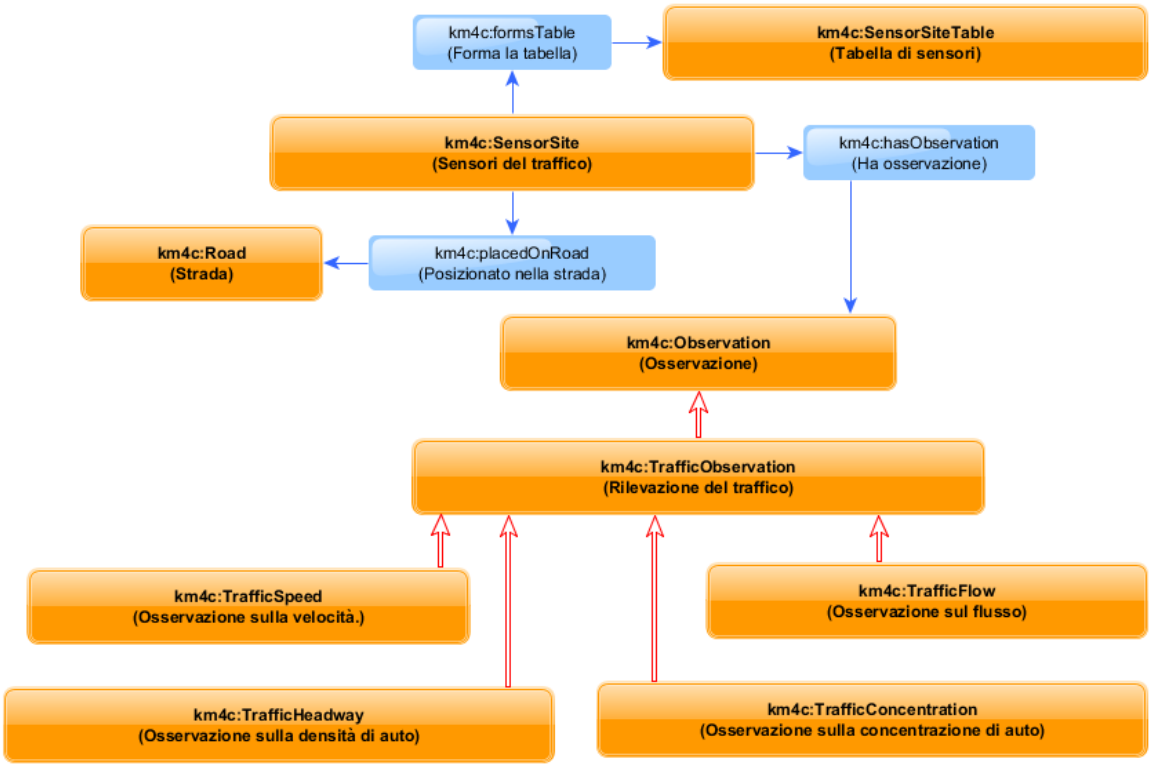



*Figura 9 Sensori di traffico*

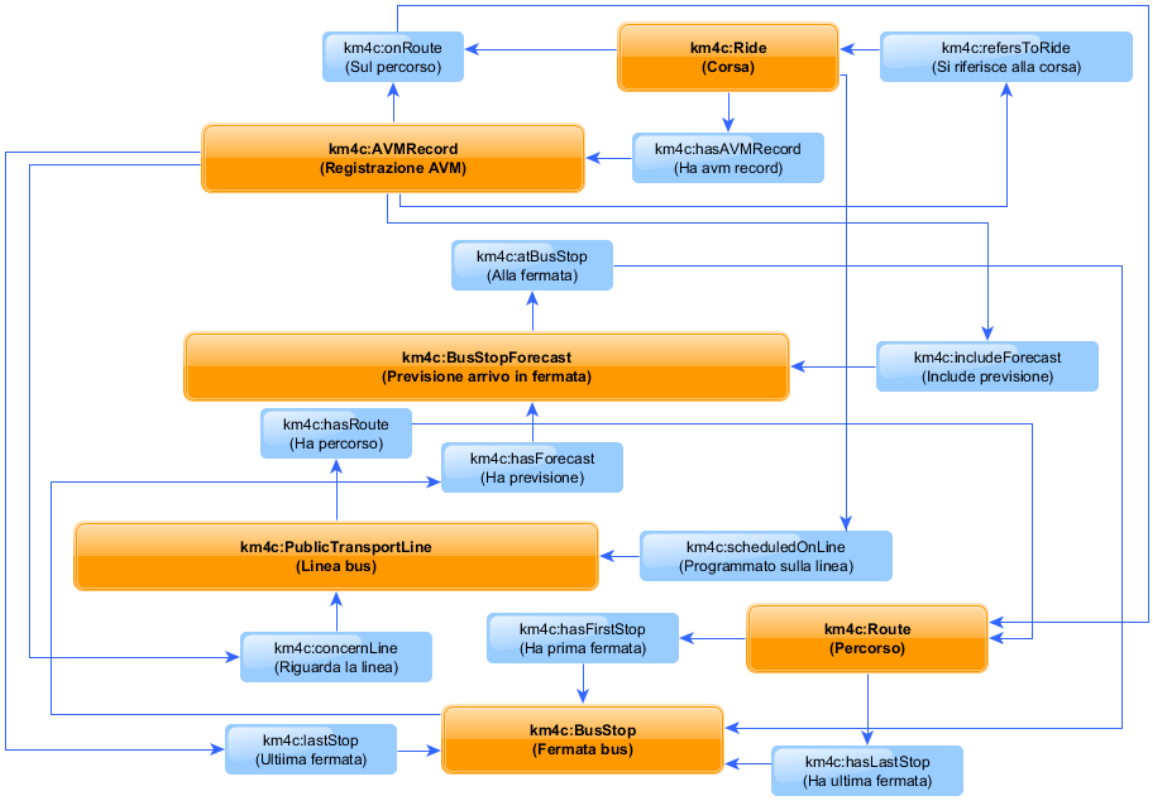



*Figura 10 Trasporto pubblico in tempo reale*

Per quanto riguarda infine i beacons, per essi sono state create due classi, la km4c:Beacon, che rappresenta il singolo dispositivo installato nelle città, e la km4c:BObservation, che rappresenta invece una singola

osservazione riportata da ciascun elemento funzionante; le due classi sono connesse tramite la coppia di object property inverse km4c:hasBObservation e km4c:measuredByBeacon. Ciascun beacon è univocamente identificato tramite un codice identificativo, e da altri tre codici, l'uuid (Universally Unique Identifier, che contiene 32 cifre esadecimali, divise in 5 gruppi separati dal trattino) che identifica univocamente il proprietario del beacon (quindi se un negozio possiede 10 beacons, tutti e 10 avranno lo stesso uuid), il major ed il minor che invece identificano rispettivamente un gruppo di beacons e il numero di ciascun elemento all'interno del gruppo identificato con uno stesso major. Sono inoltre disponibili delle coordinate che identificano il luogo in cui i beacons sono posizionati. Le informazioni che invece un beacon è in grado di restituire sono la potenza del segnale, le coordinate GPS dell'utente con cui è stato stabilito il contatto, e la data e ora del contatto.

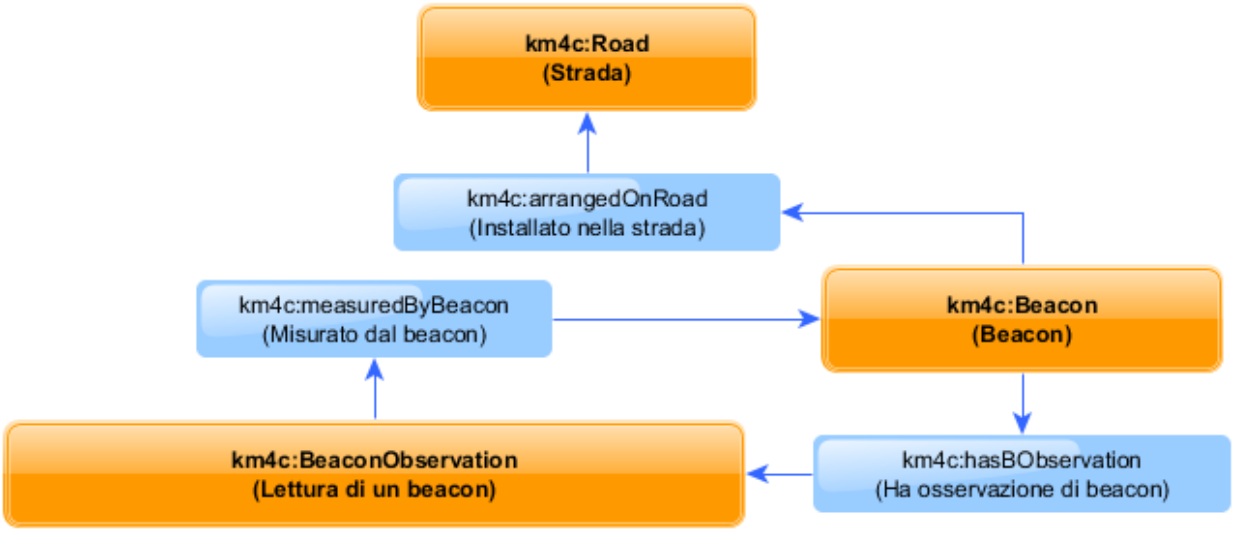



*Figura 11 Beacons*

Infine, l'ultima macro-classe, quella cioè relativa agli aspetti temporali, è stata soltanto "abbozzata" all'interno dell'ontologia, basandosi sulla Time ontology (http://www.w3.org/TR/owl-time/) e sull'esperienza fatta in altri progetti come OSIM. Si rende necessaria l'integrazione del concetto del tempo in quanto sarà di fondamentale importanza riuscire a calcolare differenze tra istanti di tempo, e l'ontologia Time ci viene in aiuto in questo.

Si definisce un URI fittizio #instantForecast, #instantAVM, #instantParking, #instantWreport, #instantObserv da associare in seguito all'URI identificativo della risorsa a cui si riferisce il parametro temporale, che può essere un'istanza di km4c:BusStopForecast, di km4c:AVMRecord, di km4c:SituationRecord, di km4c:WheatherReport o di km4c:Observation.

L'URI fittizio #instantXXXX, sarà dato dalla concatenazione di due stringhe: ad esempio, nel caso delle istanze di km4c:BusStopForecast, si andranno a concatenare il codice fermata (che ci permette di individuarle univocamente) e l'istante temporale nel formato desiderato. È necessario creare un URI fittizio che leghi l'istante temporale a ciascuna risorsa, per non creare ambiguità, in quanto potrebbero essere presenti istanti temporali identici associati a differenti risorse (anche se il formato con cui è espresso l'istante temporale ha una scala piuttosto fine).

L'ontologia Time permette di definire istanti come informazioni temporali puntuali, e di usarli come estremi per la definizione di intervalli e durate, che permettono di conseguire una maggiore espressività.

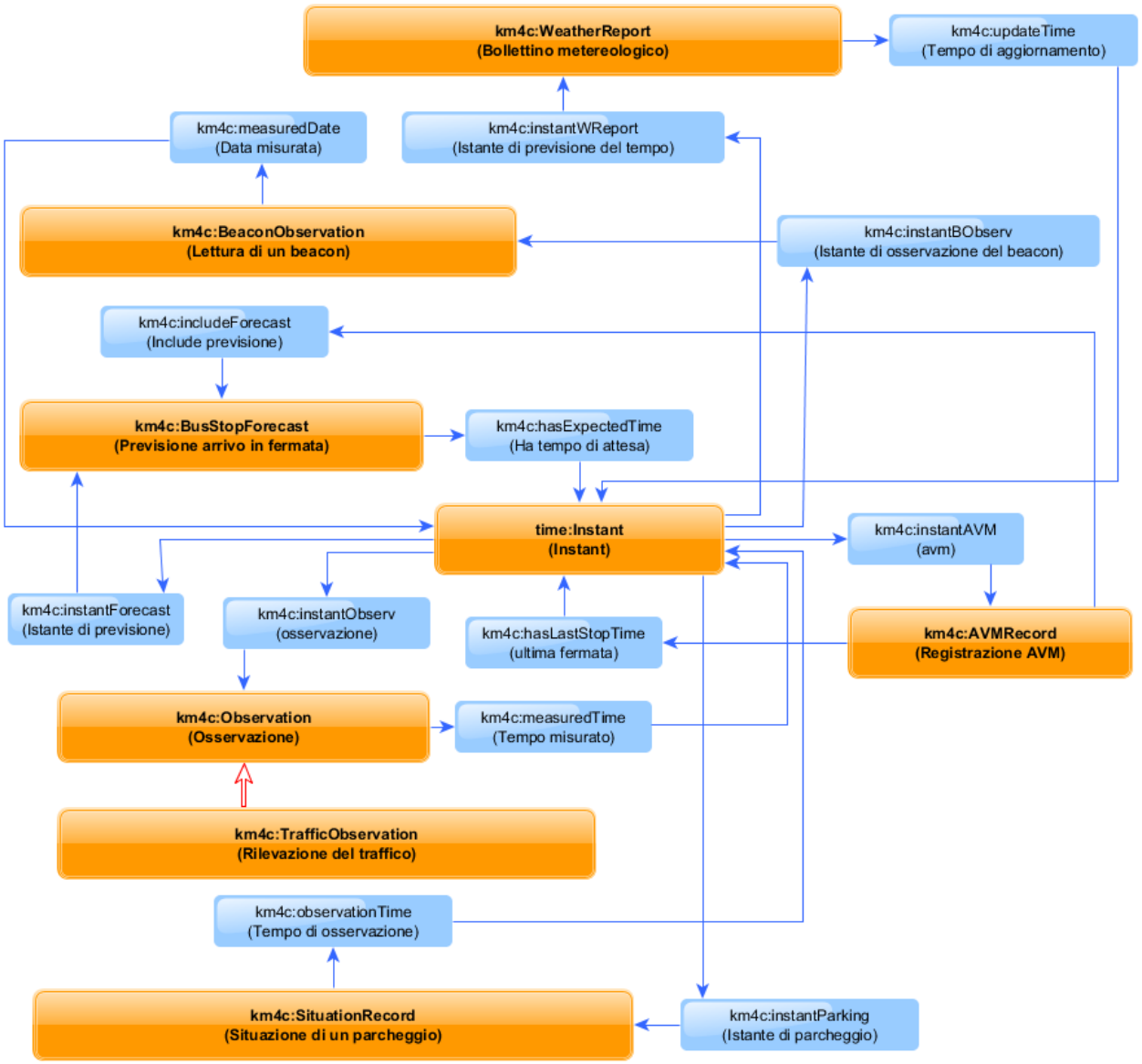



*Figura 12 Tempo*

Sono state inoltre definite delle coppie di object property per ciascuna classe che necessita di esser collegata alla classe time:Instant. Tra le classi km4c:SituationRecord e time:Instant sono state definite le object property inverse km4c:instantParking e km4c:observationTime. Tra le classi km4c:WeatherReport e km4c:Instant sono state definite le object property inverse km4c:instantWReport e km4c:updateTime. Tra km4c:Observation e time:Instant si hanno le object property inverse km4c:measuredTime e km4c:instantObserv. Tra km4c:BusStopForecast e time:Instant abbiamo km4c:hasExpectedTime e km4c:instantForecast. Tra km4c:AVMRecord e time:Instant, si hanno le object property inverse km4c:hasLastStopTime e km4c:instantAVM. Tra km4c:BeaconObservation e time:Instant infine, sono state definite le proprieta' inverse km4c:instantBObserv e km4c:measuredDate.

Il dominio di tutte le object property con nome instantXXXX è definito da elementi di classe time:temporalEntity, in modo tale da poter espandere tali proprietà non solo ad istanti, ma anche a possibili intervalli di tempo.

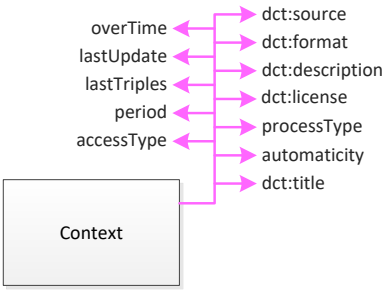



*Figura 1 Metadati*

La settima macro-classe, come già anticipato in precedenza, riguarda i metadati associati ai vari dataset. Sesame permette di definire all'interno dell'ontologia, dei Named Graph, che non sono altro che dei grafi a cui si va ad associare un nome, detto anche contesto. Il contesto è in pratica un ulteriore campo che permette di espandere il modello a triple in un modello a quadruple; Owlim ci permette in fase di caricamento triple, di associare i vari contesti a differenti set di triple. In questa macro-classe abbiamo quindi definito delle data property che ci permettano di memorizzare tutte le info utili legate ad un certo dataset, ad esempio: data di creazione, sorgente dati, formato del file originario, descrizione del dataset, licenza associata al dataset, tipo di processo di ingestion, quanto è automatizzato tutto il processo di ingestion, tipo di accesso al dataset, overtime, periodo di riferimento, parametri associati, data di aggiornamento, data di creazione triple.

La macro-classe comprende i concetti peculiari dell'Internet delle cose (IoT), riassunti in Figura 14, che sono parte dell'ontologia di Km4City dalla versione 1.6.5 in avanti. Sensori e attuatori sono modellati attraverso i concetti km4c:IoTSensor e km4c:IoTActuator rispettivamente. Sia i sensori che gli attuatori si interfacciano con l'esterno attraverso componenti hardware/software dedicati, i cosiddetti broker. Un broker generico è modellato attraverso il concetto km4c:IoTBroker. Ogni broker implementa uno o più protocolli di comunicazione, come NGSI, MQTT, AMQP e altri. Sono stati introdotti concetti specializzati per raggruppare quei broker che comunicano attraverso lo stesso protocollo, come km4c:NGSIBroker, km4c:MQTTBroker, km4c:AMQPBroker e altro. Le misurazioni che i sensori sono in grado di eseguire e i segnali di ingresso accettati dagli attuatori sono modellati attraverso il concetto km4c:DeviceAttribute. Per fornire le istanze di km4c:DeviceAttribute di una semantica, è stata introdotta la proprietà km4c:value_type. In questo modo, tutti i sensori che misurano una temperatura saranno collegati ad un'istanza di km4c:DeviceAttribute la cui proprietà km4c:value_type sarà riempita da una specializzazione di ssn:Property che rappresenta genericamente la temperatura, mentre tutti i sensori di traffico che misurano il flusso di veicoli saranno collegati ad un'istanza di km4c:DeviceAttribute la cui proprietà km4c:value_type verrà riempita con una specializzazione di ssn:Property che rappresenta genericamente un flusso di traffico. L'ontologia della rete dei sensori semantici e l'ontologia IoT-Lite sono riutilizzate per una migliore interoperabilità con altre ontologie.

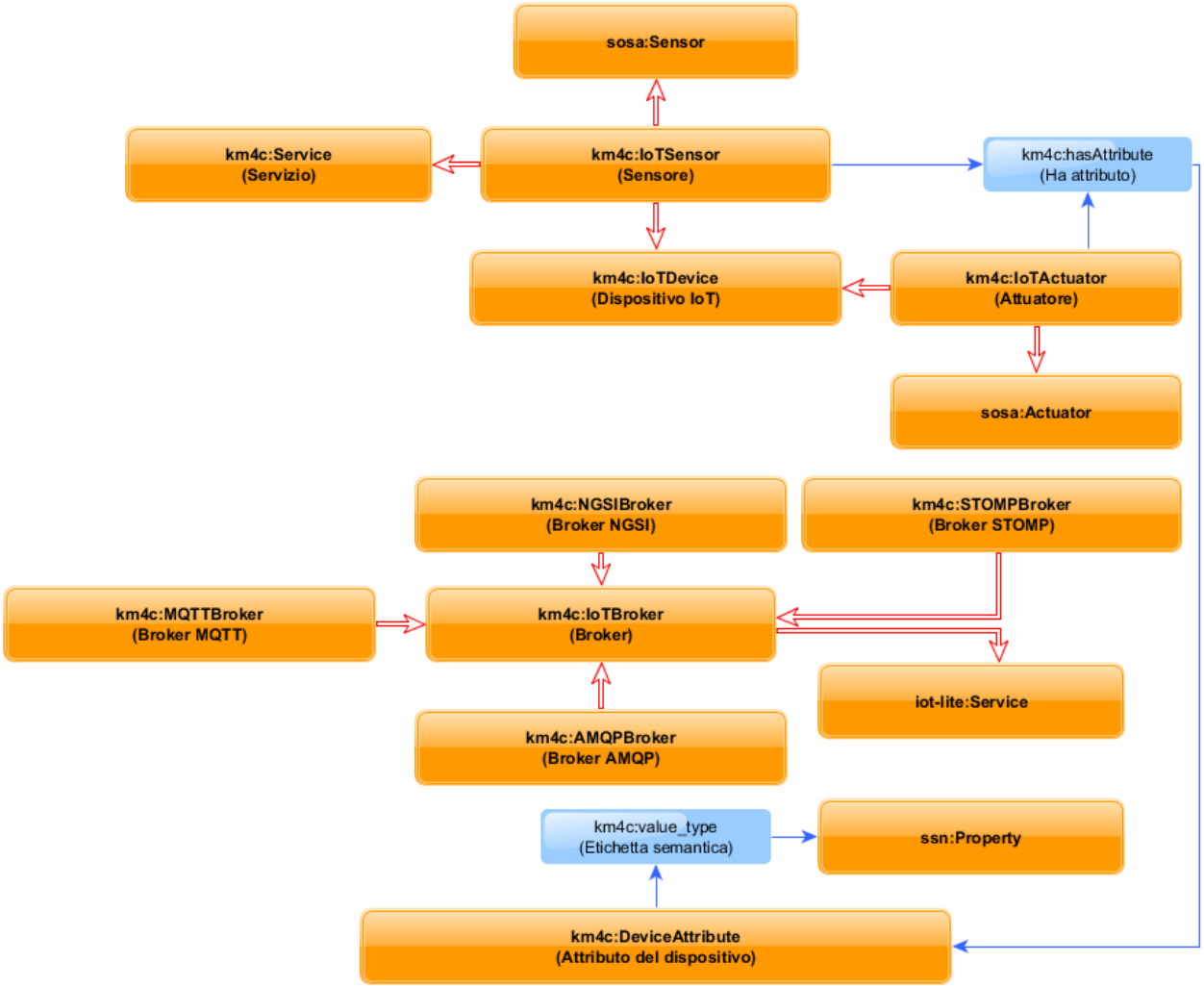



*Figura 134 Internet of Things*

Nell'agosto 2020, l'Ontologia ricomprende inoltre concetti e proprietà che la rendono adatta per la modellazione dei glussi di acqua e di alter sostanze in condutture installate su linee di produzione industriali (Fig. 14). Così facendo, è stato compiuto un primo passo nella direzione di rendere il modello dati di Km4City/Snap4City idoneo a supportare progetti nel settore della Smart Industry. In qualche maggiore dettaglio, un nuovo vocabolario è stato importato, precisamente la *SAREF extension for building devices*, disponibile all'indirizzo https://saref.etsi.org/saref4bldg/, e sono stati inoltre aggiunti all'Ontologia un insieme di concetti e proprietà basati su tale vocabolario. Si tratta in particolare di oncetti che abilitano la rappresentazione degli stabilimenti produttivi sia dal punto di vista fisico (introdotta ad esempio la classe http://www.disit.org/saref4bldg-ext/Site come suddivisione di primo livello di un edificio adibito alla produzione industriale, ed anche la classe http://www.disit.org/saref4bldg-ext/Area per la rappresentazione delle partizioni dei siti, quindi come suddivisione di secondo livello), sia anche dal punto di vista logico/funzionale, con l'introduzione delle http://www.disit.org/saref4bldg-ext/ProductionLine in cui si hanno http://www.disit.org/saref4bldg-ext/Flow di http://www.disit.org/saref4bldg-ext/Water, http://www.disit.org/saref4bldg-ext/Energy, o http://www.disit.org/saref4bldg-ext/ChemicalCompound, e che http://www.disit.org/saref4bldg-ext/startIn e http://www.disit.org/saref4bldg-ext/endIn un qualche https://saref.etsi.org/saref4bldg/PhysicalObject come una https://saref.etsi.org/saref4bldg/Pump, ad esempio la http://www.disit.org/saref4bldg-ext/pump1.

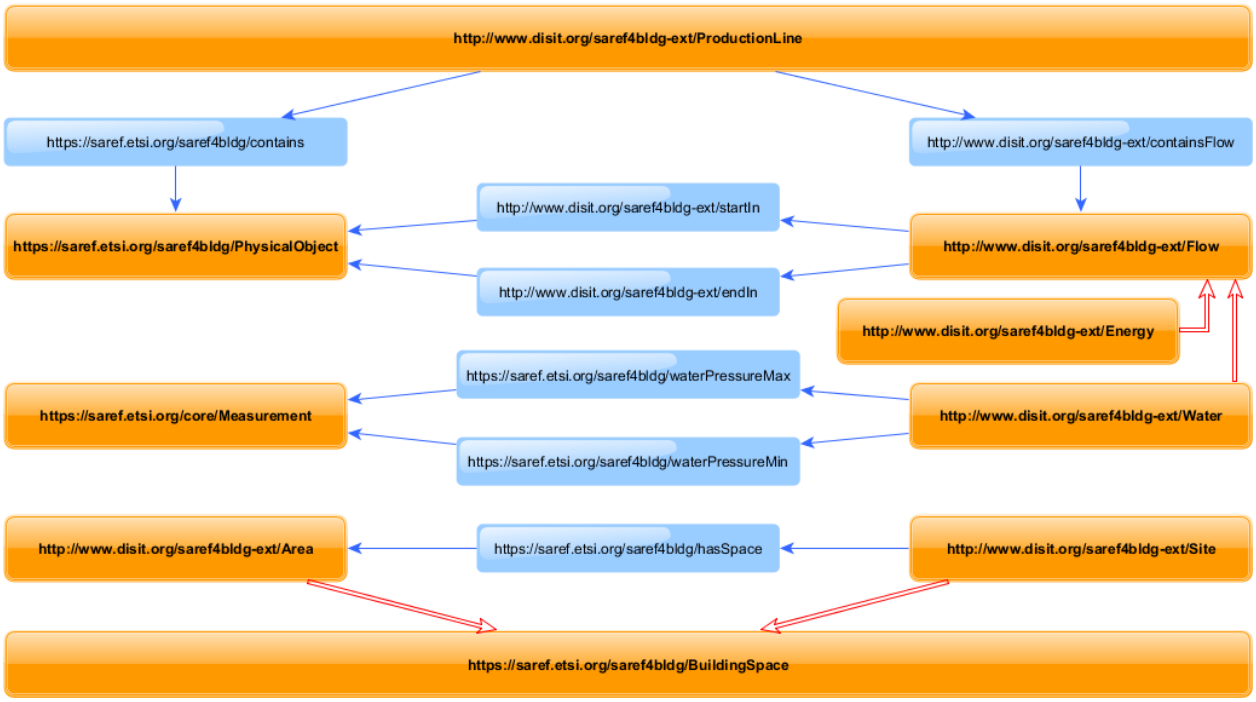



*Figura 2 Smart Industry*

## Data property delle classi principali

Le data property delle ontologie incluse sono state riutilizzate quando possibile. Quando non è stato possibile identificare una proprietà adatta da riutilizzare, una nuova proprietà ad-hoc è stata definita nell'ontologia Km4City. Analizziamo di seguito, partizionato per classe di dominio, le proprietà dei dati che sono state definite all'interno dell'ontologia Km4City.

La data property km4c:typeLabel è stata definita per tutte le classi esistenti all'interno dell'ontologia, e rappresenta il campo testo nel quale è salvato il tipo a cui appartiene l'istanza; grazie a tale proprietà è possibile attivare una ricerca full-text che comprenda anche il tipo di classe delle istanze.

All'interno della classe km4c:PA sono state definite soltanto tre data property:

- foaf:name, che rappresenta il nome della pubblica amministrazione rappresentata dall'istanza;
- l'identificatore univoco presente nel sistema regionale, contenuto in dct:identifier, ripreso dall'ontologia DublinCore;
- dct:alternative, in cui è memorizzato il codice comunale presente nel codice fiscale.

Altre informazioni relative alla PA possono essere individuate attraverso il legame con la classe km4c:Service, dove appunto sono presenti più informazioni dettagliate che altrimenti risulterebbero ridondanti.

La classe km4c:Resolution, le cui istanze come visto in precedenza sono le delibere, presenta data property per:

- la rappresentazione del codice identificativo della delibera, attraverso la proprietà dct:identifier;
- l'anno di approvazione, contenuto in km4c:year;
- l'oggetto della delibera, contenuto nella proprietà dct:subject, anch'essa riusata dalla DublinCore.

Ciascuna istanza della classe km4c:Road è univocamente individuata attraverso la data property dct:identifier dove viene memorizzato l'identificativo del toponimo per tutta la rete regionale, formato da 15 caratteri e definito secondo la seguente regola: una sigla *RT* iniziale, seguita dal codice ISTAT del comune a cui appartiene il toponimo (6 caratteri), seguito da un numero progressivo a 5 cifre, seguito dalla sigla *TO*. Si noti che per quelle istanze generate attraverso la triplificazione della Open Street Map, invece che attraverso l'importazione dei dati della Regione Toscana, l'identificativo assume una forma diversa. Esso inizia infatti per "OS", seguito dall'identificativo numerico dell'elemento Open Street Map che ha originato la produzione dell'istanza, seguito da un'etichetta di due caratteri che indica il tipo dell'istanza generata.

La km4c:roadType invece, rappresenta la tipologia di toponimo stradale, cioè ad esempio:

- Chiasso, Corso, Largo, Località, Piazza, Piazzale, Piazzetta, Via, Viale, Vicolo, Viottolo, Viuzzo etc.

All'interno del grafo strade sono inoltre presenti due nomi per ciascun toponimo: il nome ed il nome esteso, cioè comprensivo anche del tipo di toponimo. Il nome senza tipo, una stringa di testo, è contenuto nella data property km4c:roadName, mentre per il nome esteso, è stata definita un'altra data property, cioè km4c:extendexName. Inoltre, per tutti gli eventuali alias che si possono formare (come ad esempio Via S. Marta per Via Santa Marta) è utilizzata la proprietà dct:alternative, facendo in modo che possano essere più di uno i vari nomi alternativi, così da facilitare le successive riconciliazioni.

Per quanto riguarda la classe km4c:AdministrativeRoad, oltre alle data property dct:alternative e km4c:adRoadName che rispettivamente contengono i possibili nomi alternativi della strada e il suo nome ufficiale, è stata definita la data property km4c:amminClass rappresenta la classificazione amministrativa, cioè se si tratta di:

- strada statale;
- strada regionale;
- strada provinciale;
- strada comunale;
- strada militare;
- strada privata.

Infine, l'identificativo univoco della strada è contenuto all'interno del campo dct:identifier, il quale segue la seguente regola, definita a livello regionale: lunghezza complessiva 15 caratteri, inizia con i caratteri *RT*, seguiti dal codice ISTAT del comune a cui appartiene l'estesa amministrativa (6 caratteri), seguiti da un numero progressivo a 5 cifre, e infine dalla sigla PA. Si noti che per quelle istanze generate attraverso la triplificazione della Open Street Map, invece che attraverso l'importazione dei dati della Regione Toscana, l'identificativo assume una forma diversa. Esso inizia infatti per "OS", seguito dall'identificativo numerico dell'elemento Open Street Map che ha originato la produzione dell'istanza, seguito da un'etichetta di due caratteri che indica il tipo dell'istanza generata.

Le istanze della classe km4c:RoadElement sono univocamente individuate dalla data property dct:identifier, campo di 15 caratteri formato come segue: i caratteri *RT*, seguiti da 6 caratteri per il codice ISTAT del comune di appartenenza, seguiti da un numero progressivo a 5 cifre, ed infine i caratteri *ES*. Si noti che per

quelle istanze generate attraverso la triplificazione della Open Street Map, invece che attraverso l'importazione dei dati della Regione Toscana, l'identificativo assume una forma diversa. Esso inizia infatti per "OS", seguito dall'identificativo numerico dell'elemento Open Street Map che ha originato la produzione dell'istanza, seguito da un'etichetta di due caratteri che indica il tipo dell'istanza generata.

Anche la data property km4c:elementType è stata definita per la classe km4c:RoadElement, e può assumere i seguenti valori:

- di tronco carreggiata;
- di area a traffico strutturato;
- di casello/barriera autostradale;
- di passaggio a livello;
- di piazza;
- di rotatoria;
- di incrocio;
- di parcheggio strutturato;
- di area a traffico non strutturato;
- di parcheggio;
- in area di pertinenza;
- pedonale;
- raccordo, bretella, svincolo;
- controviale;
- traghetto (elemento fittizio).

Nel grafo strade della Regione Toscana, associato a questa classe troviamo anche la sua classificazione funzionale, definita all'interno dell'ontologia come data property km4c:elementClass, i cui possibili valori sono:

- autostrada;
- extraurbana principale;
- extraurbana secondaria;
- urbana di scorrimento;
- urbana di quartiere;
- locale/vicinale/privata ad uso privato.

La data property km4c:composition invece, è stata definita per indicare la composizione della strada a cui appartiene l'elemento stradale e per la precisione, sono possibili i valori "carreggiata unica" oppure "carreggiate separate". La proprietà km4c:elemLocation rappresenta invece la sede dell'elemento, ed i valori che può assumere sono:

- a raso;
- ponte;
- rampa;
- galleria;
- ponte e galleria;
- ponte e rampa;
- galleria e rampa;

- galleria, ponte e rampa.

Per quanto riguarda la classe di larghezza dell'elemento stradale, si fa riferimento alla data property km4c:width, che permette di individuare la fascia di appartenenza: "minore di 3,5 mt", "tra 3,5 e 7,0 mt", "maggiore di 7,0 mt" oppure "non rilevato". Per la lunghezza invece la data property di riferimento è km4c:length, valore liberamente inseribile che non fa riferimento a nessuna fascia. Altro dato disponibile sul grafo strade, e di fondamentale importanza per la definizione delle manovre consentite, è la direzione di percorrenza dell'elemento stradale, indicato dalla data property km4c:trafficDir che può assumere uno dei seguenti 4 valori:

- tratto stradale aperto in entrambe le direzioni (default);
- tratto stradale aperto nella direzione positiva (da nodo iniziale a nodo finale);
- tratto stradale chiuso in entrambe le direzioni;
- tratto stradale aperto nella direzione negativa (da nodo finale a nodo iniziale).

La data property km4c:operatingStatus invece serve per tener traccia dello stato d'esercizio dei differenti elementi stradali e può quindi assumere i valori "in esercizio", "in costruzione" oppure "in disuso". Infine è presente anche una data property che tiene conto dei limiti di velocità su ogni elemento stradale, cioè km4c:speedLimit.

Nella classe StatisticalData sono state definite data property che permettano di associare al valore effettivo (memorizzato in km4c:value), altre informazioni necessarie a mantenere intatto il suo significato, come ad esempio le proprietà dct:identifier, dct:description, dct:created e dct:subject.

Un nodo o giunzione è un punto di intersezione degli assi di due elementi stradali, ed è sempre un'entità puntuale, rappresentata in termini geometrici, da una coppia di coordinate. Le istanze della classe km4c:Node possono essere univocamente individuate grazie al dct:identifier, costituito anch'esso, come i precedenti codici, da 15 caratteri, secondo le seguenti regole: i primi due caratteri *RT*, seguiti dai 6 caratteri di codice ISTAT del comune in cui è localizzato il nodo, seguiti da un numero progressivo a 5 cifre, ed infine *GZ*. Si noti che per quelle istanze generate attraverso la triplificazione della Open Street Map, invece che attraverso l'importazione dei dati della Regione Toscana, l'identificativo assume una forma diversa. Esso inizia infatti per "OS", seguito dall'identificativo numerico dell'elemento Open Street Map che ha originato la produzione dell'istanza, seguito da un'etichetta di due caratteri che indica il tipo dell'istanza generata.

Ciascun nodo è inoltre caratterizzato da un tipo, rappresentato dalla data property km4c:nodeType, che può assumere i seguenti valori:

- intersezione a raso/biforcazione;
- casello autostradale;
- minirotatoria (raggio di curvatura < 10m);
- cambio sede;
- terminale (inizio o fine elemento stradale);
- cambio toponimo/titolarità/gestore;
- variazione classe di larghezza;
- area di traffico non strutturato;
- passaggio a livello;
- nodo di supporto (per definire i loop);

- variazione classificazione tecnico – funzionale;
- variazione stato di esercizio;
- variazione composizione;
- nodo intermodale per ferrovia;
- nodo intermodale per aeroporto;
- nodo intermodale per porto;
- limite di regione;
- nodo fittizio.

In questo caso saranno presenti anche le data property per la localizzazione, cioè geo:lat e geo:long.

Le regole di accesso sono descritte attraverso le istanze della classe km4c:EntryRule, identificabili univocamente attraverso il dct:identifier di 15 caratteri così formato: i caratteri *RT*, seguiti da 6 caratteri che rappresentano il codice ISTAT del comune, seguiti da un numero progressivo a 5 cifre, e infine i caratteri *PL*. Si noti che per quelle istanze generate attraverso la triplificazione della Open Street Map, invece che attraverso l'importazione dei dati della Regione Toscana, l'identificativo assume una forma diversa. Esso inizia infatti per "OS", seguito dall'identificativo numerico dell'elemento Open Street Map che ha originato la produzione dell'istanza, seguito da un'etichetta di due caratteri che indica il tipo dell'istanza generata.

Le regole di accesso sono poi caratterizzate da un tipo, rappresentato dalla data property km4c:restrictionType, che può assumere i seguenti valori:

- Blank (Solo in caso di manovre);
- Direzione flusso di traffico;
- Passaggio bloccato;
- Restrizioni speciali;
- In costruzione;
- Informazioni relative ai pedaggi;
- Biforcazione;
- Manovre proibite;
- Restrizioni su veicoli.

Oltre al tipo, le regole di accesso hanno anche una descrizione, detta anche valore della restrizione e rappresentata dalla data property km4c:restrictionValue, che può assumere differenti range di valori, in base al tipo di restrizione con la quale abbiamo a che fare:

- Valori possibili per *Blank*:
  - Default Value = "-1";
- Valori possibili per *Direzione flusso di traffico* e per *Restrizioni su veicoli*:
  - Chiusa in direzione positiva;
  - Chiusa in direzione negativa;
  - Chiusa in entrambe le direzioni;
- Valori per possibili per *Passaggio bloccato*:
  - Accessibile solo per veicoli di emergenza;
  - Accessibile con chiave;
  - Accessibile con guardiano;

- Valori possibili per *Restrizioni speciali*:
    - Nessuna restrizione(Default);
    - Restrizione generica;
    - Solo Residenti;
    - Solo impiegati;
    - Solo per personale autorizzato;
    - Solo per il personale;
- Valori possibili per *In costruzione*:
    - In costruzione in entrambe le direzioni;
    - In costruzione in direzione di Marcia della corsia;
    - In costruzione in direzione opposta alla direzione di Marcia della corsia;
- Valori possibili per *Informazioni relative ai pedaggi*:
    - Strada a pedaggio in entrambe le direzioni;
    - Strada a pedaggio in direzione negativa;
    - Strada a pedaggio in direzione positiva;
- Valori per possibili per *Biforcazione*:
    - biforcazione multi corsia;
    - biforcazione semplice;
    - biforcazione di uscita;
- Valori possibili per *Manovre proibite*:
    - Manovra proibita;
    - Svolta Implicita.

La classe km4c:Maneuver presenta una sostanziale differenza dalle altre viste finora: ciascuna manovra è infatti identificata univocamente da un ID composto da 17 cifre numeriche. Una manovra è descritta dalla successione degli elementi stradali che interessa, che variano da due a tre, e dal nodo che interessa, quindi sarà quasi completamente descritta attraverso le object property che rappresentano questa situazione. All'interno del Grafo Strade, troviamo dati relativi al tipo di manovra, tipo di biforcazione e tipo di manovra proibita, ma visto che gli ultimi due tipi sono quasi sempre "non definiti", è stata associata una data property solo al tipo di manovra, cioè km4c:maneuverType, che può assumere i seguenti valori:

- Biforcazione;
- manovra proibita calcolata (Calculated Maneuver);
- manovra obbligatoria (Restricted Maneuver);
- manovra proibita (Prohibited Maneuver);
- manovra prioritaria (Priority Maneuver) .

La classe km4c:StreetNumber, presenta anch'essa un codice, dct:identifier, per individuare univocamente le istanze di questa classe, stesso formato dei precedenti: i caratteri *RT*, seguiti da 6 caratteri per il codice ISTAT del comune, poi un numero progressivo a 5 cifre, e infine i caratteri *CV*. Si noti che per quelle istanze generate attraverso la triplificazione della Open Street Map, invece che attraverso l'importazione dei dati della Regione Toscana, l'identificativo assume una forma diversa. Esso inizia infatti per "OS", seguito dall'identificativo numerico dell'elemento Open Street Map che ha originato la produzione dell'istanza, seguito da un'etichetta di due caratteri che indica il tipo dell'istanza generata.

A Firenze sono presenti due numerazioni, l'attuale in colore nero, e l'originaria, di colore rosso; è stata quindi inserita una data property, km4c:classCode, che tiene appunto conto del colore del civico e può

assumere i seguenti valori: rosso, nero, nessun colore. Ciascun numero civico può inoltre essere formato, oltre che dalla parte numerica sempre presente, da una parte letterale, rappresentate rispettivamente dalle data property km4c:number ed km4c:exponent. È stato inoltre inserito un ulteriore valore, km4c:extendNumber, all'interno del quale sarà memorizzato l'insieme del numero e dell'esponente, in modo da garantire una maggior compatibilità con i differenti formati in cui potrebbero essere scritte/ricercate le istanze di questa classe.

La classe km4c:Milestone, come visto in precedenza, identifica il valore della chilometrica progressiva del tracciato, rispetto al punto d'inizio. Anche questa classe prevede la presenza di un codice univoco formato da 15 caratteri, contenuto nella proprietà dct:identifier, e così composto: la sigla *RT*, seguita da 6 caratteri che sono il codice ISTAT del comune, seguiti da un numero progressivo a 5 cifre, ed infine dai caratteri *CC*. Si noti che per quelle istanze generate attraverso la triplificazione della Open Street Map, invece che attraverso l'importazione dei dati della Regione Toscana, l'identificativo assume una forma diversa. Esso inizia infatti per "OS", seguito dall'identificativo numerico dell'elemento Open Street Map che ha originato la produzione dell'istanza, seguito da un'etichetta di due caratteri che indica il tipo dell'istanza generata.

Su ciascun cippo chilometrico, solitamente viene scritto il valore che corrisponde a quel punto e il contenuto di questa scritta è memorizzato attraverso la data property km4c:text. Grazie alle informazioni contenute nel grafo strade, risulta banale recuperare il nome della strada, autostrada etc. dove è posizionato il cippo. Potrebbe quindi essere definita un'ulteriore object property che colleghi la classe km4c:Milestone alla classe km4c:Road; questa informazione è tutt'ora recuperabile con un passaggio in più attraverso la classe km4c:RoadElement, ma può comunque essere inserita con facilità se lo si ritenesse opportuno in futuro. Anche in questo caso sono presenti le data property per la localizzazione, cioè geo:lat e geo:long.

L'accesso o km4c:Entry è l'elemento puntuale che identifica sul territorio l'accesso esterno, diretto o indiretto, principale o secondario ad uno specifico luogo di residenza/attività; l'accesso in pratica materializza la "targhetta" del numero civico. Come detto in precedenza, ogni accesso è connesso logicamente ad almeno un numero civico. Ciascun istanza della classe km4c:Entry è identificata univocamente dalla data property dct:identifier, formato da 15 caratteri, come tutti gli altri codici visti: la sigla *RT*, seguita da 6 caratteri di codice ISTAT del comune, poi un numero progressivo a 5 cifre, ed infine i caratteri *AC*. Si noti che per quelle istanze generate attraverso la triplificazione della Open Street Map, invece che attraverso l'importazione dei dati della Regione Toscana, l'identificativo assume una forma diversa. Esso inizia infatti per "OS", seguito dall'identificativo numerico dell'elemento Open Street Map che ha originato la produzione dell'istanza, seguito da un'etichetta di due caratteri che indica il tipo dell'istanza generata.

Esistono solo tre tipi di accessi, il cui valore è memorizzato nella data property km4c:entryType:

- accesso esterno diretto;
- accesso esterno indiretto;
- accesso interno.

Oltre al tipo di accesso, per questa classe può essere utile conoscere se è presente un passo carrabile o no, attraverso la proprietà km4c:porteCochere.

Sono inoltre presenti anche in questo caso le data property per la localizzazione, cioè geo:lat e geo:long.

Venendo ai miglioramenti più recenti nella modellazione del grafo strade e più specificatamente alla modellazione delle corsie, l'unica proprietà della classe km4c:Lanes è km4c:direction, che può essere trovata solo in quei casi in cui due direzioni del traffico possano essere trovate nella strada o segmento di strada, per identificare la direzione del traffico in cui le corsie sono attraversate. Infatti, se una strada ha due direzioni di traffico, è collegata a due istanze di km4c:Lanes, una per ciascuna delle due direzioni del traffico. Invece, le istanze del concetto km4c:LanesCount di solito hanno la proprietà km4c:undesignated che viene riempita da un numero intero che indica quante delle corsie non sono riservate a specifiche tipologie di veicoli, e hanno anche altre proprietà di dati come km4c:bus, km4c:motor_vehicle, km4c:psv i cui nomi derivano dalle denominazioni delle tipologie di veicoli che sono definite nella Open Street Map, ed i cui valori sono numeri interi che indicano quante delle corsie sono riservate alle diverse tipologie di veicoli. Va ricordato che ogni istanza di km4c:LanesCount è connessa ad un'istanza di km4c:Lanes che rappresenta un insieme di corsie. Infine, il concetto km4c:Lane ha una proprietà dati km4c:position che indica la posizione della corsia nell'insieme di corsie di cui fa parte. Le istanze di km4c:Lane possono anche avere una proprietà km4c:turn, attraverso la quale è indicata la manovra obbligatoria che i conducenti devono eseguire una volta raggiunta la fine della corsia.

Invece, per quanto riguarda le restrizioni del traffico, il concetto km4c:TurnRestriction ha due proprietà di dati:

- km4c:restriction, che indica se la manovra è obbligatoria o proibita;
- km4c:except, che può essere trovata solo in quei casi in cui la restrizione non si applica a una o più categorie di veicoli.

La classe km4c:AccessRestriction ha invece quattro proprietà dei dati:

- km4c:who, dove sono indicate le categorie di veicoli interessate dalla restrizione;
- km4c:direction, dove è indicata la direzione del traffico a cui si applica la restrizione;
- km4c:access, dove è indicato se i veicoli specificati hanno un accesso riservato, o se al contrario non possono accedervi;
- km4c:condition, che può essere trovata solo in quei casi in cui la restrizione si applica ad un insieme di condizioni diverse dalla tipologia dei veicoli e dalla direzione del traffico. Tra parentesi, questa data property può essere trovata impostata per tutte le istanze di km4c:Restriction, se è necessario rappresentare alcune condizioni diverse da quelle modellate mediante proprietà dedicate.

Infine, km4c:MaxMinRestriction ha le seguenti proprietà dei dati:

- km4c:what, indica a cosa si riferisce la restrizione;
- km4c:limit, che indica la limitazione stessa;
- km4c:direction, che indica la direzione del traffico in quei casi in cui solo una delle due direzioni del traffico è influenzata dalla restrizione.

La classe km4c:Service è stata corredata di una contact card, cioè di una serie di data property prese in prestito dall'ontologia schema.org (http://schema.org) per rendere più standardizzata la descrizione delle varie aziende: schema:name, schema:telephone, schema:email, schema:faxNumber, schema:url, schema:streetAddress, schema:addressLocality, schema:postalCode, schema:addressRegion alle quali abbiamo aggiunto skos:note per eventuali aggiunte come gli orari di apertura che a volte sono presenti nei dati. Oltre a queste, sono state definite altre data property come km4c:houseNumber, per isolare il numero civico dall'indirizzo e, quando possibile, la geo:lat e geo:long per la localizzazione.

Con l'introduzione delle Digital Location la classe km4c:Service è stata corredata delle seguenti proprietà:

- km4c:hasGeometry: proprietà definita per memorizzare l'insieme di coordinate associate alla Digital Location, che può essere un POINT, una LINESTRING o un POLYGON;
- km4c:routeLength: per memorizzare la lunghezza dei percorsi (ad esempio di jogging);
- km4c:stopNumber: proprietà che indica il numero di fermata all'interno dell'intero percorso della linea del trasporto pubblico locale;
- km4c:lineNumber: proprietà che indica la linea di appartenenza di una fermata;
- km4c:managingBy: proprietà che indica il gestore della Digital Location;
- dct:description: proprietà in cui è riportata la descrizione della Digital Location;
- km4c:owner: proprietà che indica il proprietario della Digital Location;
- km4c:abbreviation: proprietà che riporta eventuali abbreviazioni di nome della Digital Location;
- km4c:type: proprietà che indica la categoria a cui appartiene il gestore di una Digital Location, oppure il tipo di ZTL o di museo o dell'area di riferimento;
- km4c:time: proprietà che indica l'orario di apertura di un esercizio;
- km4c:firenzeCard: proprietà che indica se la Digital Location è affiliata con la FirenzeCard;
- km4c:multimediaResouce: proprietà per associare immagini ed mp3 alle singole Digital Location;
- km4c:districtCode: proprietà che specifica il codice del quartiere in cui si trova la Digital Location;
- km4c:routePosition: proprietà che indica la posizione della Digital Location all'interno di un percorso tematico;
- km4c:areaCode: proprietà che indica un codice comunale per l'area di riferimento;
- km4c:routeCode: proprietà che specifica il codice del percorso tematico a cui fa riferimento la Digital Location.
- schema:price: proprietà che indica l'eventuale costo di ingresso.

La classe km4c:Event invece possiede tutte le proprietà appartenenti alla schema:Event e altre proprietà aggiunte in base alle informazioni fornite dal Comune di Firenze. Per comodità di seguito tutte queste proprietà saranno riportate in forma di elenco.

- schema:startDate, proprietà che indica la data di inizio dell'evento;
- schema:endDate, proprietà che indica la data di fine dell'evento;
- schema:description, contiene la descrizione dell'evento;
- schema:image, contiene, se presente, url di un immagine di riferimento per l'evento;
- schema:name, è il nome dell'evento;
- schema:url, sito web dell'evento;
- schema:telephone, numero di telefono di riferimento per l'evento;
- schema:streetAddress, indirizzo del luogo in cui si terrà l'evento;
- schema:addressLocality, città in cui si terrà l'evento;
- schema:addressregion, provincia in cui si terrà l'evento;
- schema:postalCode, CAP della città in cui si terrà l'evento;
- km4c:houseNumber, numero civico del luogo in cui si terrà l'evento;
- skos:note, proprietà che conterrà eventuali note o altre informazioni presenti per l'evento;
- schema:price, proprietà che indica l'eventuale costo di ingresso;
- dct:identifier, contiene l'identificativo dell'evento assegnatogli dal Comune di Firenze;
- km4c:eventCategory, proprietà per indicare il tipo di evento;

- km4c:placeName, contiene il nome del luogo in cui si svolgerà l'evento;
- km4c:freeEvent, variabile float che indica se l'evento è gratuito o meno;
- km4c:eventTime, proprietà che contiene l'orario di inizio dell'evento.

Oltre a queste proprietà, un evento è corredato anche di una coppia di coordinate che lo geolocalizzano, cioè geo:lat e geo:long.

La classe gis:Geometry presenta invece soltanto la proprietà gis:asWKT, la quale permette di definire una unica stringa rappresentante l'insieme di coordinate che definiscono la forma geometrica dell'oggetto, cioè ad esempio:

LINESTRING(11.21503989 43.77879298, 11.21403307 43.77936126, 11.21385829 43.77947115)

La classe km4c:CarParkSensor presenta due data property:

- il dct:identifier, definito a livello regionale sempre attraverso un codice di 15 caratteri iniziante per *RT* e terminante con la sigla della provincia di appartenenza. Si noti che per quelle istanze generate attraverso la triplificazione della Open Street Map, invece che attraverso l'importazione dei dati della Regione Toscana, l'identificativo assume una forma diversa. Esso inizia infatti per "OS", seguito dall'identificativo numerico dell'elemento Open Street Map che ha originato la produzione dell'istanza, seguito da un'etichetta di due caratteri che indica il tipo dell'istanza generata;
- km4c:capacity, cioè la capacità in numero di posti del parcheggio.

Alla classe km4c:CarParkSensor è inoltre connessa la classe km4c:SituationRecord dove sono state definite le data property:

- km4c:fillrate, numero dei veicoli in entrata;
- km4c:exitrate, numero di veicoli in uscita;
- km4c:carParkStatus, stringa che descrive l'attuale stato del parcheggio (possibili valori sono "enoughSpacesAvailable", "carParkFull", "noParkingInformationAvailable");
- ID univoco, in dct:identifier;
- stato di validità del record (data property km4c:validityStatus, che può essere solo "active" nel caso di parcheggi);
- km4c:parkOccupancy, posti occupati in percentuale;
- km4c:occupied, numero di posti occupati;
- km4c:free, numero di posti liberi.

La classe km4c:WeatherReport è caratterizzata dalle seguenti data property:

- dct:identifier, contenente l'ID univoco che permette di identificare i differenti report;
- km4c:timestamp che indica il tempo in cui è stato creato il report espresso in millisecondi;
- fase lunare (km4c:lunarPhase);
- orario di tramonto/sorgere del sole e della luna, cioè km4c:sunrise, km4c:sunset, km4c:moonrise e km4c:moonset;
- km4c:heightHour e km4c:sunHeight che rappresentano rispettivamente l'orario in cui il sole raggiunge la massima altezza e tale altezza;

Ciascuna istanza di WeatherPrediction è caratterizzata dalle data property:

- km4c:day, che rappresenta il giorno a cui fa riferimento la previsione (che insieme all'id del report, costituisce un modo univoco per identificare le singole previsioni);
- i valori di temperatura minimi e massimi o reali e percepiti (rispettivamente rappresentati dalle data property km4c:minTemp, km4c:maxTemp, km4c:recTemp, km4c:perTemp);
- la direzione del vento, km4c:wind;
- km4c:humidity cioè la percentuale di umidità;
- la quota a cui è presente la neve km4c:snow;
- l'orario, indicato con km4c:hour;
- l'indice ultravioletto della giornata, indicato nella proprietà km4c:uv.

Le classi km4c:SensorSiteTable e km4c:SensorSite hanno soltanto una data property ciascuna, l'identificativo univoco, contenuto nella proprietà dct:identifier.

La classe km4c:Observation è invece completata dalle data property:

- dct:identifier;
- dct:date;
- km4c:averageDistance, distanza media tra due auto;
- km4c:averageTime, tempo medio tra due auto;
- km4c:occupancy, percentuale di occupazione della strada;
- km4c:concentration, concentrazione di auto;
- km4c:vehicleFlow, flusso di veicoli rilevato dai sensori;
- km4c:averageSpeed, velocità media;
- percentile calcolato sulle velocità di transito, contenuto nelle proprietà km4c:thresholdPerc e km4c:speedPercentile.

La classe km4c:PublicTransportLine e la classe km4c:Lot presentano entrambe come data property la dct:identifier e la dct:description dalla ontologia DublinCore e rappresentano rispettivamente il numero della linea/del lotto e la descrizione del percorso che effettua o del lotto.

La classe km4c:Route invece oltre alle data property dct:identifier, foaf:name e dct:description, presenta:

- la proprietà km4c:routeLenght, cioè la lunghezza del percorso espresso in metri;
- la direzione del percorso, contenuta nella proprietà km4c:direction.

La classe km4c:BusStop presenta le data property:

- dct:identifier;
- foaf:name per il nome della fermata;
- geo:lat;
- geo:long.

Queste ultime due data property sono anche le uniche presenti nella classe km4c:RouteJunction, oltre al dct:identifier.

La classe km4c:BusStopForecast contiene soltanto le data property per il tempo di arrivo previsto e per l'identificativo, rispettivamente km4c:expectedTime e dct:identifier.

La classe km4c:AVMRecord necessita invece delle data property per:

- identificare il mezzo (km4c:vehicle);
- il tempo di arrivo all'ultima fermata (km4c:lastStopTime);
- lo stato della corsa, cioè se è in anticipo o in ritardo o puntuale (km4c:rideState);
- l'ente gestore e l'ente proprietario del sistema AVM (km4c:managingBy e km4c:owner);
- l'identificativo univoco del record (dct:identifier);
- le coordinate geo:lat e geo:long, che indicato la posizione del mezzo al momento dell'invio del report.

Infine la classe km4c:Ride presenta solo la data property dct:identifier, come la classe km4c:RouteLink.

La classe RouteSection, invece presenta oltre all'ID anche la data property km4c:distance, dove viene salvata la distanza tra due fermate successive all'interno di un percorso.

Analizziamo adesso le data property relative alle classi km4c:Beacon e km4c:BeaconObservation.

La classe km4c:Beacon, oltre alla data property dct:identifier, presenta le proprietà:

- km4c:owner, nella quale è memorizzato il proprietario del singolo beacon;
- schema:name che contiene l'eventuale nome associato ad un beacon;
- km4c:uuid, km4c:minor, km4c:major, che identificano il beacon;
- km4c:public, che definisce se il beacon è pubblico oppure no;
- per la memorizzazione della posizione, cioè geo:lat e geo:long.

La seconda classe, cioè km4c:BeaconObservation, presenta invece, oltre al dct:identifier che rende univoca ciascuna lettura, anche le proprietà:

- geo:lat e geo:long, che indicano la posizione del soggetto con cui è stata stabilita una connessione;
- dct:date, che memorizza giorno e ora della osservazione;
- km4c:power, che indica la potenza con cui ha rilevato la connessione con un utente, che permette appunto di stabilirne la distanza.

Sono state inoltre definite delle proprietà da associare al Context, relative ad informazioni provenienti dalle tabelle che descrivono i singoli processi ETL che elaborano i differenti dati. Per ciascun processo infatti, il cui nome è memorizzato nel campo dct:title, viene definito:

- un tipo di sorgente, memorizzato all'interno del campo dct:source;
- il formato originale dei dati (csv, dbf, etc.) memorizzato nel campo dct:format;
- la descrizione del dataset dct:description;
- la licenza associata al dataset, salvata nella proprietà dct:license;
- il tipo di processo a cui si fa riferimento, salvato nella data property km4c:processType;
- la data property km4c:automaticity, che dice se il processo è automatizzato oppure no (ad esempio, dataset come il grafo stradale non possono essere completamente automatizzati a causa del processo di ottenimento dati che necessita di inviare e ricevere e-mail);
- km4c:accessType, che informa su come sono recuperati i dati (chiamate HTTP, Rest, ecc.);
- km4c:period, che contiene il periodo di tempo (in secondi) che intercorre tra due chiamate di uno stesso processo;
- km4c:overTime, che indica dopo quanto tempo un processo deve essere terminato;
- km4c:lastUpdate, rappresenta la data di aggiornamento dei dati;

- km4c:lastTriples, dove è conservata la data di generazione delle triple.

La classe km4c:RailwayLine ha soltanto tre data property:

- dct:identifier, che contiene l'identificativo della linea ferroviaria, che nel caso delle linee importate dagli Open Data della Regione Toscana è un codice di 12 caratteri iniziante con le lettere *RT*, seguite da 3 caratteri che identificano la regione (T09 per la Toscana), quindi da un numero progressivo a 5 cifre, ed infine dalla sigla *PF*;
- foaf:name, in cui è salvata la denominazione convenzionale della linea;
- dct:alternative, in cui è memorizzata invece la denominazione ufficiale se presente.

La classe km4c:RailwayDirection invece ha soltanto le prime due data property specificate per km4c:RailwayLine, con lo stesso utilizzo:

- dct:identifier, in cui viene memorizzato il codice identificativo dell'istanza, che nel caso degli Open Data importati dalla Regione Toscana è formato da 12 caratteri inizianti con le lettere *RT*, seguite da 3 caratteri che identificano la regione (T09 per la Toscana), poi da un numero progressivo a 5 cifre, e infine dalle lettere *ED*;
- foaf:name, in cui viene memorizzata la denominazione convenzionale.

La classe km4c:RailwayElement ha un identificativo univoco conservato nella proprietà dct:identifier, formato da 12 caratteri che seguono le seguenti regole: i caratteri *RT,* seguiti da 3 caratteri di codice regionale (T09 per la Toscana), seguito da un numero progressivo a 5 cifre, seguito dalla sigla *EF*. Si noti che per quelle istanze generate attraverso la triplificazione della Open Street Map, invece che attraverso l'importazione dei dati della Regione Toscana, l'identificativo assume una forma diversa. Esso inizia infatti per "OS", seguito dall'identificativo numerico dell'elemento Open Street Map che ha originato la produzione dell'istanza, seguito da un'etichetta di due caratteri che indica il tipo dell'istanza generata.

Per ogni km4c:RailwayElement, sono inoltre definite le seguenti proprietà:

- km4c:elementType, che può assumere i seguenti tre valori:
  - ferrovia ordinaria;
  - ferrovia AC/AV;
  - altro;
- km4c:operatingStatus, che può assumere solo i valori:
  - ferrovia in costruzione;
  - ferrovia in esercizio;
  - ferrovia dismessa;
- km4c:elemLocation che indica la sede dell'elemento ferroviario, e può assumere i valori:
  - a raso;
  - su ponte/viadotto;
  - in galleria;
- km4c:supply, che specifica se si tratta di una "linea elettrificata" oppure di una "linea non elettrificata";
- km4c:gauge, cioè il tipo di scartamento, che può assumere soltanto due valori, "ridotto" o "standard";
- km4c:underpass, che può assumere i seguenti valori:
  - l'elemento non è in sottopasso di nessun altro oggetto;

- l'elemento è in sottopasso di un altro oggetto;
- l'elemento è contemporaneamente in sovrappasso e sottopasso di altri oggetti.

Per la modellazione dei sottopassaggi, l'ontologia include anche il concetto km4c:Underpass.

Altre data property che sono state definite sempre per la classe km4c:RailwayElement sono:

- km4c:lenght, lunghezza in metri dell'elemento;
- km4c:numTrack, numero di binari, zero nel caso di linee in costruzione o dismesse;
- km4c:trackType, che specifica se l'elemento è composto da "binario singolo" o "binario doppio".

Per la classe km4c:RailwaySection, è stata inoltre definita la data property km4c:axialMass, cioè la classificazione della linea rispetto alla massa assiale, la quale può assumere i seguenti valori:

- D4 - corrispondente a massa per asse pari a 22.5 t;
- C3 - corrispondente a massa per asse pari a 20.0 t;
- B2 - corrispondente a massa per asse pari a 18.0 t;
- A - corrispondente a massa per asse pari a 16.0 t;
- Non definita.

Sono poi state definite le data property:

- km4c:combinedTraffic, che può assumere i valori: "PC80", "PC60", "PC50", "PC45", "PC32", "PC30", "PC25", "PC22", "linee con il profilo limite di carico FS" e "non definito";
- dct:identifier, che nel caso dei dati importati dalla Regione Toscana è un codice di 12 caratteri che inizia con i caratteri *RT*, seguito dal codice regionale, da un numero progressivo a 5 cifre, e dalla sigla *TR*;
- foaf:name, cioè la denominazione convenzionale della tratta.

Per la classe km4c:RailwayJunction sono state definite solo tre data property:

- dct:identifier, che nel caso dei dati importati dalla Regione Toscana è un codice identificativo di 12 caratteri formato come nei precedenti casi ma terminante per GK;
- foaf:name, cioè la denominazione ufficiale per bivi, stazioni e scali;
- km4c:juncType, che può assumere uno dei seguenti valori:
    a. passaggio a livello;
    b. terminale (inizio o fine);
    c. bivio (confluenza o diramazione);
    d. stazione/fermata/casello ferroviario;
    e. scalo merci;
    f. interporto;
    g. variazione di stato (COD_STA);
    h. variazione di sede (COD_SED);
    i. variazione numero dei binari (Num_bin);
    j. variazione alimentazione (COD_ALI);
    k. limite amministrativo.

La classe km4c:TrainStation ha un codice identificativo di 12 cifre composto dalla sigla *RT*, seguita da 3 caratteri che identificano la regione (T09 per la Toscana), seguiti da un numero progressivo a 5 cifre,

seguito dalla sigla *SF*, contenuto nella proprietà dct:identifier. Si noti che per quelle istanze generate attraverso la triplificazione della Open Street Map, invece che attraverso l'importazione dei dati della Regione Toscana, l'identificativo assume una forma diversa. Esso inizia infatti per "OS", seguito dall'identificativo numerico dell'elemento Open Street Map che ha originato la produzione dell'istanza, seguito da un'etichetta di due caratteri che indica il tipo dell'istanza generata.

Altre proprietà definite per le stazioni ferroviarie sono:

- foaf:name, in cui è memorizzata la denominazione ufficiale riportata per esteso;
- l'indirizzo reperito dall'elenco pubblicato sul sito RFI e memorizzato nei campi schema:addressRegion, schema:addressLocality, schema:postalCode, schema:streetAddress;
- km4c:managingAuth, cioè l'ente gestore riportato sul sito RFI;
- la categoria a cui appartiene la stazione, conservata nella data property km4c:category;
- campo km4c:state, cioè lo stato della stazione che può assumere i valori:
    a. Attiva;
    b. Non Attiva;
    c. Facoltativa con fermata a richiesta.

La classe km4c:GoodYard presenta:

- il codice identificativo che nel caso dei dati importati dalla Regione Toscana è un codice di 12 caratteri formato come tutti i precedenti ma terminante in SM, memorizzato nel dct:identifier;
- foaf:name, in cui è memorizzato il nome dell'impianto merci;
- km4c:railDepartment, che mantiene la denominazione del compartimento ferroviario;
- km4c:railwaySiding, cioè la definizione della caratteristica fisica del numero di raccordi;
- km4c:yardType che indica se gli scali sono pubblici (valore "scali pubblici") o se i raccordi linea sono ad uso privato (valore "raccordi in linea");
- km4c:state che indica se lo scalo è attivo o in costruzione.

Venendo alla macro-area dell'Internet delle cose (Internet-of-Things, IoT), le principali proprietà che sono definite per le classi km4c:IoTSensor e km4c:IoTActuator sono:

- km4c:format, il formato (e.g. csv, xml, json) dei dati prodotti in uscita dal sensore, o accettati in ingress dall'attuatore;
- km4c:macaddress, l'indirizzo MAC del dispositivo;
- km4c:model, il modello del dispositivo;
- km4c:ownership, se il dispositivo è pubblico o privato;
- km4c:producer, il produttore del dispositivo;
- km4c:protocol, il protocollo di comunicazione supportato dal dispositivo (MQTT, NGSI, e così via).

Il nome e la posizione dei dispositivi sono anch'essi rappresentati, ma attraverso il riuso di proprietà opportune dalle ontologie importate.
Per quanto riguarda i broker, la maggior parte delle loro proprietà sono definite su ontologie esterne importate dall'ontologia Km4City. La proprietà km4c:created è infatti la sola definita direttamente all'interno dell'ontologia di Km4City, ed indica la data ed ora in cui il broker è stato registrato all'interno della Knowledge Base. Altre informazioni che sono rese disponibili per ciascun broker son oil suo endpoint, e il suo nome.
Infine, le principali proprietà della classe km4c:DeviceAttribute sono:

- km4c:value_name, una denominazione orientata all'utente della specifica tipologia di valori prodotti in uscita o accettati in ingresso dal dispositivo;
- km4c:data_type, stringhe, piuttosto che numeri interi, numeri decimali, valori selezionati da un elenco chiuso, o cos'altro;
- km4c:value_unit, unità di misura;
- km4c:order, numero d'ordine, utile nel caso in cui il dispositivo produca in uscita o accetti in ingresso un set di valori eterogenei, appartenenti a tipologie diverse e quindi modellati con istanze di km4c:DeviceAttribute diverse;
- km4c:disabled, una proprietà booleana che indica se l'attributo debba essere tenuto in considerazione (e quindi mostrati agli utenti, indicizzato, e quant'altro), o se debba invece essere trascurato;
- km4c:editable, una proprietà booleana che indica se i valori dell'attributo possano essere soltanto letti (il che tipicamente accade per i sensori), o possano essere anche scritti (il che tipicamente accade per gli attuatori);
- km4c:healthiness_criteria, che indica come debba essere valutato lo stato di salute del dispositivo. Le possibilità sono: (i) il dispositivo non è in salute se produce lo stesso valore in uscita per più di un certo numero fissato di rilevazioni; (ii) i valori non sono prodotti in uscita con la frequenza attesa; (iii) i valori cadono al di fuori di un intervallo prefissato;
- km4c:different_values, se il dispositivo produce rilevazioni identiche in numero superiore a quello indicato da questa proprietà, e se il criterio di valutazione dello stato di salute del dispositivo è basato sul numero di rilevazioni consecutive identiche, il dispositivo viene dichiarato non in salute;
- km4c:value_bounds, che indica l'intervallo dei valori ammessi, tale che se una rilevazione cade fuori dall'intervallo, il dispositivo è dichiarato essere non in salute;
- km4c:value_refresh_rate, che indica la frequenza con cui in condizioni di funzionamento normali, il dispositivo produce valori in uscita.