UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

DIPARTIMENTO DI .....

Corso di Laurea Magistrale

in .....

Contents

1 Semantic Web **3**
1.1 Introduction to the Semantic Web 3
1.1.1 Linked data 4
1.1.2 Vocabularies 6
1.1.3 Query 8
1.2 Ontologies for smart buildings 9
1.2.1 Brick 9
1.2.2 Semantic Sensor Network ontology 10

2 Model creation **13**
2.1 State of the art 13
2.1.1 Physycal model approach 14
2.1.2 Data driven approach 14
2.1.3 Rule-based approach 14
2.2 IBM Research approach 15
2.2.1 Semantic Mapping 16

3 Diagnosis Algorithm **19**

4 IoT Brain **20**
4.1 IoT BRAIN and KITT 20
4.1.1 Graph database vs. RDF Store 21

Introduction

Industrialised countries are facing challanges that are strictly related to energy consumption. Although great efforts in studying novel energy sources are profused by both public and private entities, these solutions need to be sustained by efforts aimed at the reduction of energy wasting. Around 32% of total energy consumption in industrialized countries is used for electricity, heating, ventilation and air-conditiong (HVAC) in commercial buildings. Studies have shown that early detection of faults in those systems and their operation could lead to energy savings between 15% to 32%, improving the overall buildings efficiency. This scenario does not only offer great opportunities related to environmental protection but gives companies a chance for sensible expense reduction. Around 14% of commercial buildings in the US (as of 2012) deployed Building Management System (BMS), supported by the increasing number of available sensors and improvement in the Internet of Things technologies. These efforts end up facing scalability issues due to the lack of common schema for data coming from a variety of building, vendor and location specific sources. Applications developed for ...

Chapter 1

Semantic Web

In this chapater will follow a brief description of the concept of ontology and of Semantic Web as defined by W3C. This chapter will then furter detail the ontologies and metadata schemata used in the thesis project, namely Semantic Sensor Network (SSN) and Brick. It is to note that the work presented in this thesis is not directly using all of these technologies, but it draws from their key concepts to reach some goals. It is therefore worth to mention such concepts here.

1.1 Introduction to the Semantic Web

Semantic Web is a term that identify an evolution of the web in which every resource available through the web is associated with a semantic meaning. Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data. Semantic web layers are defined by W3C as:

- linked data
- vocabularies
- query
- inference

1.1.1 Linked data

Linked Data lies at the heart of what Semantic Web is all about: large scale integration of, and reasoning on, data on the Web. To make the Web of Data a reality, it is important to have the huge amount of data on the Web available in a standard format, reachable and manageable by Semantic Web tools. Furthermore, not only does the Semantic Web need access to data, but relationships among data should be made available, too, to create a Web of Data (as opposed to a sheer collection of datasets). This collection of interrelated datasets on the Web can also be referred to as Linked Data.

Resource Description Framework

RDF [6] is a description language, which aims to capture the semantics of data represented on the Web. The RDF standard is based on several fundamental concepts. The first such concept is "resource". Resources are the basic objects, or "things", that are to be described in the domain (e.g. rooms, floors, sensors etc.). All resources are identified through a unique, global identifier called the universal resource identifier (URI). In most applications, the URI is the uniform resource locator (URL) of a Web page, a part of a Web page, or a link to a document available on a Web server. However, the URI is a more general concept, the only condition is that it uniquely identifies a resource. Examples of resources could be "unibs:florenzi.002" as well as "unibs:thesis1234", or "https://www.w3.org/RDF/". Another essential concept is "property". Properties describe relations between resources and following the previous example we could say that "has_author", "has_name", "has_subject" are properties. Having defined both resources and properties, statements can be derived. Statements, also known as triples, in RDF have the basic form: subject-predicate-object, where the subject is an RDF resource, the predicate is a RDF property, and the object is another RDF resource or a literal (a name, a number, code, etc.). Still referring to the example, the statements could be "unibs:thesis1234 has_author unibs:florenzi.002", "unibs:thesis1234 has_subject `https://www.w3.org/RDF`" and "unibs:florenzi.002 has_name "Fabio Lorenzi"". RDF data models end up being represented as graphs where subjects and objects are nodes while properties are the edges. The resulting

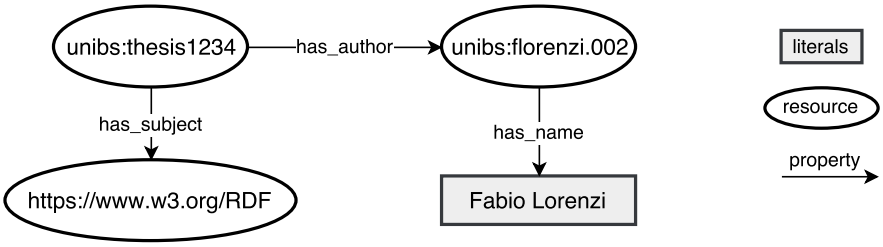

Figure 1.1: Graph representation of the example RDF model

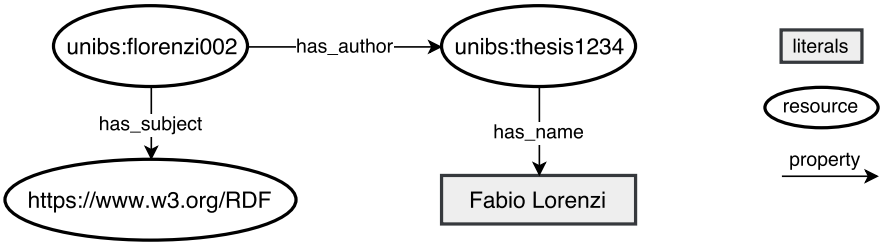

Figure 1.2: Correct RDF model with wrong semantic

graph, given the RDF notation, in the example is shown in Figure 1.1.

It is to note that RDF is a conceptual model for representing linked data, but as far as it goes, no semantic is implied in this representation, looking at this model is impossible to understand the meaning of the resources or property, nor to understand if those elements reflect and adhere to rules of a particular domain. In the example above there is no explicit information telling us that the resource "unibs:florenzi.002" is a person and that "the unibs:thesis1234" is a book. Furthermore the RDF model represented with "unibs:florenzi.002 has_author unibs:thesis1234", "unibs:florenzi.002 has_subject `https://www.w3.org/RDF`" and "unibs:thesis1234 has_name "Fabio Lorenzi"", yelding the graph representation in Figure 1.2, is perfectly fine, altough not semantically correct[1]. Adding semantic to an RDF model, that is detailing the domain ontology, is left to other frameworks as explained in **subsection 1.1.2**.

[1]from a common sense point of view. No semantic is explcitly declared

1.1.2 Vocabularies

On the Semantic Web, vocabularies define the concepts and relationships (also referred to as "terms") used to describe and represent an area of concern. Vocabularies are used to classify the terms that can be used in a particular application, characterize possible relationships, and define possible constraints on using those terms. There is no clear division between what is referred to as "vocabularies" and "ontologies". The trend is to use the word "ontology" for more complex, and possibly quite formal collection of terms, whereas "vocabulary" is used when such strict formalism is not necessarily used or only in a very loose sense. Vocabularies are the basic building blocks for inference techniques on the Semantic Web.

Resource Description Framework Schema

RDFS defines a set of RDF resources useful for the description of other RDF resources and properties. Through RDFS it's possible to define and detail the semantic of a RDF model. Among the various concepts introduced with RDFS there are

- rdfs:Resource: everything that is described in RDF is a resources
- rdfs:Literal: it is text string
- rdfs:Property: it represent the properties
- rdfs:Class: it is similar to the concept of type or class in an object oriented programming language.
- rdfs:subClassOf: it specify inheritance, eventually multiple, between classes.
- rdfs:range: it tells which resources are permitted as object in a statement
- rdfs:domain: it tells which resources are permitted as subject in a statement

it is to note that the RDFS is a RDF model itself and it's used for explicitly declare the domain knowledge of a given RDF model. We refer to the RDFS as an ontology and to the RDF model as the instances of such ontology. Following

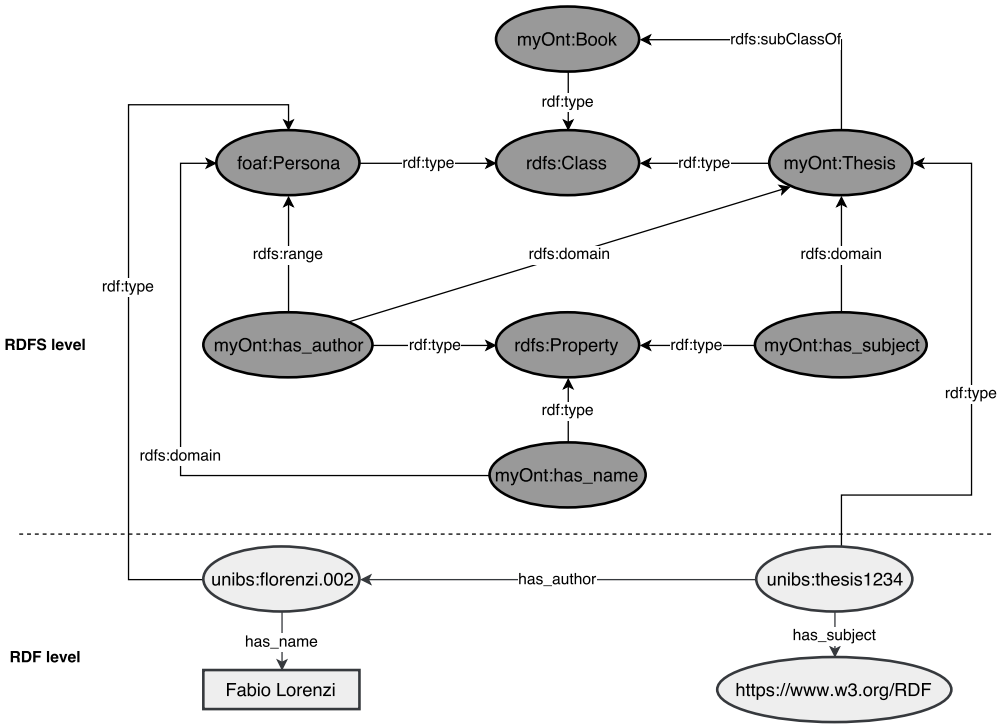

Figure 1.3: RDFS representation of an ontology

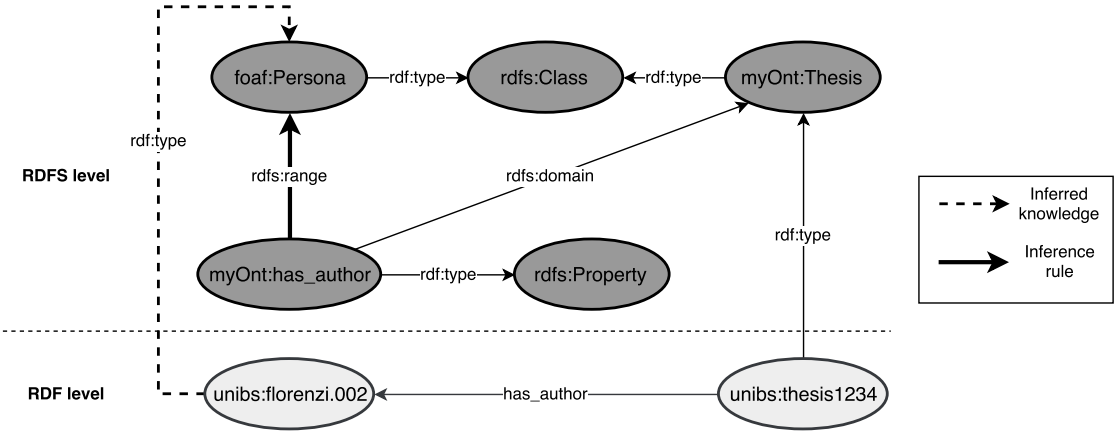

Figure 1.4: Inference process of new knowledge

RDFS, it is possible to model domain knowledge for the example in **section 1.1.1** as in Figure 1.3.

This "layering" approach is what allows for the design of reusable ontologies and gives opportunity to automatically infer new knowledge based on the RDFS model of the data. Still looking back at the example, lets suppose that the statement "unibs:florenzi.002 rdf:type foaf:Person" wasn't in the ground knowledge of the RDF model. Even without this explicit information wasn't available, a reasoner would have been able to infer that statement (dotted line) given the semantic description of the property "myOnt:has_author" (bold line). The process is represented in Figure 1.4.

1.1.3 Query

Just as relational databases or XML need specific query languages (SQL and XQuery), the Web of Data, typically represented using RDF as a data format, needs its own, RDF-specific query language and facilities.

SPARQL Protocol and RDF Query Language

A mean for querying RDF models is defined by SPARQL [9] query language. Technically, SPARQL queries are based on (triple) patterns. As RDF can be seen as a set of relationships among resources, SPARQL queries provide one or

more patterns against such relationships. These triple patterns are similar to RDF triples, except that one or more of the constituent resource references are variables. A SPARQL engine would returns the resources for all triples that match these patterns. To represent a pattern (or query) to run against the data model, SPARQL define a series of keywords

1.2 Ontologies for smart buildings

1.2.1 Brick

Brick schema [1] defines domain specific concepts aimed at describing sensors in a building with its context. It is composed of three parts

- a class hierarchy of entities describing various building subsystems
- a minimal set of relationships for connecting entities
- a methods for encapsulation in the form of Function Blocks

the main concept in Brick is the concept of tag introduced with project Haystack [5], which is further enriched by an ontology that cristallizes the concepts defined by tags. A tag is a flexible framework for annotating metadata to building data points. In Brick tags are grouped together as sets, named tagset, to represent entities; the tags room, temperature and sensor can be grouped together as room temperature sensor representing a single entity. The concept of tagset is well integrated with the class hierarchy concept of the ontologies: a room temperature sensor is a subclass of a generic temperature sensor. As the main goal of Brick is to represent points in a building's BMS, Points is one of the main classes of Brick, which subclasses define specific type of points like Sensor. The most common concepts a Point can be related to are Location, Equipement and Measurement so those classes, together with the Point class, form the core classes of Brick. Those classes are further subclassed to form a hierarchy able to comprehend every entity of a building. Alongside classes, relationships play a crucial role in connecting entities and thus providing a context for many applications. Brick designed a set

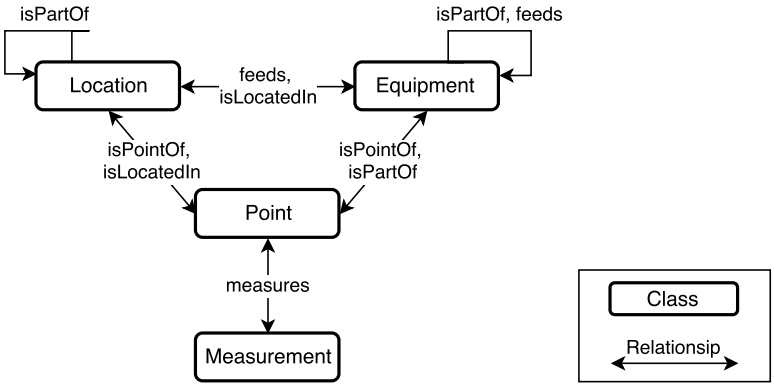

Figure 1.5: Brick schema core concepts

of minimal, intuitive and multipurpose relationship fundamental in capturing existing relationship. Figure 1.5 shows the Brick core concepts and the designed set of relationships.

1.2.2 Semantic Sensor Network ontology

While Brick schema was used to model the building domain, the Semantic Sensor Network (SSN) ontology [2] was chosen because it provides a different view of the various aspect of the sensing process, beyond the specific building domain. Peculiar to the SNN Ontology (SSNO) is the Stimulus-Sensor-Observation design pattern. This pattern differentiate between the concept of ssn:Sensor for physical devices that take measurements in form of ssn:Observation and ssn:Stimulus that is the actual change in the enviroment that caused a particular observation, so that it possible to differentiate that stimuli occur in the reality even if no sensor is monitoring them, thus modeling unobservabel states of a system, and that observation made from a sensor can be different from the real event (e.g errors, failures, noise). These concepts aside, SSN provide other useful classes that are ssn:FeatureOfInterest that represent generic entities which status is to be monitored, like rooms, fans or floors and ssn:Property that embodies specific physical quantities that a sensor ought to monitor. All the classes are connected by a set of relationships that are ssn:hasProperty that connects a ssn:FeatureOfInterest with meaningful physical properties, like a temperature in a room; ssn:isProxyFor represent that a given ssn:Stimulus influence a ssn:Property as in “another occupant

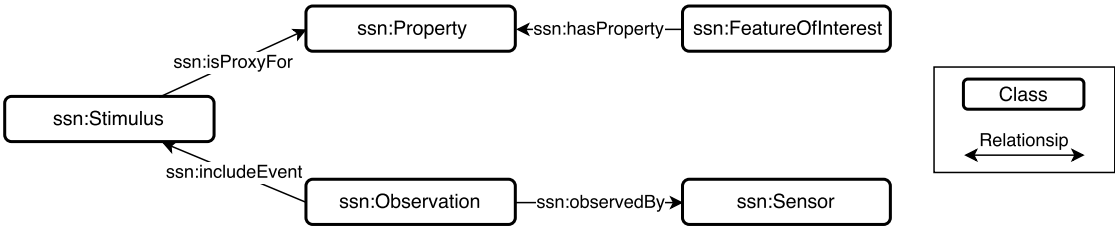

Figure 1.6: Semantic Sensor Network (simplified) ontology

in a room raises its temperature"; ssn:includeEvent create a dependency between an observation and a event that causes it; ssn:observedBy links an observation to the sensor that could measure it. Figure 1.6 show the resulting SSN ontology.

Extending SSNO

As far as it goes Brick and SSN are good ontologies for modeling a building and its sensor network, but in a Energy Management System (EMS) context, and for FDD applications in particular, it is essential to understand the cause-effect relationships in the building. In order to to that, IBM Research extended the SSNO through the introduction of additional classes and references as in Figure 1.7. phy:PhysicalProcess represents laws of physics through a simplified model. Different subclasses of physical processes are derived from LTI system processes and will be described in . phy:FeatureLink tells which kind of relationship exist between two ssn:FeatureOfInterest, such a link could be a wall between two rooms or a window between a room and the outside. phy:Cause and phy:Effect are subclasses of ssn:Stimulus useful for differentiate whether said stimulus is a cause or an effect of a ssn:Observation. phy:Anomaly is a subclass of ssn:Observation, it describes out-of-range observations that need to be diagnosed, like a high temperature in a room. phy:ObservedCause is still a subclass of ssn:Observation; it is used for describing a measurment that can be a cause of an anomaly.

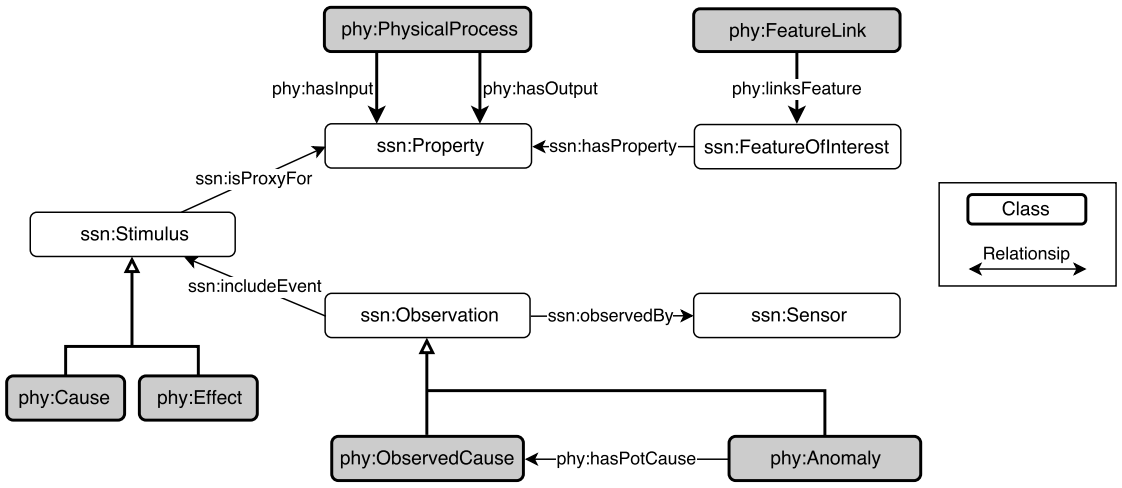

Figure 1.7: Extended SSN ontology. Extensions are highlighted with bold line and a light grey background.

Chapter 2

Model creation

This chapter will outline the approach suggested by IBM Reserch for tackling the challange introduced in **Introduction** regarding fault detection and diagnosis (FDD).

2.1 State of the art

Nowadays, as extensively analyzed by Katipamula and Brambley [3], FDD approaches fall in either one of the following three categories: physycal model based approaches, data driven approaches and rule-based approaches. The rationales behind these approaches are detailed in subsection 2.1.1, subsection 2.1.2 and in subsection 2.1.3. Still, each of these approaches share common limitations that can be summarized as

- they can be applied to a single, specific building
- they requires large manual effort and expertise
- deploy can take several weeks
- they neglect strong interactions between systems

2.1.1 Physycal model approach

These approaches require a physycal model (e.g ordinary differential equations) of the building and its components. They are highly precise in diagnosing faults given a correct model, however, deriving the models is no trivial task and requires times and expertise. On top of this, derived models are building, system and location specific and are difficult to adapt to other buildings than the one they are thought for even if they share similar structure and similar components, thus limiting the scalability of this approach.

2.1.2 Data driven approach

Data driven approaches completely rely on building's sensor data. They assume that access to a large dataset of hystorical data is granted. There's little to no need for any *a priori* knowledge of the processes involved. Black-box data driven approaches derive the model in the form of a input-output relationship which parameters aren't correlated with the actual physical parameter (e.g artificial neural networks, regression) while grey-box data driven approaches take advantage of simplified physical relationships between measured quantities (e.g principal component analysys) and rely on statistical methods for estimating their parameters. Even though these methods don't suffer from scalability issues they are limited to fault detection and lack in diagnosis capabilities.

2.1.3 Rule-based approach

The rule-based approach is the most common in traditional FDD applications. It uses domain knowledge and expertise in order to derive a series of simple *if-then-else* rules or some kind of decision trees along with a series of thresholds and confidence intervals; during system operations data are evaluated against this rules and countermeasures are eventually taken. Even though this is the most common approach, it still needs a lot of manual effort and requires access to domain knowledge so these requirements still limit portability and scalability of applications based on this technique.

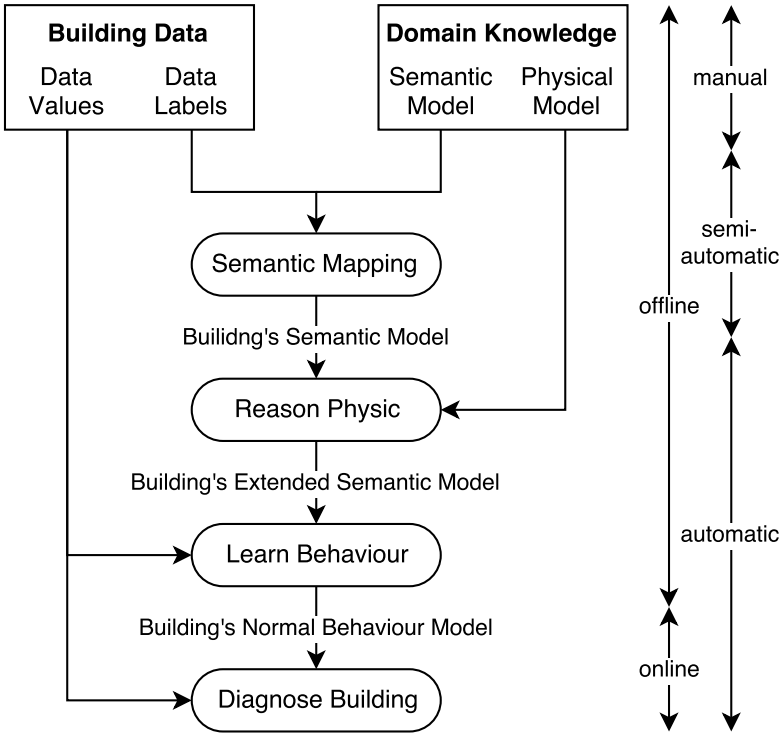

Figure 2.1: IBM Reserch approach, overview

2.2 IBM Research approach

The approach developed by IBM Research is based on a combination of the three aformentioned concepts to overcome their disadvantages. It models high level physical processes in the systems to derive diagnosis rules, it parametrize the rules using data analytics techniques and applies them during systmes operations. The strength of the semantic approach lies in its capability to automatically infer knowledge given a building, its sensors and their data. This lead to a semi-automated process that limit the manual effort needed as shown in Figure 2.1.

The proposed approach needs 3 inputs to produce the diagnosis of a given anomaly

- building data: assumed available in the building's Building Management System (BMS)
- semantic model: specified through the use of concepts from domain ontolo-

gies Brick (subsection 1.2.1) e SSNO (subsection 1.2.2)

- physycal model: the model of the physical dependencies between subsystems of the building. The physical processes are modelled through concepts presented in the extended SSNO (section 1.2.2)

these inputs are processed through the various stages to produce the complete model of the building and its internal processes. Details about the different phases are given in the following pages.

2.2.1 Semantic Mapping

Building a semantic model for a specific building needs knowledge on the type of sensors, meters and various equipment available in said building, their location and their semantic meaning in the chosen ontologies. These informations are usually stored in BMS through labels that, despite some kind of standardization efforts, are usually vendor specific; in some cases even the same vendor ends up changing its labeling schemes throughout the years. Even though there is no common ground to evaluate these labels, usually they are comprised of a series of abbreviations, eventually separated by some special character (e.g "_"), that gives informations about the device's ID, function and location (or the equipment it is part of). An air temperature sensor in a room on ground floor can be labeled AIR_TEMP_R3GF while another air temperature sensor in a room on the first floor may ends up being labeled as RATSF1R9; both the sensors represent a common semantic resource in an ontology and need to be mapped that concept. This process is called semantic mapping. Existing approaches are differentiated in three different types [8]

- semi-automatic: they offer a tool assisting the user in labelling the point to corresponding semantic types and are based on advanced text mining techniques (e.g regular expressions, classifiers)

- data driven: these approaches try to recover the meta-data given the time-series. It is based on the concept that different data-points exhibiting similar timeseries behaviour should be similar themselves.

- active feedback: in these kind of approaches a series of known events are injected into the system and datapoints semantic model is derived observing the effect of the event injection.

Building Energy Asset Discovery tool

BEAD [4] is the tool developed by IBM Research for assisting user in the process of discovery and tagging of sensors. The tool is based on dictionaries of the type $\mathcal{D} : \mathcal{A} \rightarrow \mathcal{MS}$ where $\mathcal{A}$ is the set of all relevant acronyms and $\mathcal{MS}$ is the set of markerset; each markerset is a set of markers, or keywords. These concepts are closely related to those of tagset and tag found in the Brick ontology and this duality allows the association of the sensor with the correct semantic type as described by Brick (see 1.2.1). The dictionary contains the most common acronyms found in BMS and it is further extended by the markers (tags) from the Brick ontology, such that a tag maps to itself, e.g Temperature→{Temperature}. The tool take a label from the BMS and compute a similarity score against the dictionary entries and guide the user in the labeling process. For example, given the following dictionary

d1: RAT→{Return, Air, Temperature}
d2: RAT→{Room, Air, Temperature}
d3: SAT→{Supply, Air, Temperature}
d4: OAT→{Outdoor, Air, Temperature}
d5: Temp→{Temperature}
d6: Temperature→{Temperature}

the label RATSF1R9 (Room Air Temperature Sensor, Floor 1, Room 9) has a high similarity with both RAT→{Return, Air, Temperature} and RAT→{Room, Air, Temperature}, so it's up to the user to understand the meaning and choose the right markerset. In a similar fashion it is possible to extract information on the location of a sensor or the asset it is related to, as shown in Table 2.1.

this process clearly requires human supervision, hence the definition of semi-automatic, but through this approach the operator has to manually inspect a smaller subset of all the possible marketsets [8] thus taking minutes instead of several weeks to complete the semantic mapping. The output of this process is the

Table 2.1: Dictionary extended with assets' informations

Label	**Asset**	**Marketset**
U6_RAT	AHU6	{Return, Air, Temperature}
U6_DAT	AHU6	{Discharge, Air, Temperature}
AHU7_RAT	AHU7	{Return, Air, Temperature}
AHU7_SAT	AHU7	{Supply, Air, Temperature}
AU9_RET_TEMP	AHU9	{Return, Air, Temperature}
AU9_SUP_TEMP	AHU9	{Supply, Air, Temperature}

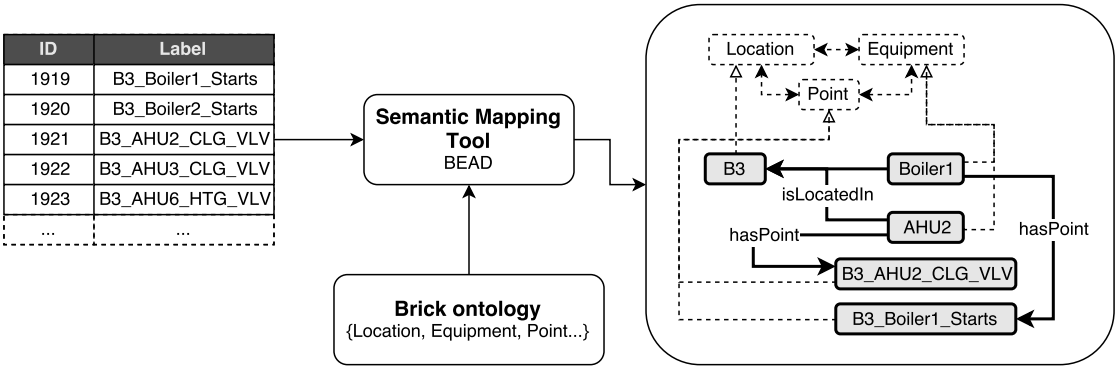

Figure 2.2: Semantic mapping process

semantic model of the specific building. An chart of the whole process is shown in Figure 2.2.

Chapter 3

Diagnosis Algorithm

Chapter 4

IoT Brain

This chapter is going to give some informations about the design choice taken while implementing the approaches in **chapter 2** and **chapter 3**. Implementation efforts were aimed at providing a further degree of scalability to the approach and leveraging cloud based technologies as well as reducing the technicality of the involved resource. It will be presented a use case that shows the

4.1 IoT BRAIN and KITT

IoT Big scale Reasoning and Analytic INsights (IoT BRAIN) allows to specify semantic models for IoT systems and run analytics across them automatically. Its core is the light-weight knowledge framework named Knowledge Inference Technology for IoT (KITT) that provides a simple API to model, manage and reason on the semantic knowledge model. It is split in multiple layers of different abstraction level that base on a common semantic modeling strategy. This architecture implements the concepts explained in the previous chapters with a strong emphasis on the scalability and availability of the approach.

IoT BRAIN, which architecture is shown in Figure 4.1, is built upon a storage layer that is comprised of a graph database, used for storing the semantic model of the building. The Watson IoT platform allows easy connection and dispatch of data securely to the cloud using the open, lightweight MQTT messaging protocol. The storage layer is completed by a dashDB, an SQL database that provides

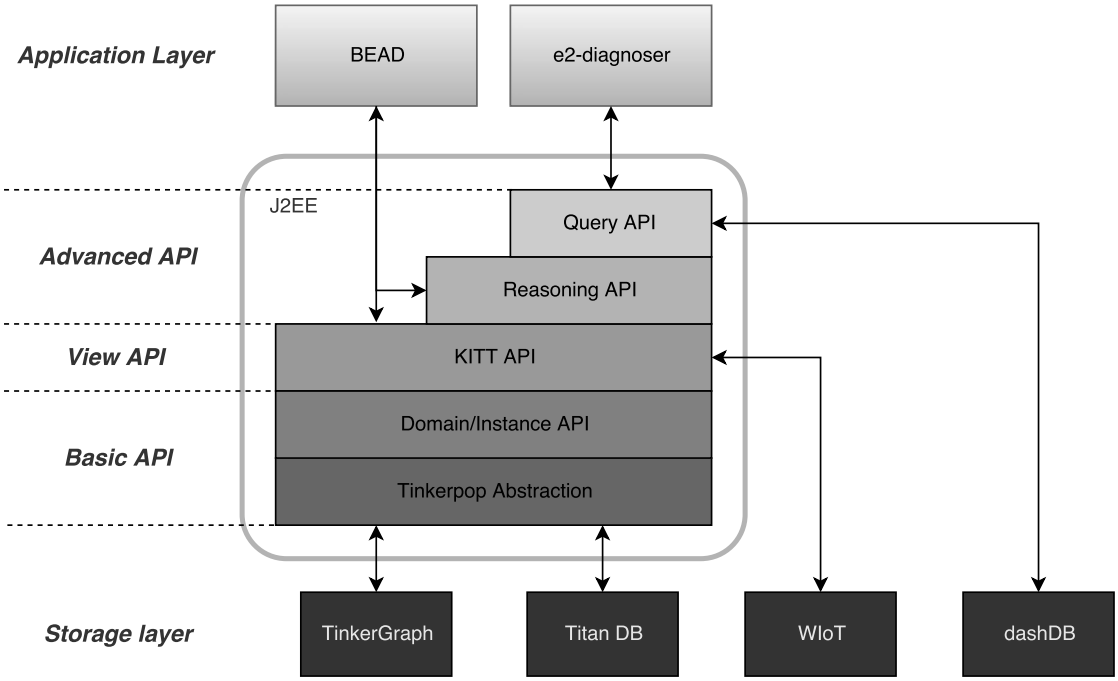

Figure 4.1: IoT BRAIN architecture

efficient storage of time series and high query performance. On top of this layer sits the core of the project which realize the theoric process shown in Figure 2.1.

4.1.1 Graph database vs. RDF Store

why graphDB: cloud property centric need to evolve SPARQL

Bibliography

[1] Bharathan Balaji et al. "Brick: Towards a Unified Metadata Schema For Buildings". In: (Nov. 2016), pp. 41–50.

[2] Michael Compton et al. "The SSN ontology of the W3C semantic sensor network incubator group". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 17 (2012), pp. 25–32. ISSN: 1570-8268.

[3] Srinivas Katipamula and Michael R. Brambley. "Methods for fault detection, diagnostics, and prognostics for building systems — a review". In: *HVAC&R Research* 11.1 (2005), pp. 3–25.

[4] Joern Ploennigs et al. "Demo Abstract: BEAD - Building Energy Asset Discovery Tool for Automating Smart Building Analytics". In: (Nov. 2014).

[5] *Project Haystack*. URL: `https://project-haystack.org/`.

[6] *Resource Description Framework*. URL: `http://www.w3.org/RDF`.

[7] Ioana Robu, Valentin Robu, and Benoit Thirion. "An introduction to the Semantic Web for health sciences librarians". In: 94 (May 2006), pp. 198–205.

[8] Anika Schumann, Joern Ploennigs, and Bernard Gorman. "Towards Automating the Deployment of Energy Saving Approaches in Buildings". In: (Nov. 2014).

[9] *SPARQL Protocol and RDF Query Language*. URL: `https://www.w3.org/TR/rdf-sparql-query`.