



UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
Corso di Laurea Magistrale in Ingegneria Informatica

Contents

1	Semantic Web	4
1.1	Introduction to the Semantic Web	4
1.1.1	Linked data	5
1.1.2	Vocabularies	7
1.1.3	Query	8
1.2	Ontologies for smart buildings	10
1.2.1	Brick	10
1.2.2	Semantic Sensor Network ontology	12
2	Model creation	14
2.1	State of the art	14
2.2	IBM Research approach	15
2.2.1	Semantic Mapping	17
2.2.2	Physic model inference	20
2.2.3	Learning the normal behaviour	23
2.3	Example of the whole approach	23
3	Diagnosis Algorithm	31
3.1	General description	31
3.2	Potential causes	34
3.3	Fault identification	36
3.4	Diagnosis example	40
4	IoT Brain	42
4.1	IoT BRAIN and KITT	42

4.1.1	Graph database vs. RDF Store	43
-------	--	----

Introduction

Industrialised countries are facing challenges 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-conditioning (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. Still those data are just barely used for simple analytics due to the lack of common schema for data coming from a variety of building, vendor and location specific sources that prevent the developers to easily integrate and process those data in a more detailed way. Energy Management Systems (EMS) are usually monolithic and poorly integrated solutions tailored for specific buildings, usually these solutions are costly in terms of both money and time. They require efforts to encode expert knowledge that can't be reused.

Chapter 1

Semantic Web

In this chapter will follow a brief description of the concept of ontology and of Semantic Web as defined by W3C. This chapter will then further 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

Resource Description Framework (RDF)[10] is a description language, which aims to capture the semantic of linked data. The RDF standard is based on several fundamental concepts. The first such concept is “resource”. Resources are the basic objects, or “things”, that have 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, a 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

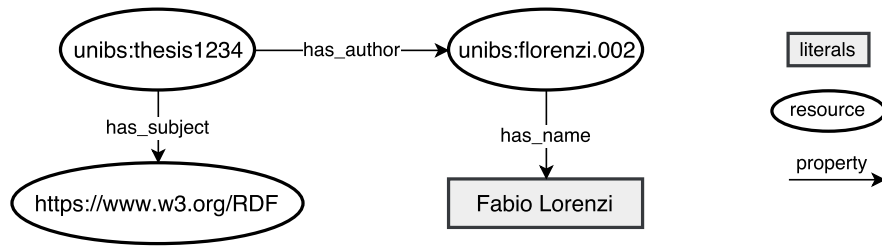


Figure 1.1: Graph representation of the example RDF model

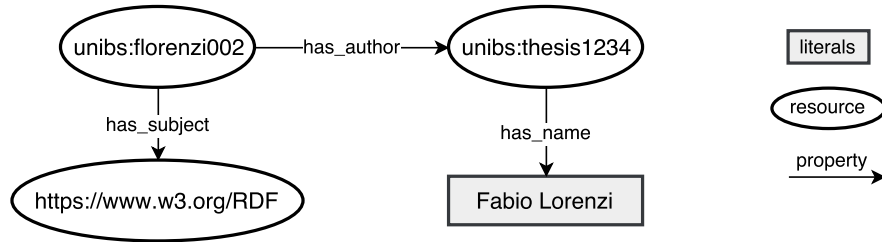


Figure 1.2: Correct RDF model with wrong semantic

are nodes while properties are edges. The resulting graph derived from the description above 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 it is impossible to understand the meaning of the resources or properties, 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””, yielding the graph representation in Figure 1.2, is perfectly fine, although not semantically correct¹. Adding semantic to an RDF model, that is detailing the domain ontology, is left to other frameworks as explained in subsection 1.1.2.

¹From a common sense point of view. No semantic is actually explicitly 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

Resource Description Framework Schema (RDFS) defines a set of RDF resources useful for the description of other RDF resources and properties. Through RDFS it is 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 resource
- **rdfs:Literal**: it is a text string
- **rdfs:Property**: it represents the properties
- **rdfs:Class**: it is similar to the concept of type or class in an object oriented programming language
- **rdfs:subClassOf**: it specifies 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 is 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 triples as the instances of such ontology. Using

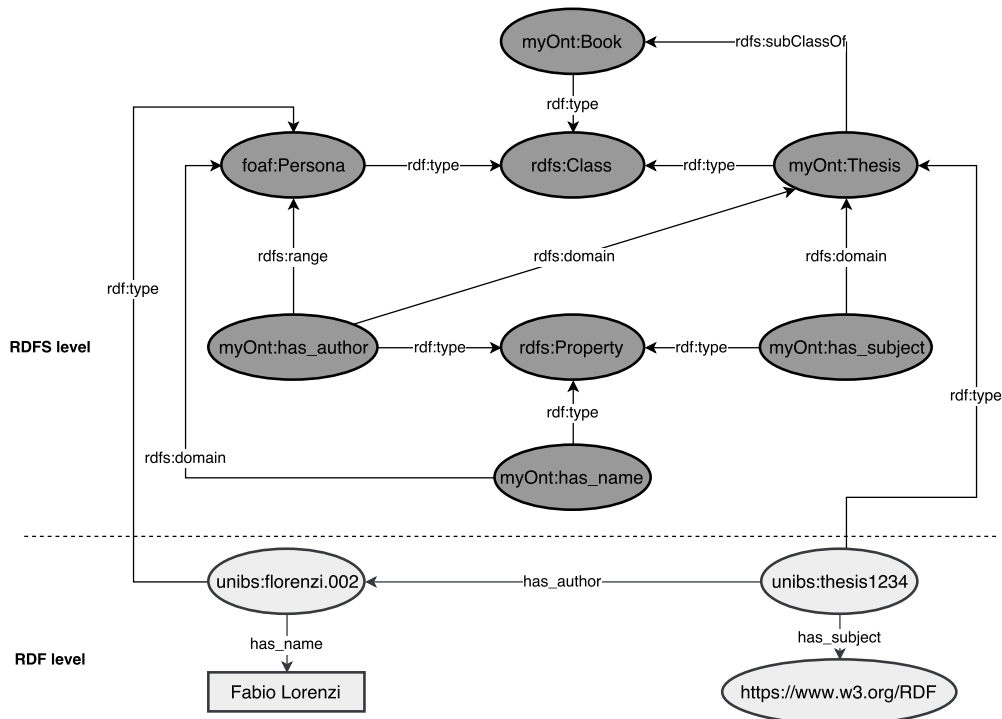


Figure 1.3: RDFS representation of an ontology

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, let's suppose that the statement “unibs:florenzi.002 rdfs:type foaf:Person” is not in the ground knowledge of the RDF model. Even if 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

Since there exist technologies that represent knowledge bases and their semantics, it is useful, if not necessary, to have access to that knowledge through some kind of query language that allows the user to extract and consume that knowledge. If the knowledge base is stored as a RDF, the de-facto standard is represented by

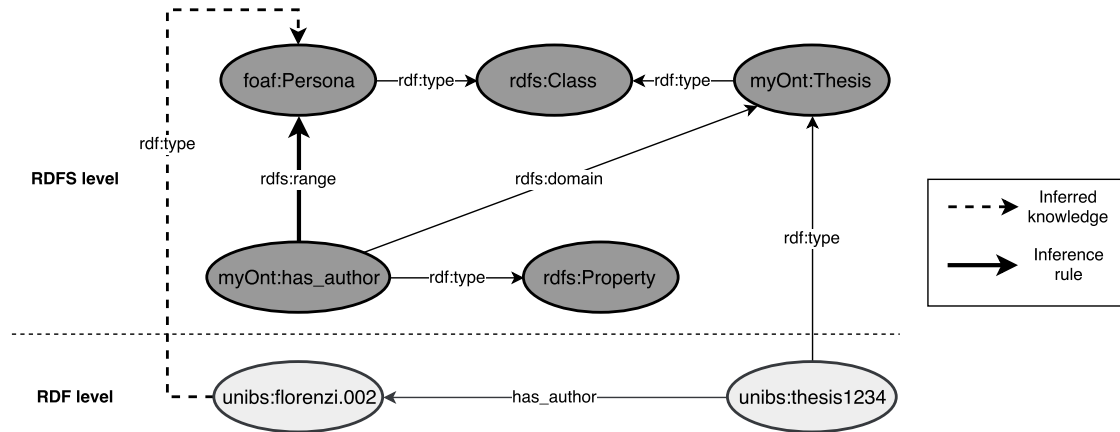


Figure 1.4: Inference process of new knowledge

SPARQL.

SPARQL Protocol and RDF Query Language

A mean for querying RDF models is defined by SPARQL [13] query language. SPARQL queries are based on patterns. As RDF can be seen as a set of relationships among resources, SPARQL queries provide one or more patterns against such relationships. A SPARQL engine would perform a pattern matching over the resources and it returns the matching triples. To represent a pattern (or query) to be run against the data model, SPARQL defines a series of keywords similar to those of SQL language:

- **SELECT** followed by the name of the variables that the user wishes to SELECT
- **INSERT** used to insert new triples pattern in the underlying RDF [defined in an extension called SPARQL/Update]
- **WHERE** followed by a set of conditions involving the variable that needs to be selected and that needs to be fulfilled in the underlying RDF.

As an example, from Figure 1.3, a standard query would be to retrieve every thesis and its author in the knowledge base and it can be written as shown below.

```

SELECT ?thesis ?author
WHERE
  ?thesis a myOnt:Thesis
  ?thesis has_author ?author

```

1.2 Ontologies for smart buildings

After the brief (and not exhaustive) introduction to the core concept of the Semantic Web, here are introduced the two main ontologies that lie at the core of the project of this thesis. These ontologies are used to represent a smart building with all its relevant attributes (Brick) and the internal processes that take place involving the building sensors and physic variables (SSN and its extension). Both the ontologies are used since they are ortogonal one another and this help to cover every aspect of interest for a EMS application in a smart building.

1.2.1 Brick

Brick schema [1] defines domain specific concepts aimed to describe 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 method for encapsulation in the form of Function Blocks.

The main concept in Brick is the concept of tag introduced with project Haystack [9], 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 tagsets, 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,

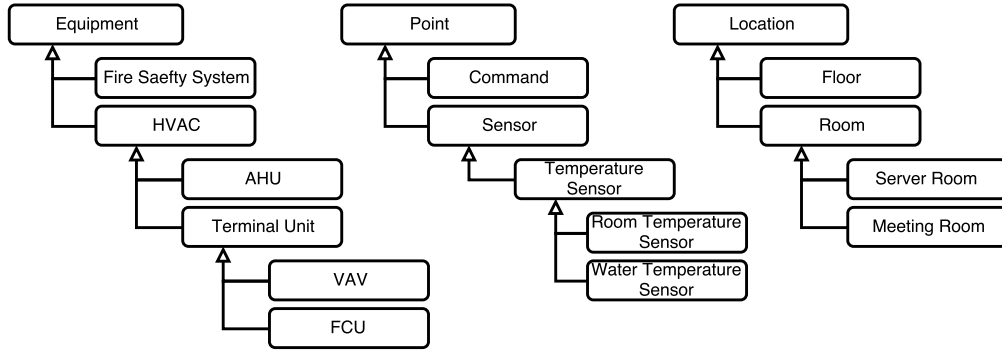


Figure 1.5: Subset of Brick hierarchy

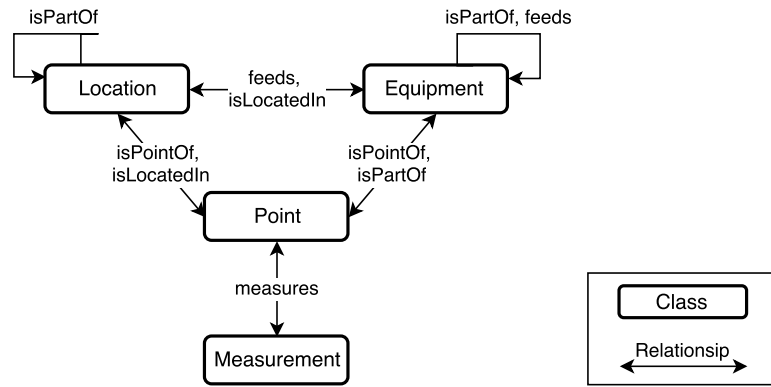


Figure 1.6: Brick schema core concepts

which subclasses define specific type of points like Sensor. The most common concepts a Point can be related to are Location, Equipment 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. A subset of the hierarchy is shown, as an example, in Figure 1.5. this taxonomy is able to comprehend almost every entity that can be found in a BMS of a buidling[1]. Alongside classes, relationships play a crucial role in connecting entities and thus providing a context for many applications. Brick designs a set of minimal, intuitive and multipurpose relationship fundamental in capturing existing relationships in a real building. Figure 1.6 shows the Brick core concepts and the designed set of relationships.

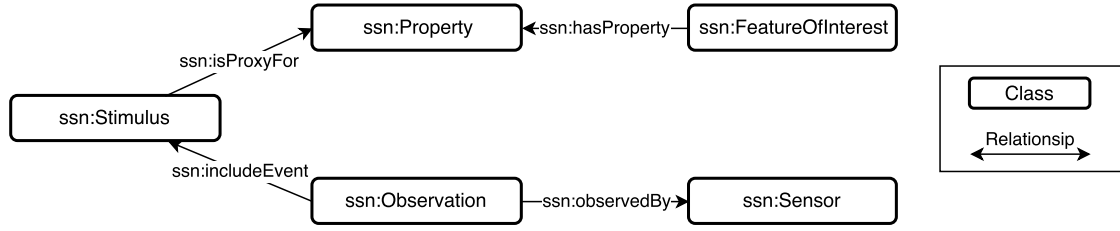


Figure 1.7: Semantic Sensor Network (simplified) ontology

1.2.2 Semantic Sensor Network ontology

While Brick schema is used to model the building domain, the Semantic Sensor Network (SSN) ontology [2] has been chosen because it provides a different view of the various aspect of the sensing process, beyond the specific building domain. Peculiar to the SSN Ontology (SSNO) is the Stimulus-Sensor-Observation design pattern. This pattern differentiates 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 environment that caused a particular observation, so that it is possible to differentiate that stimuli occur in the reality even if no sensor is monitoring them, thus modelling unobservable 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 provides other useful classes that are `ssn:FeatureOfInterest` that represents generic entities which status is to be monitored (e.g 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` represents that a given `ssn:Stimulus` influences a `ssn:Property`; `ssn:includeEvent` create a dependency between an observation and an event that causes it; `ssn:observedBy` links an observation to the sensor that could measure it. Figure 1.7 shows the resulting SSN ontology core.

Extending SSNO

As far as it goes, Brick and SSN are good ontologies for modelling a building and its sensor network, but in a Energy Management System (EMS) context, and for

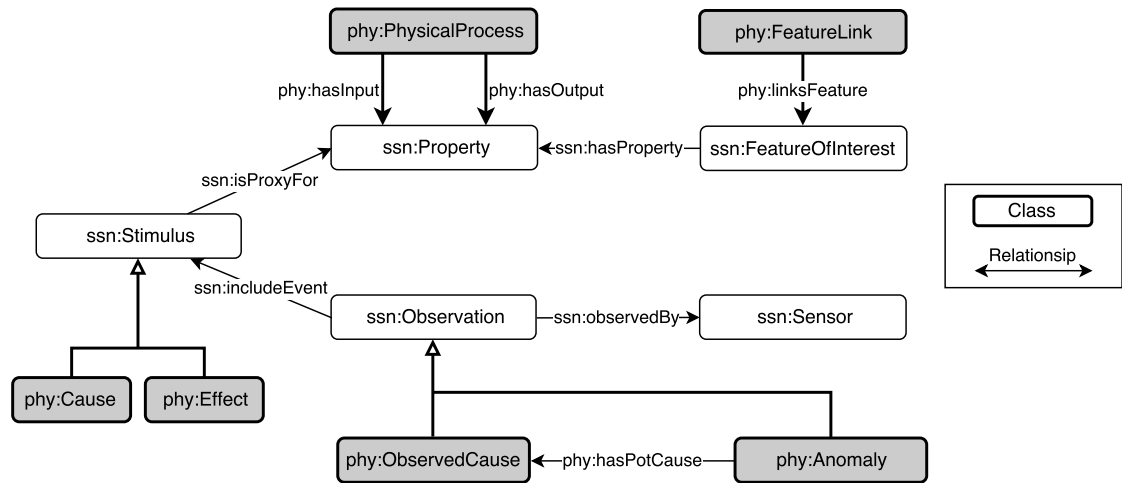


Figure 1.8: Extended SSN ontology. Extensions are highlighted with bold line and a light grey background.

FDD applications in particular, it is essential to understand the cause-effect relationships in the building. In order to do that, IBM Research extended the SSNO through the introduction of additional classes and references as in Figure 1.8. `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 chapter 2. `phy:FeatureLink` tells which kind of relationship exists 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 to 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 measurement that can be a cause of an anomaly.

Chapter 2

Model creation

This chapter will outline the approach suggested by IBM Research for tackling some of the challenges introduced in Introduction regarding fault detection and diagnosis (FDD) and for overcoming the limitations of the state of the art approaches. A step-by-step example illustrating the approach is provided at the end of the chapter.

2.1 State of the art

Nowadays, as extensively analyzed by Katipamula and Brambley [4], FDD approaches fall in either one of the following three categories: physycal model based approaches, data driven approaches and rule-based approaches. Still, each of these approaches shares common limitations that can be summarized as:

- they can only be applied to a single, specific building
- they require large manual effort and expertise
- their deployment can take several weeks
- they neglect strong interactions between systems.

Physycal model approach This approach requires 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 they 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.

Data driven approach Data driven approaches completely rely on building's sensor data. They assume that access to a large dataset of historical data is granted. There is 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 an input-output relationship whose parameters are not 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 analysis) and rely on statistical methods for estimating their parameters. Even though these methods do not suffer from scalability issues, they are limited to fault detection and lack in diagnosis capabilities.

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 these 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.

2.2 IBM Research approach

The approach developed by IBM Research is based on a combination of the three aforementioned concepts to overcome their disadvantages. It models high level physical processes in the systems to derive diagnosis rules, it parametrizes the rules using data analytics techniques and applies them during system's operations.

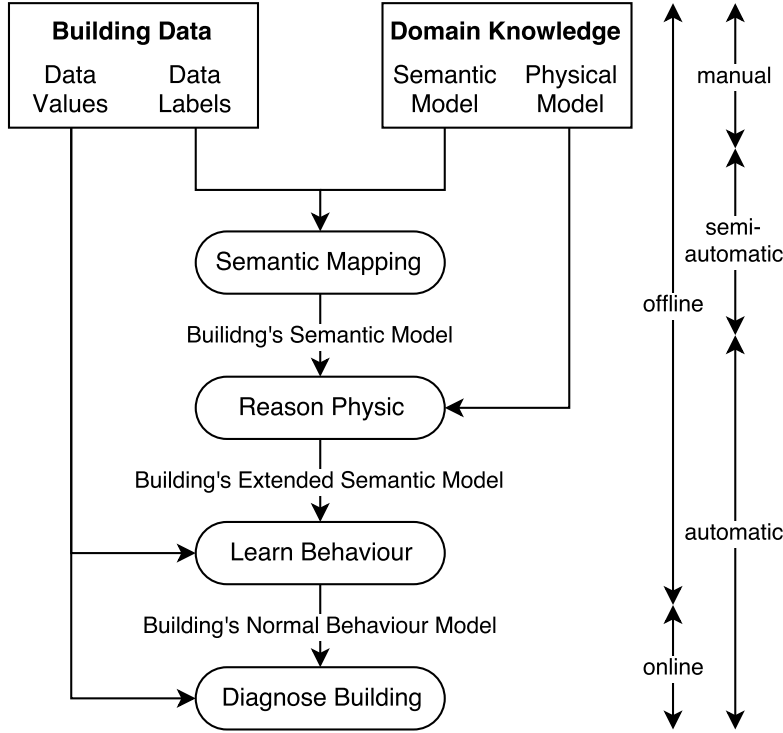


Figure 2.1: IBM Research approach, overview

The strength of the semantic approach lies in its capability to automatically infer knowledge given a building, its sensors and their data. This leads to a semi-automated process that limits the manual effort needed, as shown in Figure 2.1.

The proposed approach needs three 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 ontologies Brick (subsection 1.2.1) and 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. (subsection 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 labelling schemes throughout the years. Even though there is no common ground to evaluate these labels, usually they are comprised of a series of abbreviations and acronyms, 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 to that concept. This process is called semantic mapping. Existing approaches are differentiated in three types[12]:

- 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. They are 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

the Building Energy Asset Discovery (BEAD) tool[6] 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 subsection 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 $\text{Temperature} \rightarrow \{\text{Temperature}\}$. The tool takes a label from the BMS, computes a similarity score against the dictionary entries and guides the user in the labelling process. For example, given the following dictionary

- d1:** $\text{RAT} \rightarrow \{\text{Return, Air, Temperature}\}$
- d2:** $\text{RAT} \rightarrow \{\text{Room, Air, Temperature}\}$
- d3:** $\text{SAT} \rightarrow \{\text{Supply, Air, Temperature}\}$
- d4:** $\text{OAT} \rightarrow \{\text{Outdoor, Air, Temperature}\}$
- d5:** $\text{Temp} \rightarrow \{\text{Temperature}\}$
- d6:** $\text{Temperature} \rightarrow \{\text{Temperature}\}$

the label RATSF1R9 (Room Air Temperature Sensor, Floor 1, Room 9) has a high similarity with both $\text{RAT} \rightarrow \{\text{Return, Air, Temperature}\}$ and $\text{RAT} \rightarrow \{\text{Room, Air, Temperature}\}$, so it is 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, thus taking minutes instead of several weeks to complete the semantic mapping. Practical experiments demonstrated that through this approach the decrease in number of labels to analyze is the 7.5% of the total number of BMS labels [12]. The output of this process is the semantic model of the specific building according to the chosen ontology. A flowchart illustrating this process is shown in Figure 2.2.

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}

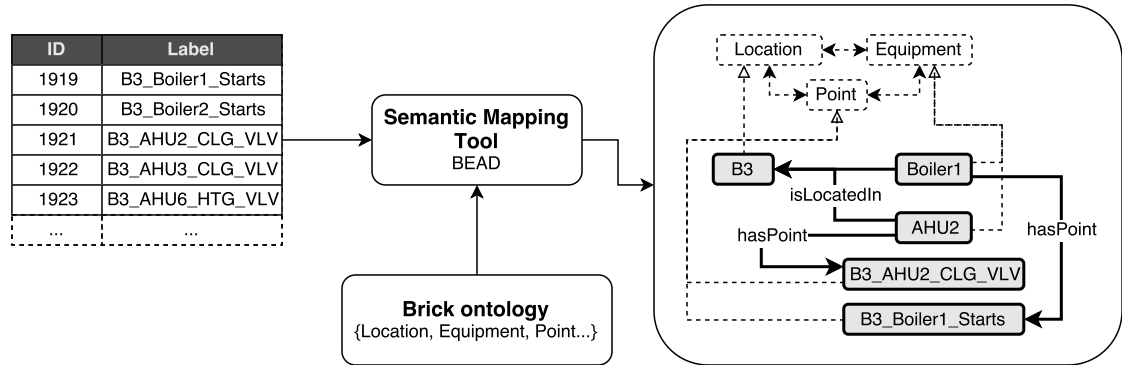


Figure 2.2: Semantic mapping process

2.2.2 Physic model inference

The semantic model obtained from the semantic mapping phase is a model of the building architecture, assets, sensors and their location across the building. The model still lacks information about the physic processes taking place in the building and how the actual physic model is monitored and influenced by the points (sensors, setpoints etc.) in the building. Manually building the physical model of a building is unpractical and costly, thus the developed framework approaches this challenge through the use of semantic inference techniques. This phase requires as inputs the semantic model of the building and an appropriate ontology that models the physics variables. Given those inputs, through the reasoning process is possible to automatically add the correct dependencies and physical processes to the model. This process is separated in two phases:

- physical properties inference: it recreates the physical variables that are involved in physical processes. The properties are either mandatory or optional
- physical processes inference: once the physical properties are set in place, it is possible to link them together according to the domain knowledge specified in the upper ontology.

Further informations about the concept of reasoning and reasoner and further details about implementation choices for this approach are given in chapter 4.

Physical properties inference Properties represent physical quantities in the real world and can be mandatory or optional. Mandatory properties are characteristic of a `ssn:FeatureOfInterest` and need to be created for each instance of that feature. This behaviour is specified in the ontology of the physic through the use of the `phy:requiresProperty` relationship, so that during the reasoning mandatory properties can be automatically created. For example, in Figure 2.3 it is shown that the concept of `:Room` `phy:requiresProperty` `:Temperature`, that means that every instance of a `:Room` needs to be associated with a property that is an instance of `:Temperature`, or in plain English that every room has a temperature. Optional properties, on the other hand, need to be created only if some requirements are met, for example a `:Room` have a `:Cooling` property only if a

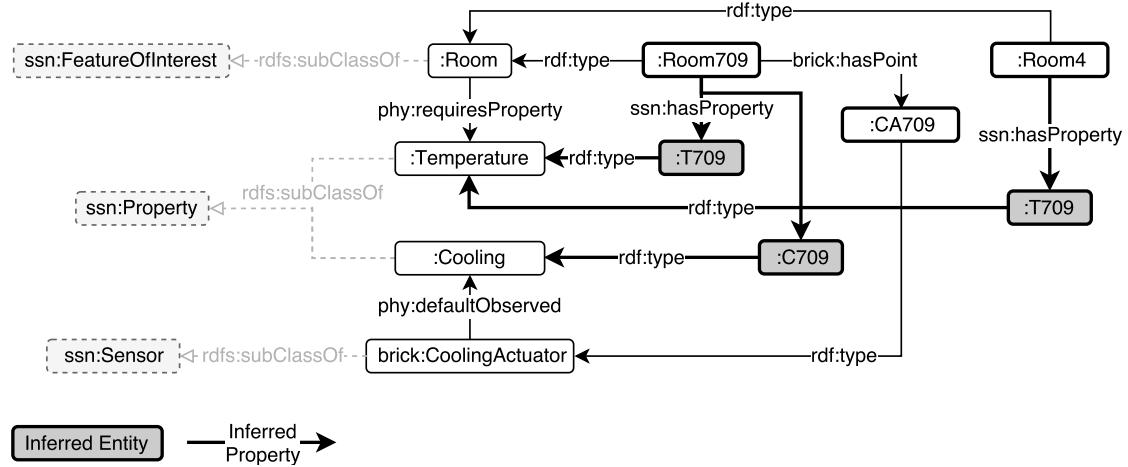


Figure 2.3: Process of inference of mandatory and optional properties

:CoolingActuator is present in said room, and it is modelled in the ontology as :CoolingActuator phy:defaultObserved :Cooling. In Figure 2.3 it can be seen that only :Room709 has a :Cooling property since it is the only room that has a :CoolingActuator. The :requiresProperty and :defaultObserved properties¹ are concept level references and model general knowledge. General knowledge describes the domain without bothering about the ground truth. This means, for example, that the concept of :Room requires the existence of the concept of :Energy even if there are no rooms instances in the building's model. The existence of said properties is the key for the automatic reasoning process.

Physical processes inference Once all the actual properties have been inferred for a specific configuration, it is time to append to the model informations about the physical processes involving such properties. As already stated in subsection 1.2.2, physical processes are modelled as Linear Time Invariant (LTI) systems based on the fact that Multiple Input Multiple Output (MIMO) processes can be

¹Intended as the semantic concept of relationship between entities, not to be confused with domain knowledge properties like :Temperature.

Table 2.2: Taxonomy of the semantic representation of LTI processes

Semantic concept	Parent	Formal definition
MISO	Process	$y_i(t) = \mathbf{c}_i \cdot \mathbf{x}_i + \mathbf{d}_i \cdot \mathbf{u}_i(t) = f(\mathbf{u}(t))$
SISO	MISO	$y_i(t) = \mathbf{c}_i \cdot \mathbf{x}_i + d_i \cdot u_i(t) = f(u(t))$
Positive Correlated (PC)	SISO	$y(t) \propto u(t)$
Negative Correlated (NC)	SISO	$y(t) \propto -u(t)$
Proportional (P)	SISO	$y(t) = k_p \cdot u(t)$
Positive Proportional (PP)	PC, P	$y(t) = k_p \cdot u(t), k_p \geq 0$
Negative Proportional (NP)	NC, P	$y(t) = k_p \cdot u(t), k_p < 0$
Negation (N)	NP	$y(t) = k_p \cdot u(t), k_p = -1$
Integral (I)	P	$y(t) = k_I \int u(t) dt$
Derivative (D)	P	$y(t) = k_D \cdot \dot{u}(t)$
Lag (PTn)	P	$\sum_{i=0}^n C_i \cdot T^i \cdot y^{(i)}(t) = k_{PT} \cdot u(t)$
1 st order lag (PT1)	PTn	$T \cdot \dot{y}(t) + y(t) = k_{PT} \cdot u(t)$
Delay (τ)	SISO	$y(t) = u(t - \tau), \tau > 0$
Multiplicative (M)	MISO	$y(t) = u_1(t) \cdot u_2(t)$

decomposed into multiple Multiple Input Single Output (MISO) processes

$$\begin{aligned}\dot{x} &= A \cdot x + B \cdot u \\ y &= C \cdot x + D \cdot u\end{aligned}$$

that splitted on the output rows and in case of independent inputs becomes

$$y_i = \sum_{j \in J} \left[\sum_{l \in L_j} c_{i,l} \cdot x_l + \sum_{k \in K_j} d_{i,k} \cdot u_k \right] = \sum_{j \in J} f_{i,j}(u_j) \quad (2.1)$$

A MISO process can be further subclassed as a Single Input Single Output process. Each process follows the rules

Rule 1 a process has at least one input and exactly one output

Rule 2 when a process has multiple inputs, they superpose additively.

Given these rules, it is possible to define a class hierarchy of LTI processes that are used to model a physical system. A subset of said taxonomy is available at Table 2.2.

2.2.3 Learning the normal behaviour

This phase is not strictly related to the ones detailed in the previous sections in this chapter, yet it is of utmost importance for the diagnosis process. This phase allows for the definition of general rules that represent a confidence interval of the normal behaviour of the sensors. These rules are derived from different statistical and machine learning methods. Ploennigs and Schumann [5] suggest the use of Artificial Neural Networks (ANN) for modelling the temperature comfort in a room. Generalized Addictive Models (GAM) can be used for modelling energy processes so that a subsequent Autoregressive Moving Avarage (ARMA) model can characterize the normal behaviour[7]. The approach determines a series of upper and lower bounds that characterize the normality interval for a given sensor. Behaviours outside these ranges are classified as anomaly and they trigger the diagnosis phase. Studying this methods is outside the scope of the thesis and so these confidence intervals are assumed to be known without bothering about how they were derived.

2.3 Example of the whole approach

In order to better explain the concepts presented in the previous sections, it is provided an example of the full process applied to a room, facing the outside environment, with a cooling Air Handling Unit (AHU). At the end of the section it is shown that a full model of the room and its internal and external processes are derived. This model will then be used during the explanation of the diagnosis process. Looking back at Figure 2.1, we start by defining the inputs needed for the process (sharp top boxes).

Building data and semantic model It is assumed that some kind of BMS is available in the building so that through the BMS is it possible to retrieve both labels (for semantic mapping) and data of various sensors. The room analyzed in the example will be named ROOM1. In ROOM1 an air temperature sensor and an occupancy sensor are available. Room1 is also equipped with an AHU, named AHU709, whose internal components are observed by a fan mode sensor, a supply

Table 2.3: Example of BMS content

ID	Label
1234	CCPC1B3
1235	CWFU709
1236	CWTU709
1237	SAFAHU709C1
1238	FMAHU709
1239	OCCR1
1240	ATSROOM1
1241	ATS_Outside

air flow sensor a coil water temperature sensor and a coil water flow sensor. This AHU709 is fed by a chiller named C1 equipped with a power sensor. Let the BMS contain the data as in Table 2.3 and let's use Brick (1.2.1) as a domain ontology. Through the semantic mapping tool (subsection 2.2.1) it is possible to map the labels in the BMS to the Semantic type in the Brick ontology. The user can also give the system additional informations about the architecture of the building and the location of equipments and sensors. It is then possible to map :B3 as a brick:Location, :ROOM1 still as brick:Location and then link one to the other as :ROOM1 brick:isPartOf :B3 and so on. The tool will also discover the semantic type of the sensor and will be able to create the correct instances. For example it will add to the model an air temperature sensor named :ATSROOM1 and will link it as a point to :ROOM1. The output semantic graph can be seen in Figure 2.4 where the instances created from the BMS and their mutual relationships are bold, while the Brick ontology concepts and relationships are light grey. This is the building semantic model, that will be used together with the physic model described in the next paragraph to infer knowledge about the physics properties and relationships in the building.

Physic model The physic model is a model of the physic phenomena occurring in an environment. It defines physical variables and processes and their interaction in the context of a specific domain. Creating a model of physic's laws is a time consuming task that involves expertise from different fields, from engineering to physics, that are very domain specific. The good side of the semantic approach is

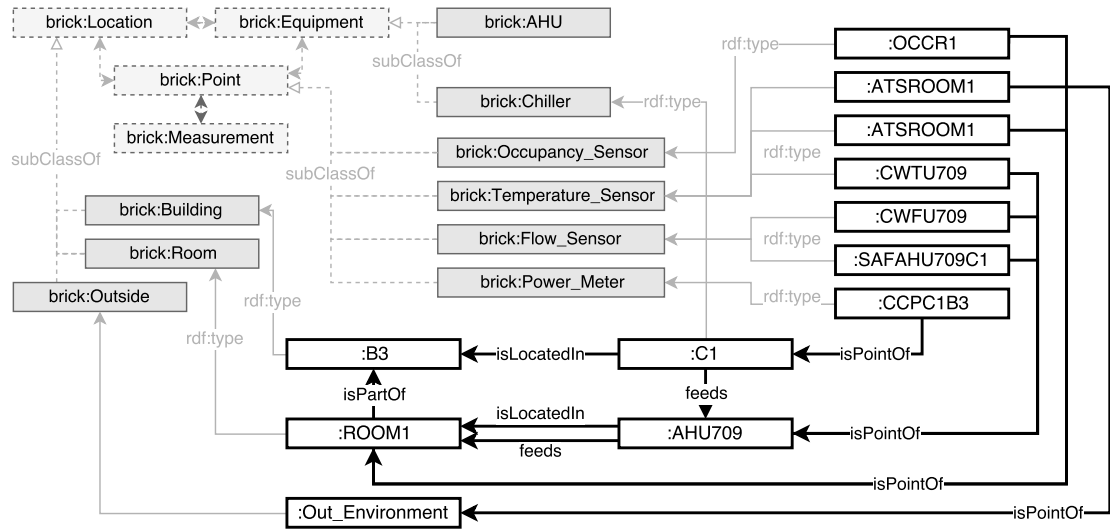


Figure 2.4: Complete semantic model of a room in a building equipped with an AHU according to Brick ontology

that, once the model is derived from a general point of view, it is automatically extended for every instance of a building, independently from its architecture. It is to note that the domain model for the physic does not necessarily model exact physical relationships but just the principal ones, since those are the most relevant to the approach. However this choice is purely practical rather than a limitation of the approach itself, and given sufficient time and knowledge, it is possible to enrich the model in order to close the gap with the reality. Here will be presented a model for the physic of a room, equipped with a AHU fed by a chiller. The model is based on a simplification that makes the assumption of fully mixed air, that allows for the modelling of a whole space as a single node with uniform temperature. Wetter [15] provides an insight of the physics involved in the example. It starts describing a heat balance equation of a room as

$$m_r c_p \dot{\vartheta}_r = Q_{Energy} \quad (2.2)$$

$$Q_{Energy} = Q_{Occ} + Q_{Cool} + \sum A_i h_i (\vartheta_i - \vartheta_r) \quad (2.3)$$

$$(2.4)$$

and from Equation 2.2 and Equation 2.3 it derives

$$m_r c_p \dot{\vartheta}_r + \sum A_i h_i \vartheta_r = Q_{Occ} + Q_{Cool} + \sum A_i h_i \vartheta_i \quad (2.5)$$

that says that the temperature of a room ϑ_r is a first order derivative of the room's inner energy Q_{Energy} that is the combination of the heat gain of occupants Q_{Occ} , the heat removed by the cooling system Q_{Cool} and the heat transfer between adjacent environments. Given these equations, semantic informations can be extracted. From Equation 2.2 it can be seen that

1. rooms have an internal energy [Property, mandatory]
2. rooms have a temperature [Property, mandatory]

while from Equation 2.3 it follows that

3. a room's temperature is influenced by the energy in a first order lag process [Process, PT1]
4. rooms can be cooled, if they have a cooling actuator [Property, optional]
5. cooling is negative proportional (NP) with the room's energy
6. room energy depends from the adjacents enviroment's temperature proportional to the shared surface A_i and the heat transfer coefficient h_i , hence the dependency is a positive proportional process [Process, PP].

Further analyzing the physics involved bring to the conclusion that

$$Q_{Occ} = q_{lat} n_{occ} \quad (2.6)$$

where q_{lat} is the latent heat gain and n_{occ} the number of occupants in the room. From Equation 2.3 and Equation 2.6 it follows that

7. rooms have a number of occupants [Property, Mandatory]
8. a room's internal energy is positively correlated with the number of occupants [Process, PP].

Additional knowledge can be added if the physics of the system is known; for example, it is possible to further enrich the model by specifying the internal processes

that regulate a AHU. Wang et al. [14] provide a simplified model for modelling the cooling load of an AHU through Equation 2.7

$$Q_{Cool} = \frac{c_a \dot{m}_a^e c_c \dot{m}_c^e}{c_a \dot{m}_a^e + c_c \dot{m}_c^e} (\vartheta_a - \vartheta_c) \quad (2.7)$$

This, given that ϑ_a is the mixed air temperature, \dot{m}_a the air flow rate, ϑ_c the chilled water temperature and \dot{m}_c the chilled water flow rate, leads to the following conclusions

9. cooling is a complex function of the air temperature, the chilled water temperature, the supply air flow and the water flow in the coil [Process, MISO].

Finally, Cui and Wang [3] provide a simple model of a chiller's physics, whose equation can be expressed as

$$\dot{m}_c c_w (\vartheta_{cret} - \vartheta_c) = \eta_{COP} W \quad (2.8)$$

from which it can be derived that

1. the chilled water temperature decreases with the increase of the power consumption [Process, NP]
2. the water flow rate increases as the power consumption of the pump increases [Process, PP].

These dependencies can be semantically mapped to a domain ontology that still use concepts from Brick, SSN and its extended version and Figure 2.5 shows how the discovered domain knowledge can be represented using those ontologies. It is clear that this part is the most costly of the whole process, especially timewise, since it requires very specific expertise. In the example, for instance, it is not captured that a room's internal energy is influenced by the radiant heat of the sun but it would have been possible to do so. The good side of this approach is that the modelling efforts need to be done just once per domain as the ontology is reusable and will adjust itself to the building instance through the automatic reasoning process.

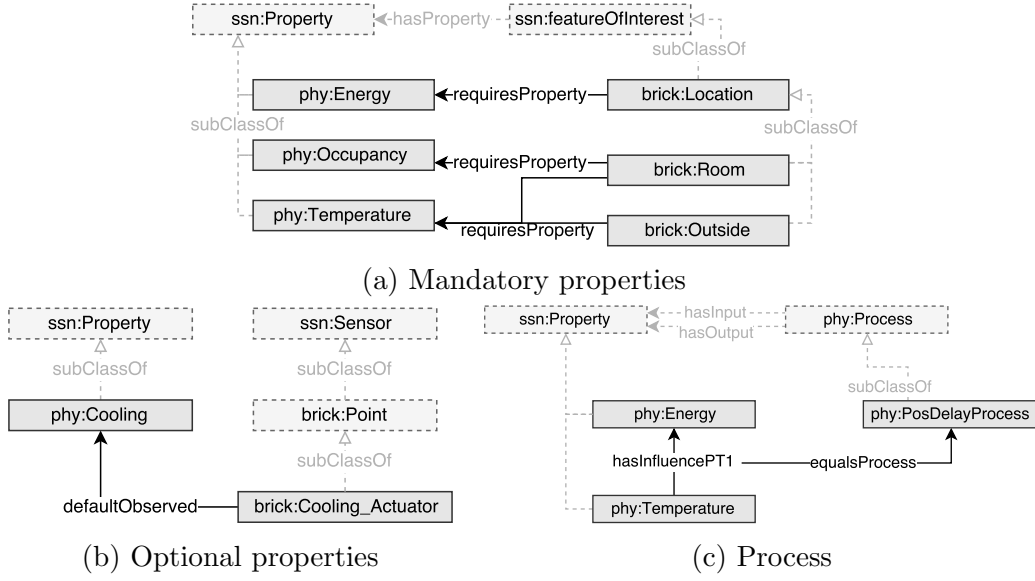


Figure 2.5: Patterns for definition of a model for a system's physics

Reasoning process The reasoning process is divided in two main phases: the discovery of the physical properties and the creation of the physical processes involving those properties. This requires access to the building semantic model created during the semantic mapping phase and the physic model derived by expert for the particular domain (e.g physic model of HVAC system). Both the phases work on the ontology (metadata) to draw conclusions about the instances. The first phase has to be the properties discovery one, since those properties are inputs and outputs for the processes. Based on the ontology derived in the previous paragraph and the semantic model of the building (Figure 2.4), through the application of simple reasoning rules, it is possible to derive the whole semantic graph of both the building and its actual physic behaviour, as seen in Figure 2.6, in a simplified form². The rules to be applied are:

- if there is a room, it has an internal energy, $Room(r) \Rightarrow Energy(e) \wedge hasProperty(r, e)$
- if there is a room, it has a temperature, $Room(r) \Rightarrow Temperature(t) \wedge hasProperty(r, t)$

²Since even the degree of complexity of the semantic graph of small example is too high, the fidelity to the real graph is dropped for clarity's sake.

- if there is a temperature sensor in a room, the room has a temperature and the temperature is observed by the sensor, $Room(r) \wedge Temperature_Sensor(ts) \wedge hasPoint(r, ts) \Rightarrow Temperature(t) \wedge hasProperty(r, t) \wedge observes(s, t)$
- energy influences the temperature through a PT1 process, $Room(r) \wedge Energy(e) \wedge hasProperty(r, e) \wedge Temperature(t) \wedge hasProperty(r, t) \Rightarrow PT1(p) \wedge hasInput(p, e) \wedge hasOutput(p, t)$
- energy of a room depends on the outside temperature if they are adjacent, $Room(r) \wedge Outside(o) \wedge isNextTo(r, o) \wedge Temperature(t) \wedge hasProperty(o, t) \wedge Energy(e) \wedge hasProperty(r, e) \Rightarrow PP(p) \wedge hasInput(p, t) \wedge hasOutput(p, e)$
- if a AHU has a coil water flow sensor then it has a coil water flow property, $AHU(a) \wedge CoilWaterFlowSensor(s) \wedge hasPoint(a, s) \Rightarrow CoilWaterFlow(f) \wedge hasProperty(a, f) \wedge observes(s, f)$.

The reasoning rules can be completed for the remaining AHU's components reusing similar patterns as those shown in the list above. These inference rules are explicitly written as implications among instances (no ontological reasoning is involved) because it is easier to convey the concept this way and, since this example is really small, it was easy and fast enough to be done by hand. Even a slightly more complex example would have required a more refined reasoning scheme involving the concepts, the instances and their mutual relationships. However it is already possible to get a grasp of the portability of this approach just noticing that, just with these simple rules, it does not matter how many instances of rooms, AHUs or chillers there can be. The reasoner would still enforce the correct properties in an automatic fashion. Looking back at the examples in this chapter, it can be noticed that, even in a small example, there are many informations encoded in the graph that starts to grow in size and complexity. Therefore it starts to surface one of the challenges of this approach, addressed in chapter 4, that is the scalability issue for big commercial buildings.

Chapter 3

Diagnosis Algorithm

In this chapter will be outlined the diagnosis algorithm[8] developed by IBM Research with the collaboration of Michael Maghella, a fellow student at Università degli Studi di Brescia. This algorithm is based on a novel reputation-based approach that makes use of the informations available in the semantic graph to identify cause-effect relationships and use these relationships to isolate relevant causes basing the diagnosis on the timeseries data.

3.1 General description

The diagnosis activity is comprised of three main step:

- fault detection, that is the discovery of the fault. A diagnoser needs to be able to tell apart the normal behaviours from faulty ones. This is done, in this approach, through the rules discovered while learning the normal behaviour of a building (see subsection 2.2.3). Whenever a fault is detected the diagnosis process ought to start.
- fault isolation, that is the discovery of the causes of the fault. It is a key step for enabling a precise diagnosis of the problem. In this approach informations derived from the semantic graph are used to narrow down the set of possible faults to the most relevant ones.
- fault identification, that is to determine with some confidence which are

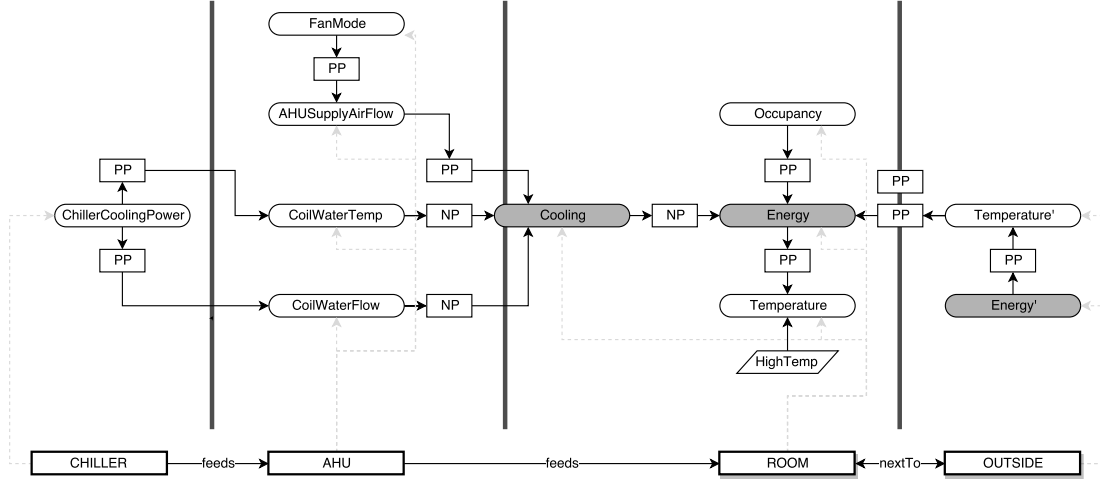


Figure 3.1: Internalized model of the example building

the actual causes of a fault among all the possible causes. The proposed diagnoser implements a voting scheme to identify these causes.

While explaining the diagnosis approach, it will be used the same model presented in chapter 2 (Figure 2.6), that will be represented in a compact notation shown in Figure 3.1, where every component is an instance of a concept with the same name (internalized representation). Moreover, for the sake of simplicity, the physic model of the system is simplified to a more lax one, dropping the MISO processes and the PT1 ones in favor of the PP and NP ones. The diagnoser is run every time an anomaly is detected in the timeseries. An anomaly is intended as a deviation from the normal behaviour clearly detectable by some preset upper and lower bounds, therefore an anomaly can be classified as high or low. It is assumed that an anomaly is symptom of some kind of fault in the system. The algorithm starts, as soon as the anomaly is detected, by identifying the semantic type of the anomaly and the potential causes in the graph. It proceeds computing the voting vector and, eventually, it is recursively executed for all those properties that get a voting score equals or greater then one. When no other causes with a high score are left, the algorithm terminates and produces a list of the detected causes along with the name of the faulty properties, the semantic type of the faults and the estimation, expressed in percentage, of the responsability of those causes towards the diagnosed anomaly. The relative pseudocode is in Algorithm 1.

Algorithm 1 General diagnosis algorithm

Require:

the anomaly A ,
the semantic graph \mathcal{G}

Ensure: a set C_A of possible causes of the anomaly

```
1: procedure DIAGNOSEANOMALY( $A, \mathcal{G}$ )
2:    $T \leftarrow \text{GETSEMANTICTYPEOFANOMALY}(A)$ 
3:    $PC_A \leftarrow \text{GETPOSSIBLECAUSESINGRAPH}(A, T, \mathcal{G})$ 
4:    $C_A \leftarrow \emptyset$ 
5:    $\mathbf{v} \leftarrow \text{COMPUTEREPUTATIONVOTEVECTOR}(A, T, PC_A)$ 
6:   for all  $p \in PC_A$  do
7:     if  $\mathbf{v}[p] \geq 1$  then
8:        $C_p \leftarrow \text{DIAGNOSEANOMALY}(p, \mathcal{G})$  ▷ diagnose subtree
9:       if  $C_p \neq \emptyset$  then
10:         $C_A \leftarrow C_A \cup C_p$ 
11:       else
12:         $C_A \leftarrow C_A \cup \{p\}$  ▷  $p$  is a root cause
13:       end if
14:     end if
15:   end for
16: end procedure
```

3.2 Potential causes

The first non-trivial operation of the algorithm is the identification of the possible causes of a given an anomaly. A potential cause is an observation made by a sensor that observe a property and such that the following hold:

1. the property observed by the sensor is an input of a process that directly output to the anomaly, or it is connected to the anomalous property by a chain of unobservable properties
2. it is observable.

The properties and their relationships with the anomaly are all encoded in the semantic graph and are therefore accessible via reasoning. The approach is also interested in the semantic type of the anomaly and of the possible causes. Retrieving the possible causes of an anomaly is possible because in the graph, anomalies are observations made by a sensor that measures a property. Figure 3.2 shows the possible ways to retrieve a potential cause of a property. In both figures the situation respect the conditions given earlier in the chapter regarding potential causes. The causes retrieved at this point are the ones that are correlated at various degrees to the anomaly and are the only ones that can help to produce a diagnosis. At this point it is important to preserve the semantic informations about the processes involved since this information is at the base of the voting algorith. When talking about “semantic informations” of the process it is meant the replacement of a chain of properties with a single, direct process whose type is infered from the ones of the chain. For example it can be seen in Figure 3.3 that the temperature in a room is influenced by the occupancy property of the room through a chain of a PT1 process and a PP process, both involving the unobservable energy property. This means that the temperature is influenced by the occupancy through a direct PT1 process. Table 3.1 shows the possible combinations for the most common type of processes usedm that are the positive correlation, the negative correlation and the lag processes. Given these new processes, that directly correlate an anomaly to its potential causes, it is possible to infer the entity of the potential cause. This information is computed thanks to the intuition that the type of process correlating two observation carries qualitative information about the output. It is possible, for

Table 3.2: Predicted value of an observation based on the process it depends on and the input to such process

Anomaly type	Process type	Cause type
High	Positive correlated	High
High	Negative correlated	Low
Low	Positive correlated	High
Low	Negative correlated	Low
Any	Lag	Lag

example, predict that an observation that is influenced by a positive correlation process whose input is an observation that is high is high itself. The ontology defines the combinations listed in Table 3.2. Thanks to those informations it is possible to implement the fault identification capabilities of the algorithm.

3.3 Fault identification

The whole approach is based on the intuitive assumption that an anomaly is caused by faults and that the measurement of the behaviour of such faults needs to be anomalous too. Moreover this anomalous behaviour needs to be semantically coherent with the behaviour predicted from the chain of process that links the anomaly to the cause. These two principles are at the base of the voting algorithm. The algorithm works by computing, from the set of anomaly-free time series, a world view (that is a subset of the historical data for each time series) where the property that presents an anomaly would be anomaly free but close in range to the anomalous value. This can be done by selecting all the anomaly-free data of the sensor that observed the fault; these values are splitted in deciles and the decile closest to the anomalous value is chosen, that is the first decile if the anomaly is of type Low or the tenth if it is of type High. For every potential cause are then extracted the samples measured at the same timestamp as the samples of the anomalous property that fall in the chosen decile. given this subset $S' \subset S$ each potential cause will derive a second subset $S''_p \subset S'$ which represent the view of the system from the point of view of one of the potential causes, that provide its point of view assuming it is not anomalous itself. The method is the same

Algorithm 2 Voting algorithm

Require:

- the anomaly A ,
- a set of possible causes of A , PC_A ,
- the semantic type of the anomaly, T ,

Ensure: the vector of votes, \mathbf{v}

```

1: procedure COMPUTEREPUTATIONVOTEVECTOR( $A, T, PC_A$ )
2:    $S \leftarrow \text{GETHISTORICALTIMESERIES}$ 
3:    $S' \leftarrow \text{GETSIMILARHISTORICALDATA}(S, A)$ 
4:    $V \leftarrow \emptyset$  ▷ the voting matrix
5:   for all  $p \in PC_A$  do
6:      $S'' \leftarrow \text{GETSIMILARHISTORICALDATA}(S', p)$ 
7:     for all  $c \in PC_A$  do
8:        $d_{p,c} \leftarrow \text{CALCULATEDISTANCE}(S'', c)$ 
9:        $v_{p,c} \leftarrow \text{CALCULATEDISTANCE}(S'', c)$ 
10:    end for
11:     $\mathbf{r} \leftarrow \text{CALCULATEREPUTATION}(\mathbf{V})$ 
12:     $\mathbf{v} \leftarrow \text{WEIGHTVOTES}(\mathbf{V}, \mathbf{r})$ 
13:  end for
14: end procedure
  
```

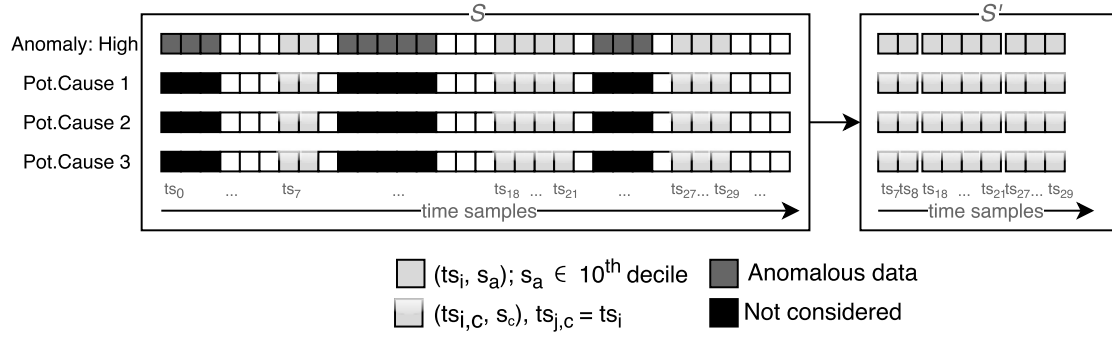


Figure 3.4: Creation of the set S'

described for the computation of S' , using the value observed for that property at the time of the anomaly instead of the one of the anomaly itself. At this point every potential cause p has the informations needed to give the opinion, that is to vote, about which is the actual cause c of the fault, according to its perceived state of the world S''_p . The voting is expressed in terms of normalized mean distance $d_{p,c}$ between the actual value observed by the suspect property at the time t_0 of occurrence of the anomaly and the mean of the samples values that represent the behaviour of the potential cause expected by the voting variable.

$$d_{p,c} = 10 \frac{s_c(t_0) - \text{mean } S''_{p,c}}{\max S_c - \min S_c} \quad (3.1)$$

Normalization occurs in a range of $[-10,10]$ with the maximum and minimum value of the whole time series in S (anomalous values included). This steps ensure that the different measurements can be comparable. The resulting distance is less then zero if the actual value observed for a property is lower than the value the voting property is expecting to see, it is greater than zero otherwise. This information about the signedness of the computed distance plays a key role during the fault identification because it helps discriminating between anomalous values that are the cause of a fault from those anomalous values that are effect of a fault. That means that if the behaviour of a property doesn't match the behaviour that a cause should have, said property can't be a cause and it's vote is set to zero.

$$v_{p,c} = \begin{cases} d_{p,c} & \text{if } d_{p,c} \geq 0 \wedge \text{type } c \equiv \textit{High} \\ -d_{p,c} & \text{if } d_{p,c} \leq 0 \wedge \text{type } c \equiv \textit{Low} \\ \|d_{p,c}\| & \text{if } \text{type } c \equiv \textit{Unknown} \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

For example, if a room's temperature is too high and the real faulty property is the outside temperature, there's a chance that the chiller power consumption will be anomalous too since the chiller is trying to mitigate the effect of the fault. Still the chiller is not a cause of the anomaly. Once the computation of the distances ends, the matrix of the votes have been filled and contains the opinions that the

potential causes have of each others. Since the votes have been calculated under the assumptions that the potential cause that is voting is under normal circumstances, if said cause is the origin of the fault then it will flag as faulty all the others, since its point of view will be so distorted that normal behaviours appears as anomalous. This tendency of the faulty property is misleading and thus need to be addressed. The way this algorithm deal with the problem is through the means of the concept of reputation. A potential cause has a high reputation r_p , hence it is trustworthy, if it doesn't receive any vote from other causes. The reputation decrease as the porperty receive more votes, as seen in Equation 3.3.

$$r_p = \frac{1}{1 + \sum_{c \in PC_A} v_{p,c}} \quad (3.3)$$

Given the reputations r_c of the potential causes, the total vote v_p received by a potential cause p is given by the sum of the vote of all the properties weighted by the reputation of the property that gave that vote (Equation 3.4). It is considered abnormal every property that has a value higher then 1.

$$v_p = \sum_{c \in PC_A} r_c \cdot v_{p,c} \quad (3.4)$$

An alternative kind of vote is the so called vote count. This method drop the magnitude of the votes and simply count how many votes greater than one have been assigned to a given property p (Equation 3.5).

$$\begin{aligned} \bar{v}_p &= \sum_{c \in PC_A} \text{discretize}(r_c \cdot v_{p,c}) \\ \text{discretize } x &= \begin{cases} 1 & \text{if } x \geq 1 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (3.5)$$

The formulae result in the vote vector $\mathbf{v} = \{v_p | p \in PC_A\}$ and $\bar{\mathbf{v}} = \{\bar{v}_p | p \in PC_A\}$

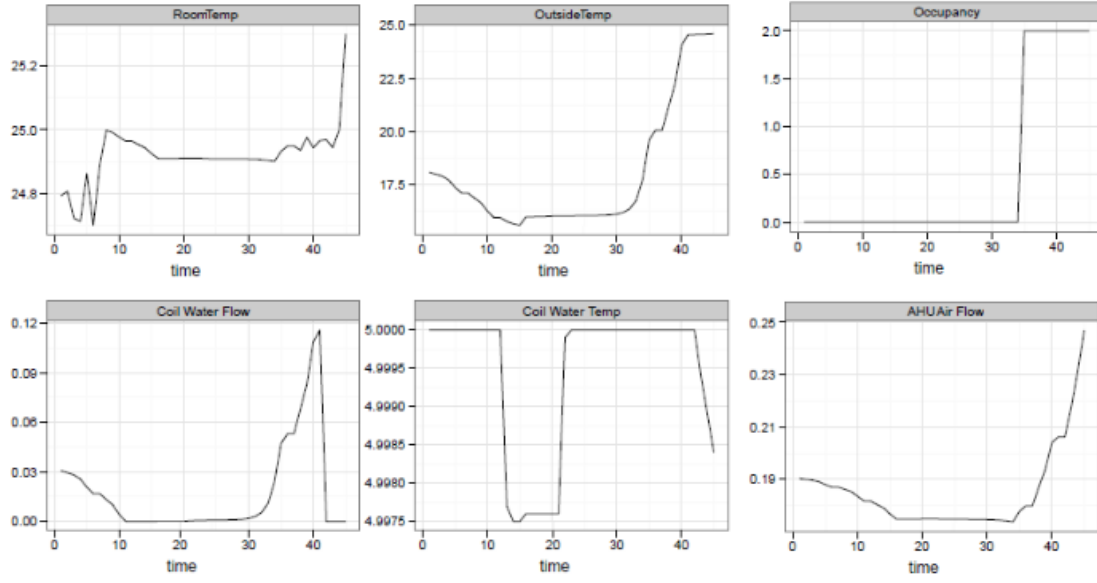


Figure 3.5: Example of timeseries of different sensors

3.4 Diagnosis example

In this section it is shown how the diagnosis take place in the context of a building as in Figure 3.1. It is assumed that the data picked up by the sensors are as in Figure 3.5, the rule that monitor the anomaly is “if the room’s temperature is greater than 25° there’s an anomaly” and the fault is due to the chiller that stopped to work at 1pm. The room temperature start rising and a first anomaly is detected, thus starting the diagnosis process. Since the anomaly is a temperature greater than 25° , the semantic type of the anomaly is High. Then the possible causes can be traced back, following the rules explained above. The backtracked causes and the derived processes are shown in bold in Figure 3.6 as the potential causes are known, it is possible to predict the semantic value of said causes. According to Table 3.2 seen earlier in this chapter it is derived the graph shown in Figure 3.7.

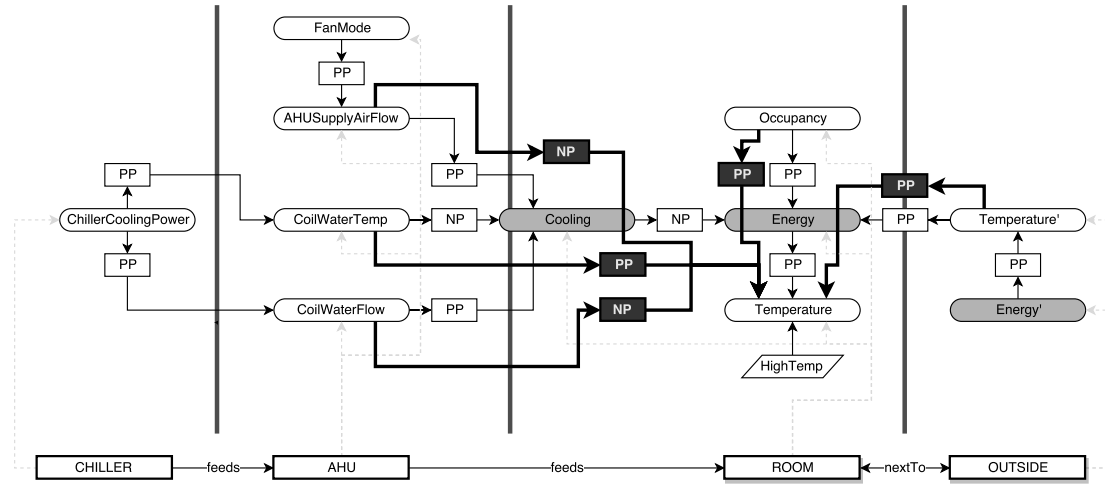


Figure 3.6: Backtracking the possible causes of a HighTemp anomaly

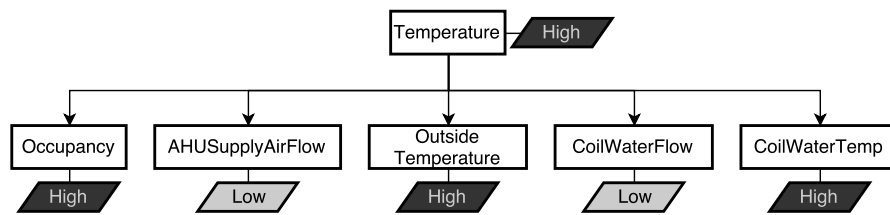


Figure 3.7: Semantic type of potential causes

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

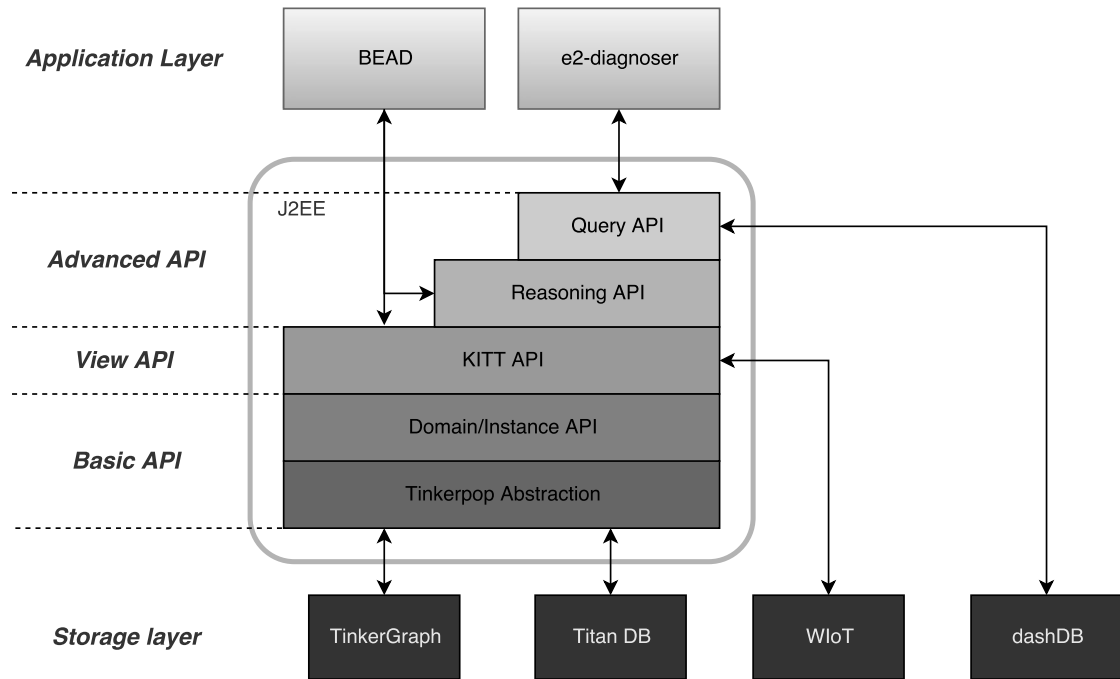


Figure 4.1: IoT BRAIN architecture

efficient storage of time series and high query performance. On top of this layer sits the core of the project which realize the theoretic 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] Jingtian Cui and Shengwei Wang. “A model-based online fault detection and diagnosis strategy for centrifugal chiller systems”. In: *International Journal of Thermal Sciences* 44.10 (2005), pp. 986–999. ISSN: 1290-0729.
- [4] 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.
- [5] Joern Ploennigs and Anika Schumann. “From Semantic Models to Cognitive Buildings”. In: *AAAI*. 2017.
- [6] Joern Ploennigs et al. “Demo Abstract: BEAD - Building Energy Asset Discovery Tool for Automating Smart Building Analytics”. In: (Nov. 2014).
- [7] Joern Ploennigs et al. “e2-Diagnoser: A System for Monitoring, Forecasting and Diagnosing Energy Usage”. In: 2015 (Jan. 2015), pp. 1231–1234.
- [8] Joern Ploennigs et al. “Semantic Diagnosis Approach for Buildings”. In: *PP* (July 2017), pp. 1–1.
- [9] *Project Haystack*. URL: <https://project-haystack.org/>.
- [10] *Resource Description Framework*. URL: <http://www.w3.org/RDF>.

- [11] Ioana Robu, Valentin Robu, and Benoit Thirion. “An introduction to the Semantic Web for health sciences librarians”. In: 94 (May 2006), pp. 198–205.
- [12] Anika Schumann, Joern Ploennigs, and Bernard Gorman. “Towards Automating the Deployment of Energy Saving Approaches in Buildings”. In: (Nov. 2014).
- [13] *SPARQL Protocol and RDF Query Language*. URL: <https://www.w3.org/TR/rdf-sparql-query>.
- [14] Yao-Wen Wang et al. “A simplified modeling of cooling coils for control and optimization of HVAC systems”. In: 45 (Nov. 2004), pp. 2915–2930.
- [15] Michael Wetter. “Multizone Building Model for Thermal Building Simulation in Modelica”. In: 2 (Jan. 2006).