Adapting Semantic Sensor Networks for Smart Building Diagnosis

Joern Ploennigs, Anika Schumann, and Freddy Lécué

IBM Research

Abstract. The Internet of Things is one of the next big changes in which devices, objects, and sensors are getting linked to the semantic web. However, the increasing availability of generated data leads to new integration problems. In this paper we present an architecture and approach that illustrates how semantic sensor networks, semantic web technologies, and reasoning can help in real-world applications to automatically derive complex models for analytics tasks such as prediction and diagnostics. We demonstrate our approach for buildings and their numerous connected sensors and show how our semantic framework allows us to detect and diagnose abnormal building behavior. This can lead to not only an increase of occupant well-being but also to a reduction of energy use. Given that buildings consume 40 % of the world's energy use we therefore also make a contribution towards global sustainability. The experimental evaluation shows the benefits of our approach for buildings at IBM's Technology Campus in Dublin.

1 Introduction

With the development of embedded cyber physical systems and large computational resources in the cloud, the availability of sensor information continuously increases towards the Internet of Things. Semantic Sensor Networks (SSN) play an important role in this development as they provide a homogeneous semantic layer for sensor information and simplify the detection and retrieval of data [1,2]. This is key for connecting cloud-based analytic tasks that process this data. However, advanced analytics need also knowledge of the system's internal processes. Diagnostics, for example, require hypotheses of the cause-effect relationships between sensors. SSN do not provide ways to model such aspects and it is often necessary to add this information manually [3,4]. However, this can turn into a very tedious task and prevent the exploitation of SSN for large scale analytic applications.

Building automation systems (BAS) are one established example of large scale sensor and control networks that contain thousands of devices in newer buildings. Analyzing the data allows to improve the building's energy consumption with large environmental impact as buildings consume about 40 % of the energy in industrialized countries and are responsible for 36 % of their CO_2 emissions [5]. For large companies building energy management is done at enterprise scale that enables them to monitor, analyze, report and reduce energy consumption across their building portfolio, including retail and office properties. However, the integration of thousands of sensors of buildings in different locations, with different systems and technologies is a challenging task.

This paper presents the architecture and approach of IBM's Semantic Smart Building Diagnoser. It allows to automatically derive complex analytic tasks in buildings from a semantic sensor network model. With some small extensions to the SSN ontology we are able to derive large models of the physical processes within the building using solely semantics techniques such as SPARQL update.

The following section reviews the state of the art. Section 3 explains the architecture of our approach that is detailed in Section 4. We present the evaluation of our approach at IBM's Technology Campus in Dublin in Section 5. The paper concludes with remarks on the lessons we learned.

2 State of the Art

The building automation domain has a long history with interoperability problems due to the diversity of systems and technologies [6]. The trend is to use semantic web technologies to describe sensors and the observed context as states and events [3,6,7]. Similar capabilities are provided by the domain independent W3C Semantic Sensor Network (SSN) ontology [2], which will be explained in Sec. 4.1. The benefit of these ontologies is that they provide a homogeneous semantic model of sensors to backend systems that then can be agnostic of the underlying technology. However, the approaches model sensors only as interfaces of the system and do not describe the processes within the system. This information is important for many in-depth analytic tasks such as diagnosis.

An estimated 15 % to 30 % of energy in buildings could be saved if faults in the BAS system and its operation could be detected in a timely manner [8]. Katipamula and Brambley [9] provide an extensive review of the common approaches. They classified them in three categories: physical model based approaches, data driven approaches, and rule-based approaches. The first require physical models of the building components like boilers or chillers. These approaches can diagnose faults precisely. However, the model development requires time and expertise and the models are difficult to adapt to different buildings. Data driven approaches like [10] are solely based on the building's sensor data. While this makes them easily adaptable to different buildings their diagnostic capability is limited to the detection of faults rather than their identification. Therefore, rule-based approaches are most established and also used by IBM's TRIRIGA Environmental and Energy Management Software. The rules capture domain knowledge of conditions for anomalies and their cause-effect relationships [11]. It was shown in [3,4] that the rules can also be executed directly on the semantic model. This leads to a more integrated approach that simplifies the deployment. However, it still requires data modeling engineers to create the diagnostic model. Herein also lies its biggest limitation: as rules can detect predefined situations only, they need to be manually adapted to each system by people with in-depth knowledge of all potential cause-effect relationships.

Our approach combines the benefits of these three concepts to overcome their disadvantages: it models high level physical processes in the systems to derive diagnosis rules; it parameterizes the rules using data analytics and applies them effectively during runtime. The strength of a semantic approach is that we can use reasoning and semantic web techniques to automate this process for the large scale sensor networks found in buildings such that the human modeling and calibration effort is minimal.

3 Architecture

Figure 1 shows the architecture of the Semantic Smart Building Diagnoser. It is designed to allow energy management of a global portfolio of buildings. The individual building's BAS are integrated via REST services that are usually available and allow retrieving time series information and non-semantic text labels for sensors. This data is enriched with additional semantic information in different steps such that we can autonomously run different analytic processes on the top layers that are agnostic of the underlying BAS. The process is divided into an initialization phase, that creates the semantic model once, and a runtime phase, that uses the semantic model to efficiently process online stream data.

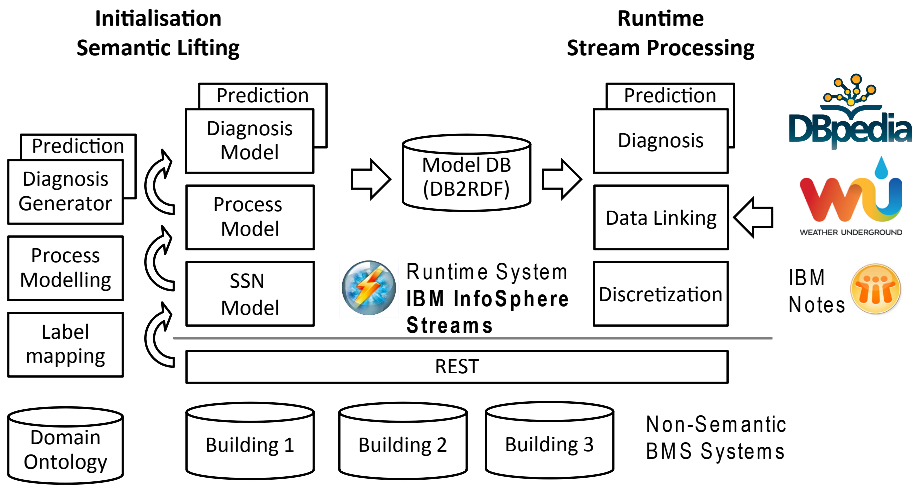


Fig. 1: Architecture of the approach

During the *initialization* phase we lift the usually non-semantic data to a semantic representation in a domain ontology that specifies common semantic types and physical processes in buildings as extension of the W3C SSN ontology (Sec. 4.1). The lifting is composed of three steps. First, we create a *SSN model* that describes the available sensors. The step is semi-automated by an internal label mapping tool that maps the non-semantic BAS text labels to their semantic equivalents in the domain ontology and asks a human for validation [12].

From this homogeneous semantic representation we then automatically derive a *physical process model* that expands the relationships between sensors (Sec. 4.2). It allows us to automate individual analytic tasks based on sensor data from the building by extending specific semantic information. For example, the diagnosis model generator extends cause-effect relationships between sensors (Sec. 4.3). The different semantic models are stored in a DB2RDF database for efficient access during runtime.

The *runtime* environment is using the semantic version of IBM InfoSphere Streams [13], which handles mixed time series and RDF data based stream processing/reasoning in real-time. This hybrid setup is important as not all time series data can be rendered in RDF for scalability reasons (Sec. 4.4). We enrich the dataset with external information by linking, for example, additional explanations of semantic types to DBpedia. Historical weather data and forecasts from wunderground are used for diagnosis and energy predictions [14]. Room booking information from IBM Notes is used to estimate and predict room occupancy. We use this semantically-enriched dataset for the different analytic tasks (Sec. 4.4). The individual steps of our approach are detailed in the following section.

4 Approach

We start by introducing a necessary extension to the SSN ontology that will enable us to automatically derive the physical process model and a diagnostic model.

4.1 Extended SSN ontology

In the first step, we create a SSN representation for each BAS using an internal label mapping tool [12]. The tool semi-automatically assigns the corresponding semantic sensor type in our domain ontology to each BAS label using common structure, acronyms, and units in the textual descriptions (e. g. we extract "Temperature Sensor in Room R1" from the label "R1_Tmp").

The resulting model is an extension of the Semantic Sensor Network (SSN) skeleton ontology defined by the W3C incubator group [2]. SSN consolidates concepts from various sensor network ontologies and was chosen because it provides differentiated views on the aspects of sensing beyond the building domain. In particular it uses the Stimulus-Sensor-Observation ontology design pattern [15]. The pattern separates the concepts of *ssn:Sensor* for physical devices taking measurements in form of *ssn:Observation* and the actual changes that happen in the environment, which is the *ssn:Stimulus* of the measurement. This separation is important as it recognizes that: 1) stimuli occur in the environment independently of the number and kind of sensors that observe them and 2) observations made by a sensor are not identical to stimuli as measurements can be incorrect due to measurement noise, outliers, or sensor failures. We use the following classes from the skeleton ontology:

- *ssn:FeatureOfInterest*: The monitored (diagnosed) system is defined as a FeatureOfInterest. In the following we will use the shorter feature as synonym.

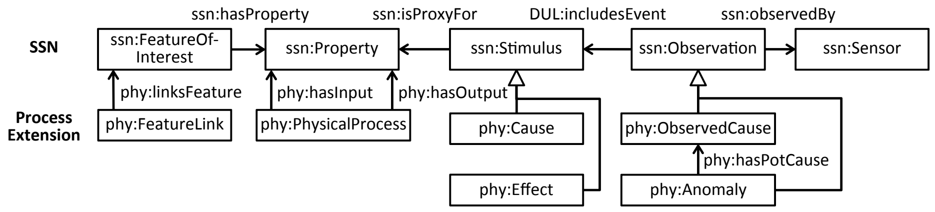


Fig. 2: Extension to the SSN ontology

- *ssn:Property*: Properties are qualities of a Feature. They can be observed like the air temperature of a room. In extension to the SSN, we define that properties may also be unobservable. Unobservable properties are often relevant to correctly understand the physical processes within features. One example is the inner energy of the room. The inner energy is the heat stored in the air in the room. It is related to the air temperature, but not measurable.
- *ssn:Sensor*: A time series provided by a physical device or computational method observing a property.
- *ssn:Observation*: Observations describe discrete states that are derived from the sensor time series data. An observation state may be that an air temperature is above 20 °C.
- *ssn:Stimulus*: A stimulus is an event or state of a property. A stimulus is not identical with the observation made by a sensor, as the sensor itself may fail.

The SSN ontology provides no means to model physical and cause-effect relationships between sensors. Therefore, we have extended the SSN ontology in [16] using the namespace *phy* by the concepts[1] shown in Figure 2:

- *phy:PhysicalProcess*[2]: The properties of many real world features are related by physical processes. A *phy:PhysicalProcess* models this as directed relationship of a source property (*phy:hasInput*) that influences a target property (*phy:hasOutput*). We differentiate different process types that relate to common models used in control theory[3]. In the paper we will only use positive and negative correlated properties. The influence is a *phy:PosCorrProc* if a factor increases with its influence and it is a *phy:NegCorrProc* if it decreases with an increase of the influence. The temperature we feel, for example, is influenced by the inner energy in a room via a positive correlation process, as it feels warmer when we increase the energy. The cooling system uses a negative correlation process as it removes heat from this energy.
- *phy:FeatureLink*: Physical processes occur not only between properties within the same feature, but, also between related features. A FeatureLink denotes such a relationship between features that are defined by the object

[1] A complete version of the provided example and an extended one is available at *https://www.dropbox.com/s/z369tzmn00f1jv9/demopackage.zip*.

[2] Not to be confused with *ssn:Process* or *dul:Process* from DOLCE subset used by SSN. The first describes the sensing process and the second an event in transition.

[3] We use classes of linear time-invariant systems such as proportional, integral, derivative, or delay processes. This is similar to data flow models such as Matlab/Simulink.

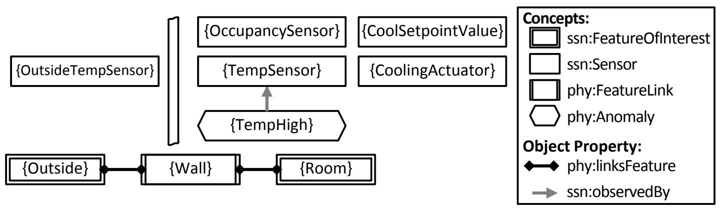


Fig. 3: Example input of the approach

property *phy*:*linksFeature*. They are, for instance, used to model spatial relationships between two adjacent rooms connected by a wall.

– *phy*:*Cause*, *phy*:*Effect*: are subconcepts of *ssn*:*Stimulus* and describe the not necessarily observed stimulus of a cause and the resulting effect.
– *phy*:*Anomaly*: is a subconcept of *ssn*:*Observation* that is used to describe abnormal observations that should be diagnosed. An anomaly may be for example a high room temperature.
– *phy*:*ObservedCause*: is another subconcept of *ssn*:*Observation* describing the observable discrete states of potential causes of an anomaly. A cause for a high temperature in a room may be an inactive cooling system.

We will later introduce additional object and annotation properties to model the generic process knowledge in a domain ontology. In the following example we use the namespace *sb* for this smart building domain ontology.

Example 1 ***(Semantic Input)***
We illustrate the approach with the running example of a single office room $\{Room\}$[4] shown in Figure 3. It is separated from to the outside by a wall. The room contain sensors for air temperature $\{TempSensor\}$ and occupancy $\{OccupancySensor\}$. A virtual sensor $\{OutsideTempSensor\}$ links to the outside temperature retrieved from wunderground.com. The room also contains a cooling system with actuator $\{CoolingActuator\}$ and setpoint $\{CoolingSetpointValue\}$. The setpoint value is automatically decreased if the room is occupied to save cooling energy if it is unoccupied.

We map the sensors in the example from Figure 3 to our domain ontology. It contains concepts for the sensors *sb*:*TemperatureSensor*, *sb*:*OccupancySensor*, *sb*:*CoolingActuatorValue*, and *sb*:*CoolingSetpointValue* $\sqsubseteq$ *ssn*:*Sensor*. The sensors observe the corresponding room properties *sb*:*Temperature*, *sb*:*Occupancy*, *sb*:*Cooling*, *sb*:*CoolingSetpoint* $\sqsubseteq$ *ssn*:*Property*. In addition we define *sb*:*Energy* $\sqsubseteq$ *ssn*:*Property* for the unobservable property of a rooms inner energy. The rooms are instances of *sb*:*Room* and the environment of *sb*:*Outside* which are both features, thus *sb*:*Room*, *sb*:*Outside* $\sqsubseteq$ *ssn*:*FeatureOfInterest*.The locations are connected by *sb*:*Wall* $\sqsubseteq$ *ssn*:*FeatureLink*.

Our aim is to determine which of the sensors, related to the room or its surrounding, can localize the cause of an abnormally high temperature $\{TempHigh\}$. Here we consider any sensor value abnormal that deviates by more than two degrees from the cooling set point of that room.

[4] We denote by $\{c\} \sqsubseteq C$ an instance c, internalized as a concept $\{c\}$ which is a specialization of C.

4.2 Deriving the Process Model

The SSN is automatically extended by the properties and processes within the system. We do this in two steps. First, we create property instances for all feature instances, even if they are not observed as they may be involved in physical processes. Second, we connect these property instances by processes. The necessary knowledge about the properties and processes is modeled on concept level in the domain ontology using annotation properties. As the same domain ontology is used by all buildings, this modeling effort needs to be done only once. For example, our smart building ontology contains concepts for 62 sensor and 18 properties as well as 53 process annotations (see footnote 1). This will enable us in Sec. 5.1 to create several thousand property and process instances for the IBM Technology Campus Dublin.

We differentiate mandatory and optional properties. Mandatory properties are characteristic physical properties of a FeatureOfInterest and need to be created for each feature instance. Optional properties are not explicitly required by a feature and are only created if they are observed by a sensor. We define these relationships on concept level of the domain ontology using annotation properties. For example, we may not find a temperature sensor in each room, but, each room has a temperature. Thus, we annotate the *sb*:*Room* class using the annotation *phy*:*requiresProperty* to link to *sb*:*Temperature* as a mandatory property. The cooling actuator on the other hand is not mandatory to a room as not all rooms have a cooling unit. We annotate the sensor subclass *sb*:*CoolingActuator* using the annotation *phy*:*defaultObserved* referring to the *sb*:*Cooling* property. It specifies that a room only possesses a Cooling property if it also has a CoolingActuator as sensor.

We use these annotation properties to create the property instances using SPARQL 1.1 update (SPARUL). SPARUL is an extension of SPARQL that does not only allow the search for specific RDF patterns, but also allows modifying and extending the RDF graph around such patterns. Figure 4 shows on the top left and top right two SPARUL queries that create mandatory and optional properties, respectively. The SPARUL blocks consist of a modification pattern (INSERT) that is executed for each match of a search pattern (WHERE). With the top left query #1 we search for instances of a feature class annotated by *phy*:*requiresProperty* that refers to a mandatory property. For each instance found we create an instance of the property using an unique URI that is computed by the function *UURI*[5]. The SPARUL query #2 for optional properties works similarly and creates properties for these features that have a sensor of a class with the *phy*:*defaultObserved* annotation.

In the second step, we connect the created properties by process instances. We use again annotation properties to model the generic relationships on concept level of the domain ontology and then use SPARUL to extend the specific SSN instances. Please note, that physical processes may exist between properties of the same feature (e. g. cooling reduces internal energy) as well as between properties of different features (e. g. two adjacent rooms exchange energy). Therefore,

[5] The function *UURI* needs to compute the same unique URI for identical inputs such that mandatory and optional properties extend each other.

```
INSERT {    #1: Create mandatory properties
  ?newuri1  rdf:type               ?propCls.
  ?newuri1  ssn:isPropertyOf       ?feat.
  ?feat     ssn:hasProperty        ?newuri1.
} WHERE {
  ?feat     a                      ?featCls.
  ?featCls  phy:requiresProperty   ?propCls.
  BIND(UURI(?feat,?propCls) as ?newuri1).
}
```

```
INSERT {        #2: Create optional properties
  ?newuri2  rdf:type               ?propCls.
  ?newuri2  ssn:isPropertyOf       ?feat.
  ?feat     ssn:hasProperty        ?newuri2.
  ?sensor   ssn:forProperty        ?newuri2.
} WHERE {
  ?sensor   a                      ?sensCls.
  ?sensCls  phy:defaultObserved    ?propCls.
  ?sensor   ssn:ofFeature          ?feat.
  BIND(UURI(?feat,?propCls) as ?newuri2).
}
```

```
INSERT {        #3: Create internal processes
  ?newuri3  rdf:type               ?proc.
  ?newuri3  phy:hasInput           ?prop2.
  ?newuri3  phy:hasOutput          ?prop1.
} WHERE {
  ?anno     rdfs:subPropertyOf     phy:hasIntInfl.
  ?anno     phy:equalsProcess      proc.
  ?propCls1 ?anno                  ?propCls2.
  ?prop1    a                      ?propCls1.
  ?prop1    phy:isPropertyOf       ?feat1.
  ?feat1    phy:hasProperty        ?prop2.
  ?prop2    a                      ?propCls2.
  BIND(UURI(?prop1,?prop2) as ?newuri3).
}
```

```
INSERT {        #4: Create external processes
  ?newuri4  rdf:type               ?proc.
  ?newuri4  phy:hasInput           ?prop2.
  ?newuri4  phy:hasOutput          ?prop1.
} WHERE {
  ?anno     rdfs:subPropertyOf     phy:hasExtInfl.
  ?anno     phy:equalsProcess      proc.
  ?propCls1 ?anno                  ?propCls2.
  ?prop1    a                      ?propCls1.
  ?prop1    phy:isPropertyOf       ?feat1.
  ?featL    phy:linksFeature       ?feat1.
  ?featL    phy:linksFeature       ?feat2.
  ?feat2    phy:hasProperty        ?prop2.
  ?prop2    a                      ?propCls2.
  BIND(UURI(?prop1,?prop2) as ?newuri4).
}
```

Fig. 4: SPARUL code to create the process model.

we use two different annotation patterns to: i) describe internal process relationships of properties within the same feature and ii) external process relationships connecting properties of linked features. An internal process relationship is detected by the SPARUL query #3 in Figure 4. It searches for two properties of the same feature, whose classes are linked by an annotation property ?*anno* that is a subproperty of *phy*:*hasIntInfl*. For each match it creates a physical process of type ?*proc*. The type is specified by a property *phy*:*equalsProcess* of the annotation ?*anno*. The SPARUL query #4 in Figure 4 uses a similar pattern, but, searches for *phy*:*hasExtInfl* annotations as well as for properties of different features that are linked by the *phy*:*linksFeature* property of a feature link.

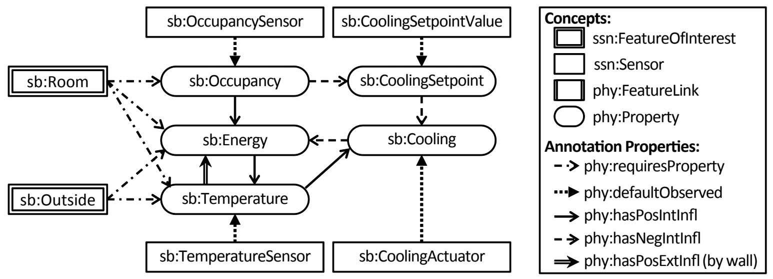
Example 2 ***(Process Model Creation)***
Let us consider that the following triples are defined in our domain ontology

sb:*Room*	*phy*:*requiresProperty*	*sb*:*Energy*.
sb:*CoolingActuator*	*phy*:*defaultObserved*	*sb*:*Cooling*.
phy:*hasNegIntInfl*	*rdfs*:*subPropertyOf*	*sb*:*hasIntInfl*.
phy:*hasNegIntInfl*	*phy*:*equalsProcess*	*phy*:*PosCorrProc*.
sb:*Energy*	*phy*:*hasNegIntInfl*	*sb*:*Cooling*.

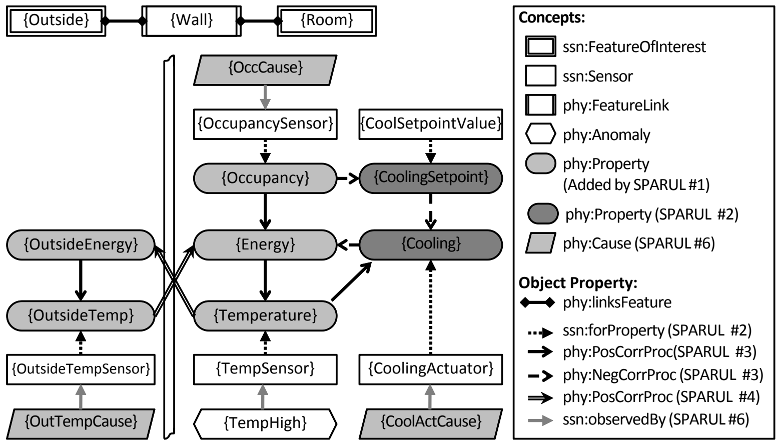
and the following part from the SSN ontology of example 1 in RDF N3 syntax

:*Room*	*a*	*sb*:*Room*.
:*CoolingActuator*	*a*	*sb*:*CoolingActuator*.
:*CoolingActuator*	*ssn*:*ofFeature*	:*Room*.

The left top SPARUL query #1 in Figure 4 matches the room as ?*feat* = :*Room* that instantiates class ?*featCls* = *sb*:*Room* and requires the property class ?*propCls* = *sb*:*Energy*. A new unique URI ?*newuri1* is computed for this



(a) Generic conceptual level for the smart building domain



(b) Process relationships for the example.

Fig. 5: Conceptual level and process model for the example.

match. The insert part then adds a new instance $\{?newuri1\} \sqsubseteq sb{:}Energy$ and links it to $:Room$ as *ssn:isPropertyOf* and *ssn:hasProperty* vice versa.

The top right SPARUL query #2 will match $?sensor = :CoolingActuator$ of class $?sensCls = sb{:}CoolingActuator$ that links to $?propCls = sb{:}Cooling$ and create a new instance of *sb:Cooling* under the URI $?newuri2$.

The bottom left query #3 in Figure 4 matches these newly created properties with $?prop1 = ?newuri1$ and $?prop2 = ?newuri2$ as their classes $?propCls1 = sb{:}Energy$ and $?propCls2 = sb{:}CoolingActuator$ are linked by *phy:hasNegIntInfl*. The query creates a new *phy:hasNegIntInfl* instance connecting the properties as the annotation property links to the class via *phy:equalsProcess*.

In a similar way the relationships between all properties can be described. We illustrate this in Figure 5a for the case of smart buildings. The generic domain knowledge was extracted from physical models such as [17]. The figure shows that occupancy, temperature and energy are defined as mandatory properties for a room by *phy:requiresProperty* annotation properties. For the outside only energy and temperature are mandatory as occupancy has a negligible influence on the temperature outside. The sensors define occupancy, temperature, cooling setpoint and cooling as optional via the *phy:defaultObserved* annotation.

The *phy*:*hasPosIntInfl* and *phy*:*hasNegIntInfl* annotations define the positive and negative correlation processes within a room. The temperature in the room is positively influenced by the internal energy. The energy is increased by people in the rooms and decreased by an active cooling system. The cooling system actuates based on the room temperature and setpoint. A room also exchanges energy with the outside and neighboring rooms depending on the temperature difference at the wall. This is defined by a *phy*:*hasPosExtInfl* annotation property between temperature and energy.

Applying this knowledge to our example in Figure 3 using the SPARUL queries in Figure 4 results in the processes shown in Figure 5b. The added mandatory properties by SPARUL query #1 are light gray. The optional properties and the links between sensors and properties added by query #2 are dark gray. The adaptation of the model is visible for the *Outside* feature, which does not have properties for the cooling actuator, setpoint or occupancy as they were not defined as mandatory. The internal processes added by query #3 are identical to Figure 5a. To keep readability we replaced the process classes by solid and dashed arrows representing positively and negatively correlated processes, respectively. The external processes added by query #4 are highlighted by a double line. After these four simple queries the SSN contains a semantic physical model that describes the relationships between sensors in the whole building.

4.3 Generating the Diagnosis Model

The physical process model can be used for automating analytics. We illustrate this for the task of extracting a diagnosis model for an anomaly of a sensor. The diagnosis model defines hypotheses of potential causes for each anomaly. The physical processes can be used to trace these cause-effect-relationships. We consider sensors as potentially observing the cause of an anomaly, if the properties they observe are either linked directly by a physical process to the anomaly or via a sequence of unobserved properties linked by processes in the direction of effect.

Example 3 ***(Potential Causes of an Effect)***
Consider the generation of diagnosis rules for the anomaly $\{\mathit{TempHigh}\} \sqsubseteq$ *phy*:*Anomaly* that is observed by the room temperature sensor $\{\mathit{TempSensor}\}$ in our example. By tracing back the physical processes plotted in Figure 5b the following potential causes can be identified:

- Observation $\{\mathit{OccCause}\} \sqsubseteq$ *phy*:*ObservedCause* of the $\{\mathit{OccupancySensor}\}$ is a potential cause since $\{\mathit{Occupancy}\}$, $\{\mathit{Energy}\}$, $\{\mathit{Temperature}\}$ is a chain of properties linked by physical processes in Figure 5b with $\{\mathit{Occupancy}\}$ and $\{\mathit{Temperature}\}$ being the only properties observed by a sensor,
- $\{\mathit{OutTempCause}\}$ observed by $\{\mathit{OutsideTempSensor}\}$ is a *phy*:*ObservedCause* since $\{\mathit{OutsideTemp}\}$, $\{\mathit{Energy}\}$, $\{\mathit{Temperature}\}$ is a chain without other observable properties,
- $\{\mathit{CoolActCause}\} \sqsubseteq$ *phy*:*ObservedCause* is observed by $\{\mathit{CoolingActuator}\}$ and a potential cause via the chain $\{\mathit{Cooling}\}$, $\{\mathit{Energy}\}$, $\{\mathit{Temperature}\}$.

There is no potential cause at the cooling setpoint value as $\{\mathit{CoolingSetpoint}\}$ is only connected through the already observed $\{\mathit{Cooling}\}$ property.

To implement the detection of process chains in SPARUL, we iteratively combine a chain of two successive physical process instances by adding a new direct process instance. We preserve the type of the process correlation as it is relevant for the diagnosis. The occupancy sensor for example influences the energy by a positive correlation process and the energy influences the temperature also positively. From these two successive positive correlation processes it follows that the occupancy influences also the temperature by a positive correlation via the energy. Thus, we can add a new positive process that connects the occupancy to the temperature. In a similar way, we can combine any successive positive and negative correlation process by a negative one and two negative ones neutralize each other to one positive correlation process. The SPARUL code #5 of Figure 6 shows an example for a process chain with positive and negative correlation. It filters observable properties. Three similar SPARUL queries replace other combinations of positive and negative correlation processes.

```
INSERT {          #5: Combine process chains
 ?newuri5 rdf:type      phy:NegCorrProc.
 ?newuri5 phy:hasInput  ?prop1.
 ?newuri5 phy:hasOutput ?prop3.
} WHERE {
 ?proc1   a             phy:PosCorrProc.
 ?proc2   a             phy:NegCorrProc.
 ?proc1   phy:hasInput  ?prop1.
 ?proc1   phy:hasOutput ?prop2.
 ?proc2   phy:hasInput  ?prop2.
 ?proc2   phy:hasOutput ?prop3.
 FILTER NOT EXISTS
          {?prop2 ssn:forProperty ?anyDP}
 BIND(UURI(?prop1,?prop3) as ?newuri5).
}
```

```
INSERT {              #6: Create potential causes
 ?newuri6 rdf:type         phy:ObservedCause.
 ?abnom   phy:hasPotCause  ?newuri6.
 ?newuri6 ssn:observedBy   ?sensor2.
} WHERE {
 ?abnom   a                phy:Anomaly.
 ?abnom   ssn:observedBy   ?sensor1.
 ?sensor1 ssn:forProperty  ?prop1.
 ?proc1   phy:hasOutput    ?prop1.
 ?proc1   a                phy:PhysicalProcess.
 ?proc1   phy:hasInput     ?prop2.
 ?sensor2 ssn:forProperty  ?prop2.
 BIND(UURI(?sensor2) as ?newuri6).
}
```

Fig. 6: SPARUL code to create diagnosis rules.

Example 4 ***(Process Chains)***
From example 3, we can combine the process chain {*Occupancy*}, {*Energy*}, {*Temperature*} of only positive correlation processes by a direct positive correlation process linking {*Occupancy*} to {*Temperature*}. The chain {*OutsideTemp*}, {*Energy*}, {*Temperature*} also contains only positive correlation processes and can be combined into a new positive correlation process between {*OutsideTemp*} and {*Temperature*}. The chain {*Cooling*}, {*Energy*}, {*Temperature*} contains one negative and one positive correlation process that can be combined into one negative correlation process between {*Cooling*} and {*Temperature*}.

The above approach creates direct process links between the influencing properties. This enables us to directly link potential causes of an anomaly with the SPARUL code #6 in Figure 6. The query first looks for anomalies defined in the ontology. It then identifies the property of the sensor that observed the anomaly. For each observable property that is connected by a physical process to the former property, a potential cause observation is created for the observing sensor. This cause state is then assigned to the anomaly as potential cause.

We utilize the semantic type of processes to narrow down the nature of the potential cause. For example, if the anomaly is characterized by a *sb*:*High* state, then a cause that is connected by a positive correlation process is probably also *sb*:*High*. If the cause is connected by a negative correlation process it is probably *sb*:*Low*. This is implemented by modifying query #6 for the different cases.

Example 5 ***(Cause Classification)***
The anomaly {*TempHigh*} is defined as *sb*:*High* in our input SSN. Using the semantic information of the direct link processes in example 4 it can be derived that {*OccCause*} and {*OutTempCause*} should also be instance of *sb*:*High* as the linking processes are positively correlated. Only {*CoolActCause*} is an instances of *sb*:*Low* as it is linked by a negative correlation process.

4.4 Discretisation and Diagnosis

To use our diagnosis model it is necessary to discretize the sensor time series values. For scalability reason, we only extract abnormal observations. An anomaly is detected if predefined rules are violated. These rules define a normal operation range and we classify observations as *sb*:*High* if they are above this range and *sb*:*Low* if they are below it. For example, all room temperatures higher 22 °C are assigned to the observation instance *sb*:*TempHigh* by adding the property *ssn*:*observationSamplingTime* with the current time. The same applies to potential causes, which are assigned to corresponding *sb*:*High* or *sb*:*Low* observation instances if the current sensor value is above or below an upper and lower threshold. These limits are determined from historical data using a statistical model [14]. The model learns from historical anomaly-free time series data what the data range is under normal circumstances. It bases on the intuition that a cause of an anomaly is also characterized by abnormal values in comparison to the anomaly-free data. The discretization allows for a fast diagnosis using SPARQL query #7 in Figure 7.

```
SELECT {                                          #7: Diagnose
  ?abnom ?cause ?time.
} WHERE {
  ?abnom a                                         phy:Anomaly.
  ?abnom phy:hasPotCause                           ?cause.
  ?abnom ssn:observationSamplingTime ?time.
  ?cause  ssn:observationSamplingTime ?time.
}
```

Fig. 7: SPARQL code to query diagnosis results.

5 Experiments

We tested the accuracy and scalability of our approach using real-world and synthetic examples. All examples are described using the domain ontology in Fig. 5a that we extended by more subconcepts to model BAS components and related properties and processes (see footnote 1).

5.1 Results at IBM's Technology Campus in Dublin

Our system is in-use for the IBM Technology Campus in Dublin. The site consists of six buildings from different IBM divisions and operated by an external

	TP	TN	FP	FN
Heating Issues	67.95	32.05	0.00	0.00
Bad Isolation	89.58	1.62	0.10	8.70

(a) Real office building.

	TP	TN	FP	FN
CoolFault	94.90	0.00	5.10	0.00
CoolFault (sem)	94.90	5.10	0.00	0.00
Occupancy	86.77	0.84	12.40	0.00
Occupancy (sem)	86.77	8.93	4.30	0.00
WindOpen	89.69	0.00	10.31	0.00
WindOpen (sem)	89.69	0.00	10.31	0.00

(b) Running example.

Table 1: Diagnosis results for different examples with: TP - true positives, TN - true negatives, FP - false positives, FN - false negatives in % of anomalies.

contractor. The buildings provide more than 3,500 sensors. Mapping sensors to the SSN representation and defining the features took more than a day in a first manual approach. We now use an internal label mapping tool that does this in a few minutes. Afterwards the Smart Building Diagnoser is initialized automatically as described above. A big benefit of our approach is the coverage of the resulting diagnostic rules that allows detecting and diagnosing many new anomalies. The campus was formerly managed by the IBM TRIRIGA Environmental and Energy Management Software that monitored 194 sensors with 300 rules. Our test system covers 2,411 sensors, with 1,446 effects and 47,284 potential cause observations linked via 10,029 processes.

We investigated deeper into a 3,500 m^2 office building on the campus to evaluate the diagnostic accuracy of the approach. The building contains 271 sensors including temperature sensors and a heating system in most of the 100 rooms. We defined as abnormal if the temperature falls two degrees below the setpoint. The potential causes for the anomaly are a low outside temperature, neighboring rooms with a low temperature, and an inactive heating system. We compared the causes identified by our approach with feedback from the operator.

In the simulation 4 % of the room temperature samples are abnormal. Table 1a summarizes further results. The diagnoser shows that 67.95 % of these cases are related to an inactive heating system. All cases could be validated by the operator by analyzing the data. He explained the behavior by the fact that the building is only heated at night to utilize low electricity prices. In 89.58 % of the abnormal cases the diagnoser relates the outside temperature which the operator largely confirmed to be the case. Only 8.7 % of the cases that he identified could not be retrieved by our approach. Most relevant for the operator was, that our approach revealed that most of these cases occurred in 11 rooms with severe isolation problems which were using an estimated 50 % of the buildings heating energy.

5.2 Benefits of a semantic diagnosis model

We use a commercial building simulator [17] to evaluate the diagnostic accuracy of the approach for the running example of a room with a cooling system. The anomaly $TempHot$ is detected if the room temperature rises above 22 °C. We first run experiments testing the situation in the room without faults where the cooling system controlled the room temperature without anomalies. We then defined temporarily break downs of the cooling system, high room occupancy, or an air exchange through an open window. We simulated the room behavior for a full year for a building in Athens to have a constantly high outside temperature and applyed the introduced semantic diagnoser as well as a non-semantic diagnoser. This non-semantic diagnoser has no information whether the cause could be tracked back to a too high or to a too low sensor reading.

Table 1b illustrates the results. In case of the *cooling system break down* the room heats slowly to the higher outside temperature. The approach correctly detects 94.9 % of the cases when the anomaly $TempHot$ occurs together with a cooling break down. The non-semantic approach has a high false positive rate of 5.1 % as it assigns also cases when the cooling actuator is active. The semantic diagnoser knows that the actuator value has to be low and correctly refuses other cases. In the scenario of a high *occupancy*, the room temperature increases due to a large group of people. 86.77 % of the cases are correctly identified by both approaches. However, the non-semantic diagnosis approach has again a higher false positive rate: 12.4 % compared to 4.3 % of the semantic approach. It avoids assigning situations with low room occupancy. If the *window is open* the room temperature quickly adjusts to the outside temperature. In this scenario both approaches have the same true positive rate of 89.69 % and the same false positive rate of 10.31 %. The latter cases occur when the effect of a high room temperature still persists for a while after the window was closed.

These results demonstrate that the semantic diagnosis approach is not only capable of correctly diagnosing different fault scenarios. The cooling fault and occupancy scenario also shows that the semantic model provides additional information and can exclude illegitimate causes. Further improvements are expected by considering more specific types of processes including delays to further reduce the false positive rate of delayed effects such as the already closed window. This simple example illustrates already that semantic information benefits diagnosis.

5.3 Scalability

Finally, we investigated in the scalability of the approach using synthetic examples. For this we evaluated the performance of the approach for examples of different size and reasoner configurations. We evaluated examples with up to 10 storeys with 100 rooms each arranged north and south of a long corridor. All rooms are equipped with heating, cooling, lighting and ventilation systems.

For the example the number of created processes, observations, and triples scales linearly with the number of sensors. It starts with 78 thousand triples for 55 sensors and reaches 20 million triples for the large example with 15 thousand sensors. The small example contains 92 processes and 99 observations and the large example has 70 thousand processes and 80 thousand observations.

Figure 8 shows the mean computational time on a PC with an Intel Xeon X5690 processor for different reasoner configurations of the Jena framework. The performance strongly depends on the reasoning capabilities of the model. The micro OWL rules inference engine takes in mean 130 minutes to apply the SPARUL queries #1 to #7 to the large example with 15 thousand sensors. The RDFS inferencer applies the same SPARUL rules in 85 minutes. An OWL model

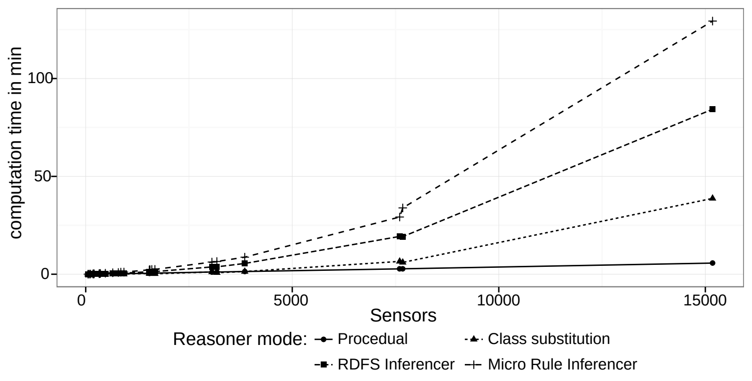


Fig. 8: Computation times for the diagnosis model generation and diagnosis.

with no additional entailment reasoning computes in 38 minutes including the required preprocessing of the necessary class subsumption by two additional SPARUL queries. The reasoner spend in total 37 % of the time for the combination of process chains with query #5 and 58 % for identifying the potential causes with query #6 in Figure 6. This is primarily related to the generality of query #5, which creates many physical processes unneeded for the diagnosis purpose. This significantly increases the search space for query #6. The best computation time performance shows a procedural implementation of the SPARUL queries #1 to #6 in Java. It benefits from a depth-first search in the graph model of only the processes connected to an anomaly. Note that, the IBM campus model computes in 55 s with the procedural implementation.

Please note that queries #1 to #6 are only executed once during the initialization phase of the system. During runtime only the discretizer and query #7 are executed. They compute in less than a second for the IBM Campus.

6 Conclusion

We have shown that semantic techniques can be used for automating analytic tasks in complex systems such as buildings. Specifically we have presented IBM's Semantic Smart Building Diagnoser which is the first of its kind that can automatically derive diagnosis rules from the sensor network definition and behaviour of a specific building. This allows not only for the diagnosis of smart building problems that existing techniques cannot diagnose but also for easier adaptability to other buildings.

Our approach was realized by using semantic techniques for: (i) integrating heterogeneous data from different buildings, (ii) extending SSN for automating the creation and configuration of physical models, and by (iii) automatically deriving the diagnosis rules from the latter. The addition and extension of new sensor types and processes is also straightforward given the annotation patterns of our domain ontology.

Our experiments have shown that we can indeed efficiently identify the causes of anomalies for real buildings. They also revealed that semantic information can even be used to improve the accuracy of the diagnosis result. Our approach currently runs on IBM's Technology Campus in Dublin and has provided several insights for improving energy performance. Future deployments on further sites are planned.

References

1. Pfisterer, D., Romer, K., Bimschas, D., et al.: SPITFIRE: toward a semantic web of things. IEEE Commun. Mag. **49**(11) (2011) 40–48
2. Compton, M., Barnaghi, P., Bermudez, L., et al.: The SSN ontology of the W3C semantic sensor network incubator group. Web Semantics **17** (2012) 25–32
3. Han, J., Jeong, Y.K., Lee, I.: Efficient building energy management system based on ontology, inference rules, and simulation. In: Int. Conf. on Int. Building and Mgmt. (2011) 295–299
4. Lécué, F., Schumann, A., Sbodio, M.L.: Applying semantic web technologies for diagnosing road traffic congestions. In: ISWC. Springer (2012) 114–130
5. International Energy Agency: World energy outlook 2012 (2012)
6. Ploennigs, J., Hensel, B., Dibowski, H., Kabitzsch, K.: BASont - a modular, adaptive building automation system ontology. In: IEEE IECON. (2012) 4827–4833
7. Bonino, D., Corno, F.: DogOnt-ontology modeling for intelligent domotic environments. In: ISWC, Springer (2008) 790–803
8. Zhou, Q., Wang, S., Ma, Z.: A model-based fault detection and diagnosis strategy for HVAC systems. Int. J. Energ. Res. **33**(10) (2009) 903–918
9. Katipamula, S., Brambley, M.: Methods for fault detection, diagnostics, and prognostics for building systems - a review. HVAC&R Research **11**(1) (2005) 3–25
10. Jacoba, D., Dietza, S., Komharda, S., Neumanna, C., Herkela, S.: Black-box models for fault detection and performance monitoring of buildings. J. Build. Perf. Sim. **3**(1) (2010) 53–62
11. Schein, J., Bushby, S.T.: A hierarchical rule-based fault detection and diagnostic method for HVAC systems. HVAC&R Research **1**(1) (2006) 111–125
12. Brady, N., Lecue, F., Schumann, A., Verscheure, O.: Configuring Building Energy Management Systems Using Knowledge Encoded in Building Management System Points Lists. US20140163750 A1 (2012)
13. Tallevi-Diotallevi, S., Kotoulas, S., Foschini, L., Lécué, F., Corradi, A.: Real-time urban monitoring in Dublin using semantic and stream technologies. In: ISWC. (2013) 178–194
14. Ploennigs, J., Chen, B., Schumann, A., Brady, N.: Exploiting generalized additive models for diagnosing abnormal energy use in buildings. In: BuildSys - 5th ACM Workshop on Embedded Systems for Energy-Efficient Buildings. (2013) 1–8
15. Janowicz, K., Compton, M.: The stimulus-sensor-observation ontology design pattern and its integration into the semantic sensor network ontology. In: Int. Workshop on Semantic Sensor Networks. (2010) 7–11
16. Ploennigs, J., Schumann, A., Lecue, F.: Extending semantic sensor networks for automatically tackling smart building problems. In: ECAI - PAIS. (2014)
17. Sahlin, P., Eriksson, L., Grozman, P., Johnsson, H., Shapovalov, A., Vuolle, M.: Whole-building simulation with symbolic DAE equations and general purpose solvers. Building and Environment **39**(8) (2004) 949–958