Improving maintainability of calendar-based IoT-driven office with knowledge graph and rule-based reasoning

Sogo Amagai, Pierre Maret, Kamal Singh

Improving maintainability of calendar-based IoT-driven office with knowledge graph and rule-based reasoning

Sogo Amagai*
samagai@s.h.k.u-tokyo.ac.jp
The University of Tokyo
Kashiwa, Chiba, Japan

Pierre Maret
CNRS UMR 5516 Laboratoire Hubert Curien, Université de Lyon
Saint-Étienne, France
pierre.maret@univ-st-etienne.fr

Kamal Singh
CNRS UMR 5516 Laboratoire Hubert Curien, Université de Lyon
Saint-Étienne, France
kamal.singh@univ-st-etienne.fr

ABSTRACT

Designing smart Internet of Things (IoT)-driven offices has recently gained considerable attention. Such smart offices are expected to enhance worker productivity and economise electricity for sustainability. However, there are several open issues related to the maintenance of IoT applications.

In this study, we implement a calendar-based IoT-driven office exploiting knowledge graphs and rule-based reasoning. The smart IoT-driven office consists of micro-controllers (MCUs) with sensors and actuators, and a controller running on PC. These smart nodes send sensor data which in turn is represented as a Knowledge graph. This has different advantages in terms of interoperability, ease of integration with other systems and improved device compatibility. Decision-making is carried out by a rule-based reasoner. The reasoning process of a rule-based reasoner is more transparent than some machine learning approaches. This allows for easy configuration and adaptation of rules.

CCS CONCEPTS

• **Information systems → Graph-based database models**.

KEYWORDS

Internet of Things, Knowledge graph, rule-based reasoning

ACM Reference Format:
Sogo Amagai, Pierre Maret, and Kamal Singh. 2023. Improving maintainability of calendar-based IoT-driven office with knowledge graph and rule-based reasoning. In *The International Conference on the Internet of Things (IoT 2023), November 7–10, 2023, Nagoya, Japan.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3627050.3630731

1 INTRODUCTION

The Internet of Things (IoT) is a network of physical devices which are embedded with sensors, software, and network connectivity, allowing them to collect and exchange data. In recent years, different solutions for smart offices based on IoT have been proposed [6]. An IoT-driven office offers several benefits, including improved employee productivity [7] and reduced energy consumption [9].

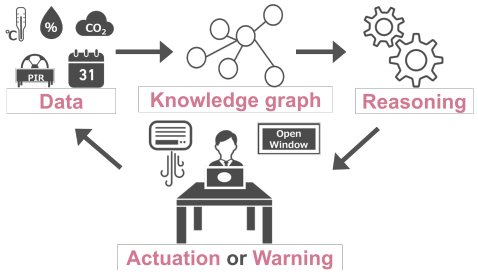

Figure 1: Overview of rule-based reasoner for IoT

However, it is well known that IoT has several challenges to overcome [11]. Some of the challenges are interoperability, integration and maintenance which needs to be overcome for enabling automation using IoT in large systems such as Smart Cities [10]. In this paper, we argue that the maintainability of IoT applications can be improved through the use of Knowledge graph (KG). This will bring benefits such as interoperability, ease of changing hardware devices, ease of integrating with other systems, and expressivity of rule based reasoning. IoT systems with higher maintainability are expected to better address the multiple challenges faced by the IoT domain.

This work proposes a calendar-based IoT-driven office that utilizes KG as well as rule-based reasoning. A calendar-based IoT is suitable for office workers to easily manage conditions of many rooms. However, as the number of rooms increase, the amount of data will increase and managing data will become complicated. Thus, a KG which represents knowledge in a graph format [5] is more suitable for such scenarios. KG can connect different types of data in meaningful and flexible ways. The decision-making process is then carried out by a rule-based reasoner that operates on explicit rules and the knowledge base, such as the KG [1, 3], as illustrated in Figure 1.

Through the use of KG and rule-based reasoners, a prototype of a calendar-based IoT-driven office with higher maintainability has been developed. Our work proposes easily maintainable IoT systems that can be generalised to other IoT-based systems. This system serves as a basis for the development of IoT-based systems on a larger scale. The problems identified during the actual development of this system will be valuable for future IoT development.

This paper is organized as follows. Section 1 provides the background and overview of this work. Section 2 outlines how we implemented our proposed system. Section 3 presents the tasks

IoT 2023, November 7–10, 2023, Nagoya, Japan

ACM ISBN 979-8-4007-0854-1/23/11.
https://doi.org/10.1145/3627050.3630731

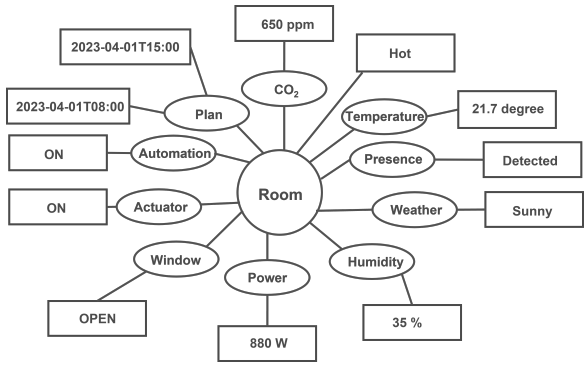

Figure 2: knowledge graph for this application

accomplished in our work. Section 4 discusses the outcomes as well as future works.

2 MAINTAINABILITY OF IoT APPLICATIONS

2.1 Graph-based Data Model

Graph-based data models, such as KGs, are a popular way of modeling and analyzing data that contains complex relationships between entities. This type of data model represents entities as nodes and their relationships as edges, creating a network of interconnected nodes that can be queried and analyzed in a variety of ways [5].

Figure 2 is a simplified illustration of the KG used in this system. This KG is based on Semantic Sensor Network (SSN) Ontology [4], Sensor, Observation, Sample, and Actuator Ontology (SOSA) [8] and Time Ontology [2]. The entity called Room is associated with an entity of physical quantities observed by sensors and information extracted from several APIs. The room entity also has entities for objects in the room, such as windows and actuators. In addition, room's automation is kept off when not needed in order to reduce the power consumption of the room. This on or off state is determined by calendar event information. KG can meaningfully represent different types of data and knowledge, such as sensor output and calendar event information. The room entities hold information about the state of the room derived from the reasoning results. Figure 3 gives snippets of Resource Description Framework (RDF) data to create a knowledge graph. The first few lines are used for the triple that describes the windows, actuators, etc. in the room. In the following lines, another triple describes the information about the room. Information such as the state of the room and the time is stored. The entity named as Observation stores information related to temperatures observed by sensors. The entity calendarEvent holds the event's start time, end time, and event title retrieved from Google Calendar API. Google Calendar can also be easily used by other systems. That would be helpful during the integration with other systems.

Such knowledge graphs in general can easily integrate with other data models and are a powerful and flexible way to represent knowledge. Indeed, this integration is expected not only for smart offices but also for large-scale systems such as smart cities. Another advantage is high compatibility with hardware devices for the maintenance of IoT systems: by representing device capabilities and relationships in a standardised way, knowledge graphs can facilitate interoperability between different devices and systems,

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix sosa: <http://www.w3.org/ns/sosa/> .
@prefix ssn: <http://www.w3.org/ns/ssn/> .
@prefix time: <https://www.w3.org/TR/owl-time/#time:> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@base <http://example.org/data/> .

<window> ssn:hasProperty <window/state> .

<window/state> rdf:value "ON" .

<actuator> ssn:hasProperty <actuator/state> .

<actuator/state> rdf:value "ON" .

<automation/room/108>
  a <automation>;
  ssn:hasProperty <automation/state> .

<automation/state> rdf:value "ON" .

<room/e108>
  a sosa:Platform;
  <levelOfTemperature> "Hot" ;
  sosa:resultTime  "2023-03-07T18:45:32Z"^^xsd:dateTime ;
  <hasStatus> <automation/state>;
  sosa:hasFeatureOfInterest <calendarEvents> ;
  sosa:hosts <window>, <actuator>.
<Observation/temperature/1>
  a ssn:Observation ;
  sosa:observedProperty <temperature>;
  sosa:hasFeatureOfInterest <room/e108> ;
  sosa:resultTime "2023-03-07T18:45:36Z"^^xsd:dateTime ;
  sosa:hasSimpleResult 2.167000e+1 .

<calendarEvents>
  a "work" ;
  time:hasEnd "2023-04-01T16:00:00Z"^^xsd:dateTime ;
  time:hasBegin "2023-04-01T13:30:00Z"^^xsd:dateTime .
```

Figure 3: Example of RDF data

making it easier to integrate new devices into IoT applications [8]. Consequently, knowledge graph data models will be one of the necessary elements of IoT applications. This will become even more prevalent in the future as IoT applications become larger.

2.2 Rule-based reasoning

Rule-based reasoners are a type of reasoning engine which infer new knowledge from existing knowledge using a set of logical rules.

```
[IsPlanComingSoon:
 (<calendarEvents> time:time:hasBegin ?x)
 (<room/e108> sosa:resultTime ?v) lessThan(?x ?v)
 -> (<automation/state> rdf:value "ON")]

[ruleIsAboveTempThreshold:
 (?x sosa:hasSimpleResult ?v)
 (?x sosa:observedProperty <temperature>)
 (<automation/state> rdf:value "ON") greaterThan(?v, 28)
 -> (<room/e108> <levelOfTemperature> "Hot")]

[ruleInCaseNeedVentilation:
 (<room/e108> <levelOfAirQuality> “NeedVentilation")
 (<room/e108> sosa:hosts <window>)
 -> (<window/state> rdf:value "OPEN")]

[ruleInCaseHot:(<window> ssn:hasProperty "CLOSED")
 (<room/e108> <levelOfTemperature> "Hot")
 (<room/e108> sosa:hosts <actuator>)
 -> (<actuator> ssn:hasProperty "ON") ]
```

Figure 4: Snippets of rules

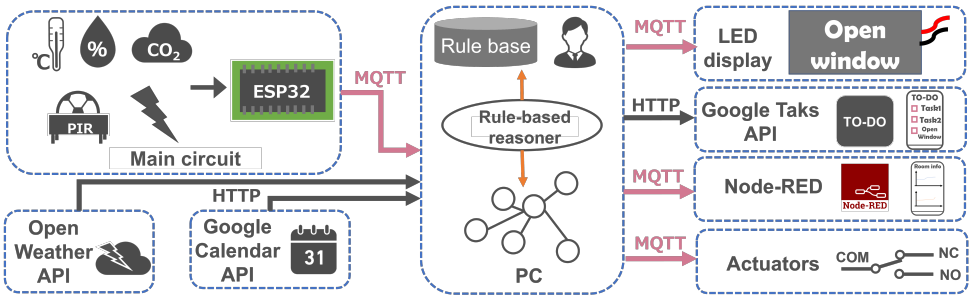

Figure 5: System configuration

We used Apache Jena as the reasoning engine. These reasoning engines work by inputting a set of rules and a set of explicit facts, which are RDF triples in our case, and generating new implicit facts derived by using the rules. The main advantage of rule-based reasoners is their flexibility. Rules can be defined in many different ways, allowing for a wide range of reasoning tasks. In addition, rules can be updated or modified as new information becomes available, allowing continuous improvement and refinement of the reasoning process. Figure 4 shows the example rules used in our application. These rules are written in Notation 3 syntax. These rules are written manually by the developers. Here we describe four rules among several. The first rule is to turn on room automation if the calendar event comes within 30 minutes. Thus, the event start time is used in the rule to compare it with the current time. The lessThan and greaterThan operators used for comparison are builtin primitives from Apache Jena. The second rule is to determine if the room temperature is higher than the threshold value. The third rule is to open the room window when the room needs ventilation. The fourth rule is to operate the actuator when the room window is closed and the room is hot.

Table 1: General rule expressions setting the levels of temperature, air quality, humidity, and energy consumption respectively

IF	THEN
Room temperature is higher than 30 degrees	Very Hot
Room temperature is higher than 28 degrees	Hot
Room temperature is lower than 18 degrees	Cold
Room temperature is lower than 16 degrees	Very Cold
Concentration of CO_2 is larger than 1500 ppm	Need Ventilation
Room humidity is higher than 65 %	Humid
Room humidity is less than 45 %	Dried
Energy consumption is higher than 10 kW	Over Used

Table 1 shows the meaning of these rule expressions used in this application. These rules relate to physical quantities such as temperature and humidity in the room, as observed by the sensors. These rules set a threshold and add new information, such as alerts, to the room entities in the knowledge graph when they are above or below that threshold. The threshold can be changed by the users.

Table 2: Rule expressions related to alerts

IF	THEN
"Need Ventilation"	"OPEN" Windows
"Hot" and window is "CLOSED"	Turn the actuator "ON"
"Cold" and window is "CLOSED"	Turn the actuator "ON"
"Over Used" but "Very Hot"	Turn the actuator "OFF"
"Over Used" but "Very Cold"	Turn the actuator "OFF"

Table 2 shows the rule expressions that are based on alerts or alarms. The actual actions to be done by the actuators and the material to be described in the user interface are obtained from these rule expressions via room alarms derived from the general rule expressions. Furthermore, regulations are defined not just for a single alarm but also for many alarms and window conditions. This method of expressing rules allows for workplace comfort as well as energy savings.

Table 3: Rule expressions related to calendar events

IF	THEN
Next plan is coming in 30 mins	Automation is "ON"
Current plan is ending in 15 mins	Automation is "OFF"
Next plan is NOT coming	Automations is "OFF"
Next plan is NOT coming BUT your presence is detected	Automation is "ON"

Today, many workers use virtual calendars such as Google Calendar. Knowledge of room occupancy information from the calendar allows more sophisticated inferences to be made to improve workspace conditions. Table 3 shows the rule expressions for reasoning based on calendar events. As shown in the first row of the table, the information from the calendar can be used to condition the room in advance. A pre-conditioned room makes it easier for workers to start their work. As shown in the second row of the table, energy savings can also be expected by switching off the automation to make the room comfortable ahead of the leaving time. Conversely, there is the problem that these functions cause the room automation to be switched off regardless of whether people are in the room. We used PIR sensors for presence detection and wrote rules as shown in the fourth row of the table. With KG, it's easy to utilize sensor data and calendar information simultaneously, enabling sophisticated reasoning.

2.3 Implementation

Figure 5 depicts a detailed system diagram of this application. The main circuit comprises several sensors surrounding the ESP32 that measure temperature, humidity, CO_2, human presence, and power consumption. The measured values are sent to the PC via Message Queuing Telemetry Transport (MQTT). MQTT communication is bidirectional, and it has many advantages, such as being lightweight and battery-friendly, which makes it widely used in machine-to-machine systems. Furthermore, information on calendar events and weather forecasts is sent to the PC via HyperText Transfer Protocol (HTTP). The knowledge graph is constructed from this information, and the rule base is developed by considering use cases and scenarios. The rule base could include more rules by considering more detailed situations. Based on the rule base and the knowledge graph, reasoning is performed to derive the workspace conditions and based on that the actuators can take actions. On the PC, the code for MQTT communication, API usage, knowledge graph composition, and rule engine execution is written in Java due to the availability of the library. The new information derived by reasoning is then sent to a LED display, smartphone, and actuator. The LED display shows basic information, such as the room's temperature and humidity, as well as tasks that the user needs to perform. For many people, room conditions may not be interesting enough to be constantly monitored. Hence, a to-do list is used to encourage users to take actions regularly, and it is less likely to be forgotten than other methods. It also does not reduce productivity by avoiding high number of notifications that are usually sent by smartphones and laptops. The user interface, which can be viewed on smartphones and laptops, was developed using the Node-RED visual programming language. Electromagnetic relays are used to switch devices on and off for room automation.

3 DISCUSSION

As explained so far, we have developed a calendar-based IoT-driven office that is expected to be actively expanded and integrated into larger IoT applications in the future. Through this development, IoT applications with knowledge graphs do not need to redesign the data model when adding new sensors. In other words, the addition of new sensors, etc., is easier as compared to IoT applications which lack knowledge graphs. Additionally, the feature of knowledge graphs, which can be easily integrated with other data models, can be expected to be integrated with larger-scale systems.

The use of a rule-based reasoner makes it easy to improve and add new rules. Furthermore, the open-box nature of the system allows developers and others to understand the reasoning process. On the other hand, a disadvantage of this system is the developer effort required to write rules for possible scenarios. There are quite a lot of possible scenarios in a room, and it can be quite difficult and labor-intensive to write rules that cover all of them. Another potential major problem is rule conflict. It is possible that several logically correct but incompatible rules could be created, causing the system to behave unexpectedly.

Furthermore, in the calendar-based IoT, using KG and rule-based reasoner can be suitable because it can operate a lot of data in a human-readable way and can easily integrate with other IoT systems.

4 CONCLUSION

In conclusion, we have developed a calendar-based IoT-driven office system with a knowledge graph and a rule-based reasoner to improve the maintainability of IoT systems. The use of knowledge graphs in this system has several advantages, such as high device compatibility and an open-box rule base. Thus, this study has contributed to improving the maintainability of IoT systems that can perform sophisticated reasoning by combining sensor and calendar information.

However, it has become clear that there are still more challenges that need to be addressed to further improve maintainability. Future challenges include the following.

- Generalisation of the knowledge graph
- Simplifying rule editing and compatibility checking
- Experimental validation and analysis on larger systems
- Running the reasoning system itself on the embedded devices

The IoT's high maintainability is likely to lead to further adoption, and we intend to address these challenges in the future.

ACKNOWLEDGMENTS

This work was partially supported by grant ANR-19-CE23-0012 from the Agence Nationale de la Recherche, France, for the CoSWoT project.

REFERENCES

[1] Xiaojun Chen, Shengbin Jia, and Yang Xiang. 2020. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications* 141 (2020), 112948. https://doi.org/10.1016/j.eswa.2019.112948

[2] Simon Cox, Chris Little, Jerry R Hobbs, and Feng Pan. 2017. Time ontology in OWL. *W3C recommendation* 19 (2017).

[3] Soumitra Dutta and Piero P. Bonissone. 1993. Integrating case- and rule-based reasoning. *International Journal of Approximate Reasoning* 8, 3 (1993), 163–203. https://doi.org/10.1016/0888-613X(93)90001-T

[4] Armin Haller, Krzysztof Janowicz, Simon J.D. Cox, Maxime Lefrançois, Kerry Taylor, Danh Le Phuoc, Joshua Lieberman, Raúl García-Castro, Rob Atkinson, and Claus Stadler. 2019. The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation. *Semantic Web* 10 (2019), 9–32. https://doi.org/10.3233/SW-180320 1.

[5] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. 2022. Knowledge Graphs. *ACM Comput. Surv.* 54, 4 (2022), 71:1–71:37. https://doi.org/10.1145/3447772

[6] Andrea Horch, Michael Kubach, Heiko Roßnagel, and Uwe Laufs. 2017. Why Should Only Your Home Be Smart? - A Vision for the Office of Tomorrow. In *2017 IEEE International Conference on Smart Cloud (SmartCloud)*. 52–59. https://doi.org/10.1109/SmartCloud.2017.15

[7] Tahera Hossain, Kizito Nkurikiyeyezu, Yusuke Kawasaki, and Guillaume Lopez. 2022. Toward the Prediction of Environmental Thermal Comfort Sensation Using Wearables. In *Workshops at 18th International Conference on Intelligent Environments (IE2022), Biarritz, France, 20-23 June 2022 (Ambient Intelligence and Smart Environments, Vol. 31)*, Hernán Humberto Álvarez Valera and Mitja Lustrek (Eds.). IOS Press, 312–324. https://doi.org/10.3233/AISE220058

[8] Krzysztof Janowicz, Armin Haller, Simon J.D. Cox, Danh Le Phuoc, and Maxime Lefrançois. 2019. SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics* 56 (2019), 1–10. https://doi.org/10.1016/j.websem.2018.06.003

[9] Carsten Röcker. 2010. Services and Applications for Smart Office Environments - A Survey of State-of-the-Art Usage Scenarios. *World Academy of Science, Engineering and Technology* 61 (01 2010).

[10] Carsten Röcker. 2010. Services and Applications for Smart Office Environments - A Survey of State-of-the-Art Usage Scenarios. *World Academy of Science, Engineering and Technology* 61 (01 2010).

[11] C. C. Sobin. 2020. A Survey on Architecture, Protocols and Challenges in IoT. *Wireless Personal Communications* 112, 3 (01 Jun 2020), 1383–1429. https://doi.org/10.1007/s11277-020-07108-5