Towards an emulation tool based on ontologies and data life cycles for studying smart buildings

Christophe Cérin, Frédéric Andres, Danielle Geldwerth-Feniger

Towards an emulation tool based on ontologies and data life cycles for studying smart buildings

Christophe Cérin, Frédéric Andres, Danielle Geldwerth-Feniger

Towards an Emulation Tool based on Ontologies and Data Life Cycles for Studying Smart Buildings

Christophe Cérin
University Sorbonne Paris Nord,
LIPN UMR CNRS 7030
Villetaneuse, France
christophe.cerin@univ-paris13.fr

Frédéric Andres
National Institute of Informatics,
Digital Content and Media,
Sciences Research Division
Tokyo, Japan
andres@nii.ac.jp

Danielle Geldwerth-Feniger
University Sorbonne Paris Nord,
CSPBAT UMR CNRS 7244
Villetaneuse, France
danielle.geldwerth-feniger@univ-paris13.fr

ABSTRACT

In this paper, we share our vision to study a complex Information Technology (IT) system handling a massive amount of data in the context of 'smart buildings.' One technique for analyzing complex IT systems relies on emulation, where the final software system is fully deployed on real architectures, and is evaluated in considering "small" instances of situations the system is supposed to solve. We propose a software architecture for studying the ecosystem of 'smart buildings'. This software architecture is built: 1) on top of ontologies for the description of smart buildings; 2) on a special tool for mastering the life cycle of data produced by sensors and actuators inside the buildings.

We assume that it is equally important to model both the building's components and the flow of data produced inside the building. We use existing software components for both goals and to make real our concerns. According to a translational methodology, we also discuss use cases for illustrating the potential of our approach and the particular challenges associated with making the two main components of our emulation tool inter-operate.

Therefore, our main contribution is to propose a comprehensive, ambitious and realistic research plan to guide communities. The paper illustrates how computer scientists and smart buildings domain scientists may communicate to address and solve specific research problems related to Big Data in emergent distributed environments. We are also guessing that experimental results that can demonstrate the practicality of the proposed combination of tools could be devised in the future, based on our broad vision. The paper is, first and foremost, a visionary paper.

CCS CONCEPTS

• **Computer systems organization** → **Sensors and actuators**; • **Information systems** → **Information integration**; • **Applied computing** → **Computers in other domains**.

KEYWORDS

Smart buildings; Ontology; Data life cycle; Big data tools, systems and methods; Emulation principles

ACM Reference Format:
Christophe Cérin, Frédéric Andres, and Danielle Geldwerth-Feniger. 2021. Towards an Emulation Tool based on Ontologies and Data Life Cycles for Studying Smart Buildings . In *Big Data in Emergent Distributed Environments (BiDEDE'21), June 20, 2021, Virtual Event, China.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3460866.3461772

1 INTRODUCTION

Frost and Sullivan[1] predicts smart city development worldwide will create business opportunities worth US$2.46 trillion by 2025, adding that the uncertainties of the post-pandemic work will compel cites to focus more on developing collaborative, data-driven infrastructure to provide healthcare facilities as well as public security services.

Smart Cities offer new opportunities for novel applications of the core concepts of AI, Cloud, and Big Data[2]. In the case of smart buildings, if we decide to give more control over data privacy and security to residents, one may decide to avoid sending data produced by the residents of a building into any

[1]https://ww2.frost.com/news/press-releases/smart-cities-to-create-business-opportunities-worth-2-46-trillion-by-2025-says-frost-sullivan/
[2]https://www.oecd.org/cfe/cities/OECD_Policy_Paper_Smart_Cities_and_Inclusive_Growth.pdf

cloud (GAFAM - Google Amazon Facebook Apple Microsoft, or BATX - Baidu, Alibaba, Tencent, Xiaomi clouds). This new vision entails a new engineering of the building data, processors, and overall controllers. This positioning raises several new research questions about 1- the efficient use of the computing, networking, and storage capabilities owned by the building residents and visitors; 2- the performances of the sensors installed in the building. Both are central if we want to effectively learn from the data produced in-situ inside a building, and later negotiate their economic values with lesser supervision, control and administration from GAFAMs or BATXs.

A building contains many different kinds of computing power units, ranging from smartphones and tablets to small clustered, communicating devices endowed with varying computing capabilities (programmable arrays, sensors, actuators), and also computer heaters and boilers equipped with processors. Altogether, this makes a highly heterogeneous distributed system, reinforced by the fact that computing devices, nowadays, are heterogeneous and include CPU (Central Processing Unit), GPU (Graphical Processing Unit), accelerators and DSP (Digital Signal Processor) on the same silicon substrate.

The first rationale in our long term work consists in the fact that we do not want to send data into high-performance super-computers and data centers, but rather learn and plan from data produced locally. This option leads to optimizing machine learning kernels for the ecosystem of smart buildings. As a second rationale in the long term, the heterogeneous computing infrastructure implies some AI techniques to be revisited. For instance, if the processors work at different speeds, we can spread the data produced in a way proportional to those speeds, but this will lead to heterogeneity in the quality of results since some processors have the opportunity to learn "more" than others.

Our research plan is aligned on the principles of the "Living-in Europe" initiative[3]. For instance, the Open urban data platforms initiative enables cities and communities to:

- Customise the platform according to their needs;
- Avoid vendor lock-in and technology-debt;
- Share data with third parties;
- Connect services and data more easily, and
- Provide better digital services to their citizens at lesser costs.

Moreover, designing and optimizing Machine Learning (ML) kernels and algorithms for computing devices deployed inside buildings requires developing special insights. For instance, we need to capture raw data from the Qarnot heaters or boilers[4] before their efficient analysis. There are not yet efficient and specialized ML libraries for such hardware devices to the best of our knowledge.

Smart buildings constitute a new domain for which machine learning kernels and planning algorithms may be used. The above mentioned Frost & Sullivan's report highlights key findings on smart cities, and mention that *"technologies like artificial intelligence and big data will be in high demand to combat the pandemic, with growing opportunities for crowd analytics, open data dashboards, and online city services. Smart cities' spending on technology in the next six years is expected to grow at a CAGR of 22.7%, reaching $327 billion by 2025 from $96 billion in 2019"*.

Before this vision becomes a reality, we argue that we need to research emulation tools for studying smart buildings. This is the primary purpose of the paper.

We postulate that an emulation tool, or a software system for validating new ideas may constitute a valuable testbed for experimentally validating new pieces of software architectures for smart buildings. In short, in this paper, the term *emulation'* refers to the process of mimicking the observable behavior of a system in such a way that it matches an existing target, here a building. In this paper, we mainly tackle the challenging problems of designing an emulation tool for smart buildings, and testing some part of it experimentally.

In the long term agenda, we hope that the emulation tools could be matured, industrialized, and also serve as building management systems, we mean systems for managing digital data circulating inside the buildings, and deployed inside the buildings.

If we want to counterbalance the influence and omnipresence of GAFAM/BATX, we must: 1- propose another architectural organization of AI computer systems for buildings; 2- use the best of the few existing testbeds in the domain. Our general objective in the long term is to make the building becomes the data center, and learns from its own, leading to a so-called "thinking clever/smart building." The first step consists in designing an instrument that will help to validate (or not) such a vision of smart buildings. Dedicated goals may arise from having artificial intelligence works in the building such as respecting the privacy of the building's users; promoting short circuits for data transformation; allowing better empowerment from the citizens on valuable marketplaces emerging from smart buildings.

Clouds are centralized systems. We do not claim that cloud databases or cloud solutions, in general, have data privacy issues, which is not currently valid since many companies and government entities are now using cloud servers to process their data. We claim that we could imagine distributed ways to sharing, storing and managing data. This vision is our long-term research project. The paper considers only the brick that we will need to be connected to other bricks in the

[3]https://www.living-in.eu/

[4]See https://qarnot.com/en/home/

future to form a federation of buildings. Our will is to raise hopes to study smart buildings through distributed systems.

The organization of the paper is as follows. In Section 2 we provide an overview of related works on Smart Buildings, experimental validation in Computer Science, and translational research. Section 3 introduces the architecture of our emulation tool. Section 4 reports on our experimental work. Section 5 discusses the research issues and potential of our experimental work. Section 6 concludes the paper.

2 RELATED WORKS

The section is divided into $3 + 2$ subsections. The first three initial subsections are related to technologies and related vocabulary, and the last two subsections are related to appropriate methodology to conduct our research.

2.1 Smart Buildings

We address in this paper, as computer scientists, the scientific and technological domains of Smart Buildings. Semantic and technical papers introducing the vocabulary, definitions, and concepts of smart buildings are [1–5]. The view shared by the community defines a smart building as construction with appropriate design and technological supports that allow maximizing the functionalities and comfort offered to the occupants while reducing operational and maintenance costs, and extending the life of the physical structure [1].

In [2] authors present an initial guide to understanding the layers, taxonomy of services, and best practices for the development of smart buildings. Open standards are described as increasing interoperability between layers and services.

In [3] authors explain variations between different notions and clarify the border between an intelligent and a more advanced Smart Building. From a computer science system point of view, We may think of an Intelligent Building as a building reacting to some events. A Smart Building is a building "which integrates and accounts for intelligence, enterprise, and control, with adaptability, not reactivity, at the core." It aims at ensuring the "building progression: energy and efficiency, longevity, and comfort and satisfaction." At the end of the paper, authors categorize the conceptual views by pairs: "building category, control flow" "intelligent buildings, reactive", "smart buildings, adaptive" and, finally, "thinking buildings, predictive". In the present proposal, we continue to use the terminology of 'smart building' rather than 'thinking building' because the latter does not seem to be widespread.

Siemens's white paper [4] is of particular interest. It provides facts and numbers related to the building ecosystem. It also tells us that "In the past, individual elements such as energy efficiency, safety, or even e-mobility were addressed separately - today, we are proud to pioneer holistic smart buildings, which are so much more than the sum of their parts".

The INTEL online document [5] is oriented towards the Internet of Things (IoT) and Building Management Systems (BMS). A BMS is analogous to a supervisory control and data acquisition system, as used in general manufacturing. It monitors and controls various systems in a building, such as heating, ventilation, air conditioning (HVAC); lighting; additional and often separated systems for controlling the elevators, the general safety, security, and accesses.

The technical document [6] gives more details about BMS, Direct Digital Control (DDC), Building Automation System (BAS), Facility Master System Integrator (FMSI), all concepts being defined according to a computer science system point of view. We instead propose here, as the operating system for the building, an orchestrator of machine learning tasks and computing tasks.

Last, the Berkeley Lab provides a good source of papers related to Smart Buildings, from 1978 until today, with its Residential Buildings System project [7] Special focus is made on the movement of air, pollutants, and energy within the building, with associated penalties. Open data regarding energy efficiency are also available.

2.2 Ontologies and Smart Buildings

Gruber introduces in [11] the ontology term as a specification of a conceptualization: *An ontology is a description (like a formal specification of a program) of the concepts and relationships between those concepts that can formally exist for an agent or a community of agents. This definition is consistent with using an ontology as a set of concept definitions but more general. And it is a different sense of the word than its use in philosophy.*

The fifth edition of the Linked Data in Architecture and Construction workshop describes ontologies about the ecosystems of Smart Buildings. The workshop aimed to evaluate whether and how available ontologies are able to describe building related data. Reviewed ontologies include: the ifcOWL ontology (buildingSMART), the Building Topology Ontology (BOT - W3C), the PRODUCT ontology (W3C), Geospatial ontologies, infrastructure related ontologies (oil&gas, roads, tunnels, rail), HVAC ontologies, building automation ontologies (BACS, DogOnt, SAREF), semantic sensor ontologies (SSN).

For instance, in [8] the authors introduce an ongoing research called WiseNET (Wise NETwork) [9]. This OWL-2 ontology provides a vocabulary set for integrating, repurposing, and analyzing information about various domains related to the building context. The various data sources (DUL, event, ifcowl, person, SSN, time ontologies) are integrated by using semantic rules and linked data techniques,

such as Uniform Resource Identifiers(URIs) and Resource Description Framework (RDF).

Using linked data for integration allows the acquisition of extra information about the environment directly from the ifcowl ontology (e.g., the dimensions of a door/wall). It also allows the use and integration of small portions of different ontologies models. This reduces the size of the WiseNET ontology, facilitates its maintenance, and should improve its reasoning speed. As seen below, we reuse these principles in our work.

The field of IoT is rapidly evolving and is often confused with related areas, paradigms, and technologies such as Ubiquitous Computing, Pervasive Computing, Ambient Intelligence, WSN (Wireless Sensor Networks), M2M (Machine-2-Machine) communication, CPS (Cyber-Physical Systems), WoT (Web of Things), Cloud Computing, Big Data, and Context-Awareness. In [35] authors settle on a core horizontal group of extensible concepts and interoperable with domain-specific concepts of IoT applications. As application requirements keep varying with time (such as new requirements recognized by introducing user-centric laws and regulations such as GDPR), researchers need to redefine ontologies. Even unified/comprehensive ontologies need to be updated.

To summarize, the work in [35] proposes a methodology for the ontology developers in the IoT domain and identifies the core concepts for a future standard IoT ontology. The authors highlight the need for flexibility and extensibility in the base ontology to support the integration of vertical concepts for application-specific IoT systems.

2.3 Data life cycle management

Data Life Cycle Management (DLM) refers to actions and tools aimed at structuring the steps followed by information (i.e.; data) within an organization for optimizing its usefulness. It includes the description of the various operations performed on data, such as transfer, indexing, archiving, replication, processing, deletion…DLM has several advantages, which include:

- allowing people to fulfill the requirements of each scientific sector so that data storage can be performed.
- ensuring a good infrastructure for data protection in case of unexpected risk or emergency.
- ensuring appropriate extraction, curation, maintenance and update of information throughout the data cycle.
- allowing availability of useful, clean and accurate data to all users, and thus increasing the agility and efficiency of the scientific studies.

DLM faces three main challenges. The first one is managing a massive volume of data for storage, which requires distributed storage supports and parallel processing [12]. The second challenge is linked to the highly dynamic character of the data, i.e., the fact that they may be incrementally produced, modified, or temporarily unavailable. The third challenge is related to the fact that data are produced and spread across a large variety of infrastructures and systems.

Percolator [13] is an example of a language and its implementation, which considers the arrival of the data, and incrementally updates, from the modification of the data sets, the result of a computational process. Percolator allows distributed incremental processing on the base of an observer/notifier paradigm. This design choice means that a set of Percolator workers can look for changed columns in a BigTable [14], and execute trigger-like procedures (called "observers") to update the previously computed results. Presto [15] is a distributed implementation of the R language, with incremental features and a semantic which is close to the one of Percolator. Presto uses HBase [16] as a storage backend, and it allows programs to attach callbacks to arrays of data (or partitions of an array).

To reduce the complexity of the data life cycle management, authors in [17–19] propose 'Active Data', a programming model for automating and improving data management applications. They introduce the concept of data life cycle in Information Technologies and define a formal model based on Petri Net [20]. Then, they present the concept of the Active Data programming model, which allows code execution at each stage of the data life cycle. With Active Data, routines provided by programmers are executed each time a given piece of data undergoes an event such as creation, replication, transfer, or deletion. Authors implement [19] and evaluate their solution in three use cases, which altogether illustrate the adequateness of the tool to manage distributed and dynamic data.

From an industrial point of view, we would like to mention Kensu's Data Intelligence Manager[5]. This solution integrates data science and artificial intelligence, and monitors, identifies, checks, and provides real-time insights into how sensitive data are managed in organizations. A single panel of glass observation and comprehensive reporting solve the currently manual process of identifying, tracking, and updating thousands or millions of records across various systems, tools, and data repositories.

2.4 Validation in Computer Science

The internal state of the emulation mechanism may not accurately reflect the internal state of the target being emulated. In our case, we use ontologies for representing the concrete elements of the buildings (walls, rooms, sensors…), and a dedicated tool for managing the life cycle of data captured inside the building. Both the building ontologies and the

[5]https://www.kensu.io/

data life cycle tools are used for representing the targeted building.

Simulation, on the other hand, involves modeling the underlying states of the target. We do not provide a simulation of a building by software, albeit this technique may be of particular interest in many contexts.

Both emulation and simulation techniques are used for the validation of software systems developed by engineers or researchers. Classical approaches for validation in sciences and computer science rely on either formal methods (equations, proofs, etc.) or experimental methods using scientific instruments. In this paper, we develop an emulation tool for describing a building and its various components and use an experimental method [10] for validating our vision.

We assume that formal validation is untractable or unsuitable in our case, since we are looking for a vision of a building as close as possible to reality. Our emulation of a building may serve as a testbed for studying the ecosystem of smart buildings, and this constitutes an original approach.

2.5 Translational research in computer science

The notion of experimental validation refers to the broader framework of the fundamental characteristics of translational research. In short, it relates to the scientific roadmap in terms of the type of research we are conducting.

In the seminal paper entitled Translational Research in Computer Science [21], David Abramson, and Manish Parashar noticed that "Recently, there have been dramatic changes in the nature of research-application workflows. These are driven, in part, by the greater availability and increasing scales of experimental and observational data, the ability to model phenomena more holistically, and the desire for near-real-time data processing and actuation.

There have also been dramatic changes in the technology and resource landscape, complementing this change in application workflows". The objective of the paper is to formalize the translation process for computer science research.

Inspired by the concepts behind the discipline of translational medicine (TM) and its tremendous impact, authors explore translational research in computer science (TCS) as a viable discipline with its own research processes and research agenda.

TM is defined as an *"interdisciplinary branch of the biomedical field supported by three main pillars: bench (basic research), bedside (clinical research/trials), and community (new practices in patient care)."*

The goal of TM is to combine disciplines, resources, expertise, and techniques within these pillars to promote enhancements in prevention, diagnosis, and therapies". TM/TCS ideas are inspiring our work, in particular for the definition of the full scientific roadmap (see subsection 5). For the impatient, complementary readings on translational research are [22] and [23].

3 ARCHITECTURE OF THE EMULATION TOOL

This section completes the arguments given in the previous section, by specializing them, and in terms of the technologies we need to architect the software solution.

3.1 General overview

With the development of the Internet of Things (IoT), the data generated by a building have become more dynamic and complex to capture and merge. Any building can be enhanced with information from events, sensors, and environment, which all make a source of data for possible innovative services.

Event information concerns the different events that may occur in the building environment, and produces data such as their location, their time of occurrence, the agents involved, the relation to other events, and their consequences.

Sensor information arises from the different sensing devices deployed in the building, and also concerns the processing or treatment implemented on the collected data.

Environment information details the building structure, its topology, and the location of various elements (sensors, actuators, smartphones, other devices) in the different rooms.

The integration of all those elements is required for completely and correctly understanding, planning, managing and monitoring the building. To deal with these goals, this paper presents ongoing research for studying and modeling realistic building ecosystems.

Typical queries in this context about data may be as follows:

- Space Query: the query deals with geometric information about elements of the building;
- Space and Time Query: the query deals with environmental parameters that impact a building (sunlight, humidity, air quality, etc…), such as "What is the relative air quality of the rooms located at the first floor?";
- Learnable Space and Time Query: the term 'learnable' means that the query is based on concepts that evolved computationally (via induction or deduction) over historical data. Such a query should ideally reason in terms of information flows (also called data streams) rather than on given stacks of data, produced by sensors and further analyzed through data mining operations. An example of such a query is: "Which garage rooms are most likely available for parking?" The answer is not only dependent on the structure and state

of the building, but also on the pattern and trend of availability of the parking rooms;

Note that for the last category of query, the reader can follow the ideas of Valiant in [24]. Moreover, both unsupervised and supervised learning work in a complementary way. Our long-term objective is to settle a software architecture that facilitates studying novel learning algorithms and validate them experimentally. In a supervised algorithm, we need the 'ground truth' to build the model, whereas, in the framework of an unsupervised algorithm, we do not need the 'ground truth' to build the model. As described below, our software architecture includes global and local learning algorithms for dealing with the data life cycles. This association is one of the most valuable and original characteristics of our tool.

3.2 Managing ontologies

Ontologies in our work are used to represent the physical elements of a building. Ontologies are increasingly becoming essential tools for solving problems in many research areas. An ontology is a complex object for information because it often includes and is represented by hierarchies among many terms. Managing complex information objects requires using information systems technology. An information system dedicated to managing ontologies is called an ontology server.

Contemporary ontology servers share many structural similarities, regardless of the computer language in which they are expressed. Most ontologies describe individuals (instances), classes (concepts), attributes, and relations.

Nowadays, most ontology servers provide a user graphic interface for defining and selecting individual ontologies. They also include deductive classifiers that help check the consistency of various models of ontologies and infer new information from the analysis of a given ontology. The Web Ontology Language (OWL) allows authoring ontologies managed by servers. Moreover, OWL is a family of languages for knowledge representation, characterized by formal semantics. Formal languages are built upon a common standard for objects, the World Wide Web Consortium's (W3C) XML Resource Description Framework (RDF) [25].

SPARQL[6] is a popular query language and protocol released by the W3C RDF Data Access Working Group. It allows to search, add, modify and/or delete any RDF data available through the Internet.

3.3 Managing the data life cycle

We also need a model for managing the life cycles of all data produced inside the building, to later be able to learn from those data. The ultimate goals are to improve the building management system, and to develop new advanced applications for the residents (such as safety, health, gaming, efficient water/electricity management, sustainability).

We aim at developing mechanisms that react to events. They indeed help us to carry out data curation and deletion operations, and also to control learning from the data. Every single operation might ideally be highly customizable so that any single operator might plugin his own new learning algorithm into the software architecture.

3.4 Our software architecture

The combination of ontology and data life cycle managers leads to the software architecture illustrated in Figure 1. Our emulation tool is built onto three components: an ontology manager, a data life cycles manager (both described above), and a 'data lake.' We need at this stage a mechanism to check the coherency between the ontologies and the data life cycle models. Such a sub-component would for instance, aim to check that every sensor (each described by one instance in an ontology) is included in every model of the data life cycle.

Both the ontology manager and the data life cycle manager generate data which are stored in the shared resource called 'data lake'. Researchers and users can also make a query on the data lake, for visualizing or inferring knowledge about the smart buildings. Models for the Data Life Cycles can be also be inserted or deleted... Queries about ontologies that represent the physical building can also be carried out for the insertion or deletion of new components (devices), for instance.

The ultimate goal is to deploy one instance of the emulation tool per building. The rationale is to keep the data inside the building and to learn only on these data. We do not yet envision, in the paper, the possibility to learn from different buildings by exchanging information between buildings which is a challenging task.

As explained in the related work section, there is no available tool, to the best of our knowledge, to manage simultaneously physical data (wall, corridors, sensors...) and stream data coming from the live utilization of buildings by residents. Again, we postulate that we need to provide a system able to manage and to analysis data in-situ.

4 EXPERIMENTAL WORK

This section is devoted to selecting and integrating existing tools to form the software solution as a whole. We first present our selection of ontologies, then the chosen tool to manage the life cycle of the data generated by the sensors. We also give an example. Finally, we present the database tool's choice containing structured and semi-structured data

[6] https://www.w3.org/wiki/SparqlImplementations

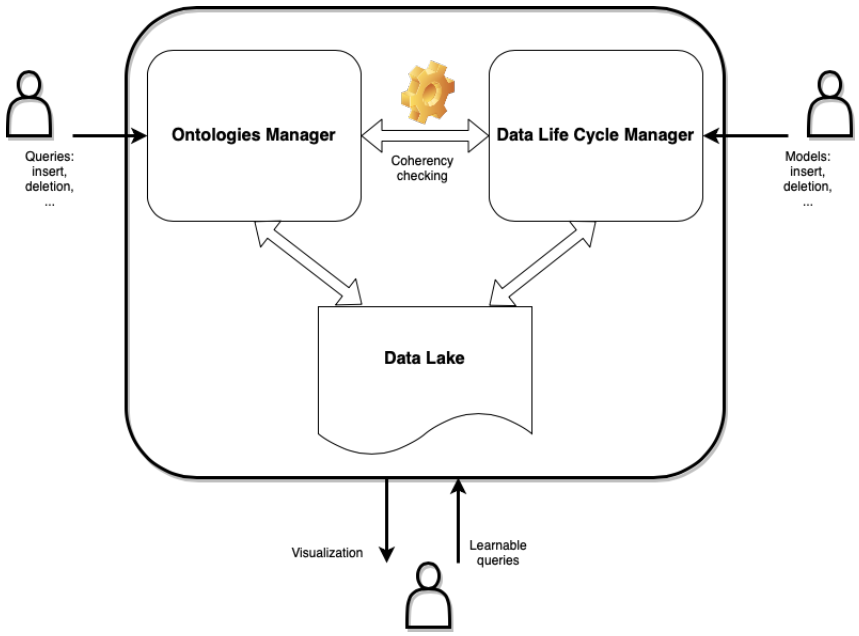

Figure 1: Software architecture of the emulation tool.

(ontologies and life cycle). Then, we introduce examples regarding the possibilities offered by the selected tool.

4.1 Protégé software and our domain ontologies

A domain ontology (or domain-specific ontology) represents concepts related to a specific field or domain, which belongs to the real life or to scientific and general knowledge. Each domain ontology typically describes domain-specific definitions of various terms, about buildings and sensors in our case.

Regarding the management of ontologies, our work considers the well-established Protégé system [26]. Protégé is a free, open-source ontology editor and an ontology manager. Like Eclipse, Protégé is a software framework for which various projects suggest plugins. Protégé is written in Java and makes heavy use of Swing to create the user interface. Note that Oracle does not anymore support Swing, which has been replaced by the JavaFX technology[7].

Table 1, shows the ontologies related to the building domain, that we reviewed for our tool. We needed to set up a hierarchy before merging several ontologies over Protégé. The ontologies were selected to cover most of the required concepts of the different domains (construction, sensors), particularly a Building Information Modeling (BIM) ontology.

[7]https://github.com/openjdk/jfx

Let us focus on ontologies for Building Information Modeling (BIM). BIM is an intelligent 3D process that provides information and tools needed to plan, design, build, and manage buildings and physical infrastructures more efficiently. It is widely used by architects as well as engineering and construction professionals.

The BIM benefits are that it allows the design of and documentation of buildings. All pieces of information about a given infrastructure are included and modeled in the BIM. The model can analyze different variants of a project, and it allows visualizations that help to understand the whole building before its construction better. The model is also used to generate the documentation and guide for the construction. Note that, in Table 1, as for BIM ontologies, we use the Industry Foundation Classes (IFC) [30], which are standards for representing buildings and construction data (with instances for the structural elements of the building, its topology and the different elements it may contain).

Table 1 lists the different domains related to smart buildings and the few ontologies we studied for each domain before designing our emulation tool. Ontologies are mainly related to BIM, sensors, and actuators, persons, to time and space. The DUL ontology provides a set of concepts about interoperability between various ontologies [31]. It also underlines the properties required for combining spatial information with various kinds of data. The Event ontology deals with the notions of events, and it provides most of the vocabulary required to describe their different characteristics, such as location, time, agents, factors, and products [32].

We conducted a pairwise comparison of the ontologies listed in Table 1 to refine our final choice before integrating them in the ontology manager of our emulation tool. The options are as follows:

- **BIM Ontologies:** IFC4, IFC4_ADD1, IFC4_ADD2 are identical. As for IFC2X3_TC1 and IFC2X3_FINAL we selected IFC2X3_FINAL because it includes the IfcBinary subclass. Between IFC4_ADD2 and IFC2X3_FINAL we selected IFC2X3_FINAL because it is more general;
- **Sensors/Actuators ontologies:** we kept both ontologies because SSN is specialized for sensors and SOSA for actuators;
- **Time/Space ontologies:** the Geo ontology includes the spatial 'thing' class which is also present in the Event ontology. Moreover, time classes are all included into the Timeline ontology;
- **Person ontologies:** they are all identical, so we kept the RDF format;
- **Others:** the RDF schema has 3 classes that appeared not to be useful for our needs (Container, ContainerMembershipProperty, Literal).

The ontologies that we finally selected are presented on Figure 2. They are also available on GitHub[8]. In short, the hierarchy for some of our selected ontologies is as follows : a building includes components (doors, windows, also referred to as 'zones',...), zones include spaces, in spaces we have persons, sensors and actuators.

4.2 Active Data

As explained above in the 'Related Works' section, Active Data [17–19] is a data management system that participates in the life cycle of the data. Data management systems store, transfer, index, and query data. They perform the necessary maintenance operations before processing tasks. A massively distributed application involves including several data management systems, each one performing one or several tasks. In short, Active Data allows the specifications, i.e., what we have to do to trace the data, according to the Petri Net modeling approach [20]. It also attaches handlers to programming actions on data. To illustrate our work with Active Data, let us consider the following concrete example.

In Covid 19 times, our purpose is to allow efficient ventilation of the room(s) in a building (professional offices, hospital services, retirement homes, nurseries, teaching premises - from kindergarten to University -, individual apartments,...). The ventilation must be carried out in good understanding with the different users and fulfill sanitary security and sober energy consumption requirements. In this scenario, the data to be captured through sensors and actuators in the building could be the followings:

- Location and identification of the room (via a QR code and/or a mobile application through GPS);
- The CO_2 concentration in the ambient air (definition of acceptable threshold(s), determination of measurements duration and frequency, other meta-data related to signal acquisition - see CO_2 sensor technology...);
- The state of the windows (open or closed), and the window opening parameters such as amplitude, tilt and turn mode, duration,...;
- The room temperature (eventually that of the outside?);

A simplified version of the above scenario that we implemented with Active Data deals with the data produced by a CO_2 sensor in a given room. The ultimate decision, in the life cycle of data related to Co_2 ambient air concentration measurements, consists in deciding whether or not heat the room, after windows have been opened and closed to lower back the CO_2 rate to values warrantying safe/correct air refreshment.

Generally, it is common to apply throttling on the data before pushing them on a computing platform and further archiving them and the associated results. Throttling the data may consist in filtering, compressing and/or curating (i.e., looking for and deleting invalid data). Once the data throttling has been performed, we need to decide whether to open/close the windows in the room and ultimately increase or decrease the room temperature. This decision concludes the data life cycle.

The corresponding life cycle is diagrammed in Figure 3. The CREATED (p_1) and TERMINATED (p_9) places are the initial and final places for the life cycle of the CO_2 rate measurement data in a given room. The LOCATION place (p_2) serves to check if we are in the correct room. The FILTER (p_3), COMPRESS (p_4), CURATION (p_5) places converge to the THROTTLING place (p_6) and mimic some filtering, compression, or curation over the measured value for CO_2. The DECIDE place (p_7) models the decision taken, i.e., to increase or decrease the room temperature. The decision leads to the BEING-HEATING place (p_8).

For more practical details, the reader is invited to read the tutorial available online[9].

Nowadays, computer applications are developed according to multiple programming languages. To develop data handlers in various programming languages, we rewrote the Active Data recipe in order to use GraalVM. This high-performance polyglot runtime provides significant improvements in the performance and efficiency of applications, and is ideal for micro-services. The Active Data recipe for GraalVM, as well as full examples are available online[10].

[8] See https://github.com/imtiaz-abdullah/IUTONTOLOGY

[9] http://lipn.univ-paris13.fr/~cerin/HDU/ActiveData-GraalVM.html

[10] https://github.com/christophe-cerin/ and https://www.graalvm.org/

Table 1: Reviewed ontologies for building related domains

BIM	Person	Sensors	Time/Space	Others
IFC4	Person	SSN	Time	Prov
IFC4_ADD1	Person.ttl	SOSA	Timeline	Foaf
IFC4_ADD2			Geo	Schema
IFC2X3_TC1			Event	RDF schema
IFC2X3_FINAL			DUL	

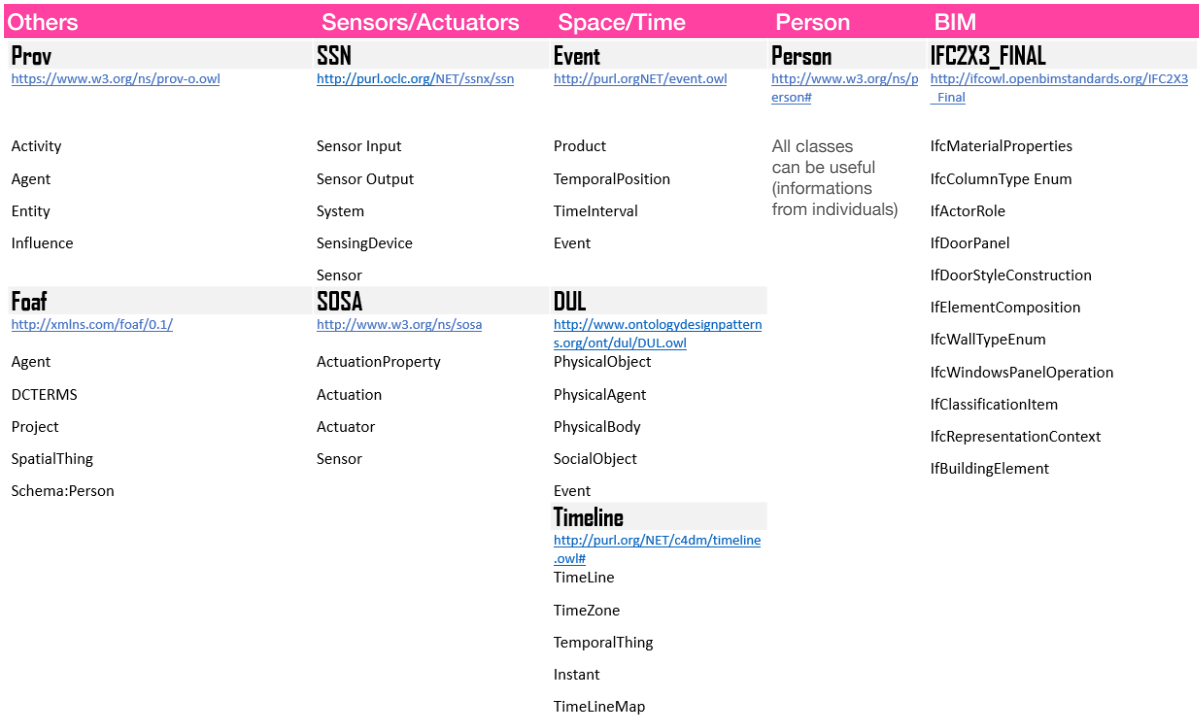

Figure 2: Selected ontologies and provenance.

4.3 Data lake

The implementation of the data lake depends on the requirements we propose to fulfill, among them the distributed foundation for the data base, the integration of ontologies, the integration of data life cycle models, and the storage of codes for the handlers associated to the sensors and actuators.

4.3.1 Distributed foundation for the data base system. Nowadays, people recognize that, to a first approximation, all developers are cloud developers, and all applications are cloud-native and all operations are cloud-first.

Major cloud providers such as Google and Amazon provide data-stores, according to the SQL or NoSQL paradigms. Databases are distributed over the physical infrastructure for fault tolerance issues or load balancing issues. Cloud providers also provide facilities in using popular Big Data systems. For instance, Amazon EMR is a leading cloud-based Big Data platform dedicated to processing large amounts of data using open source tools such as Apache Spark, Apache Hive, Apache HBase, Apache Flink, Apache Hudi, and Presto.

Based on these beliefs and observations, it becomes essential to choose a database manager that integrates with the cloud. As an example, we would like to mention the CockroachDB project [34]. CockroachDB is a scalable SQL DBMS built from the ground up to support these global OLTP workloads while maintaining high availability and strong consistency. Its novel transaction protocol supports performant geo-distributed transactions that can span multiple

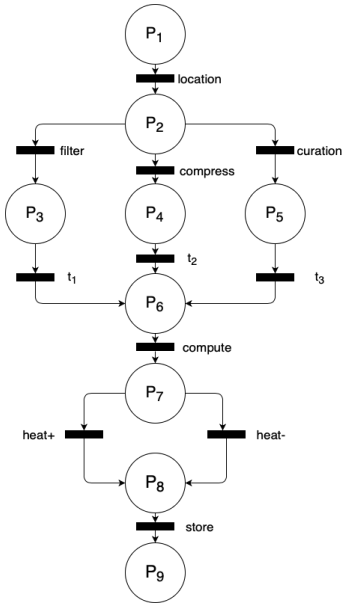

Figure 3: Example of Active Data specifications scheme for the life cycle of CO_2 detection data.

partitions. It provides serializable isolation using no specialized hardware; a standard clock synchronization mechanism such as NTP is sufficient. As a result, CockroachDB can be run on off-the-shelf servers, including public and private clouds.

A comparison between CockroachDB, MongoDB, PostgresSQL, and Cassandra is available[11]. The comparison shows that the main features of CockroachDB are that the database schema may change (Online, Active and Dynamic), data are geo-partitioning, and it uses multi-clouds. Moreover, CockroachDB is wire compatible with PostgreSQL, and it supports JSON[12]. In this last case, the JSONB[13] datatype stores JSON data as a binary representation of the JSONB value, which eliminates whitespace, duplicate keys, and key ordering.

Indeed, CockroachDB implements the NewSQL paradigm [27–29]. NewSQL is a class of relational database management systems that seek to provide the scalability of NoSQL systems for online transaction processing (OLTP) workloads while maintaining the ACID (Atomicity, Consistency, Isolation, Durability) guarantees of a traditional database system.

CockroachDB is representative of NewSQL tools because it has the additional benefits of NoSQL databases that provide the ability to use semi-structured data. In other words, CockroachDB has options for managing rapid development, data without an explicit schema, or the schema you do not control entirely.

4.3.2 Integration of ontologies in the data base system. A database manager, we mean a relational database manager, is not enough because we also need to manage ontologies, for instance, according to the RDF format. To the best of our knowledge, the maturity of tools transforming RDF representation to SQL is not so high, despite the existence of tools such as rdf2sql[14] and rdfto2ml[15]. A more mature project is the easyrdf[16] project. Easyrdf can export RDF into the JSON (JavaScript Object Notation), which is a lightweight data-interchange format. Moreover, JSON is a popular format for managing hierarchical information.

Imagine you are dealing with a data building model[17] that includes a table in which we use JSONB to store meta-data about the material of the roof:

```
--
-- CREATE TABLE: BUILDINGSROOFMAT
-- Represent the table material of roof
--
CREATE TABLE BUILDINGSROOFMAT (
        UUID  VARCHAR2(70) NOT NULL,
        ROOFMAT  VARCHAR2(80) NOT NULL,
        metadata JSONB
);
```

Within the metadata field, we might store something like:

```
{
  "observations": ["fissures", "foam"]
}
```

After our initial launch, we checked in with a few of our users and learned that they would like to insert the year for the next visit of the roof. Since we are using JSON, we do not need to commit a schema change. Instead, we can insert a new data field:

```
{
   "next-visit": "2022",
   "observations": ["fissures", "foam"]
}
```

4.3.3 Integration of the data life cycle models in the data base system. We propose, in this paper, to use automata as data models for data life cycles. We have illustrated our concerns with the ActiveData [17–19] framework.

[11] https://www.cockroachlabs.com/docs/stable/cockroachdb-in-comparison.html
[12] https://www.cockroachlabs.com/docs/v20.2/demo-json-support.html
[13] https://www.cockroachlabs.com/docs/v20.2/jsonb.html
[14] http://dig.csail.mit.edu/2009/IARPA-PIR/test/air-server/SUT/sutlite/rdf2sql.py
[15] http://www.semwebtech.org/rdf2xml/
[16] https://www.easyrdf.org/converter
[17] https://github.com/GeoSmartCity-CIP/Buildings-SQL-Data-Model/blob/master/GSC_Building_2.1e_Oracle.sql

A couple of tools exists for representing automata. PySimpleAutomata[18] is a Python library to manage Deterministic Finite Automata (DFA), Nondeterministic Finite Automata(NFA) and Alternate Finite state automata on Word (AFW). This library is not meant for performance nor space consumption optimization, but for academic purposes: PySimpleAutomata aims to be an easily readable but working representation of automata theory. According to the documentation, the JSON format can depict automata.

The JSON Finite State Machine (JFMS) Representation, Validation and Generation project[19] is yet another project related to the JSON representations for automata. In this project, users can handle mathematically defined automata such as Mealy, Moore, or Mixed models. The project home page also points to alternative projects regarding state machine notation for control abstraction.

All these projects are candidate projects for inclusion in our emulator project. Many of them are based on the JSON format. Thus they will be natively supported by CockroachDB. Note that if a developer likes all of these automata formats, we have to store different information. Having JSONB columns available lets us model this scenario without resorting to a complex table structure with many foreign keys or redundant columns.

4.3.4 What options do we have? Considering that a pure SQL or NoSQL database manager is not sufficient to capture all our project requirements, we need to consider NewSQL tools for the implementation of our data lake. CockroachDB is one possibility. Moreover, we also need to consider, at least, available drivers for the development of the emulator, data access frameworks (e.g., ORMs), application frameworks, graphical user interfaces (GUI), integrated development environments (IDEs), and schema migration tools.

Regarding CockroachDB, the third-party database tools are discussed on a dedicated Web page[20]. We can consider that they are all mature and witnesses of a great diversity for the imaginable developments.

For more practical details regarding the integration of ontologies and data life cycles with CockroachDB, to form the "data lake," the reader is invited to browse the tutorial available online[21]. The tutorial highlights the general principle of keeping the software architecture as simple as possible. In our case, we mean to use only one tool for the management of structured and semi-structured data. The tutorial also gives the current status of our implementation work for the "data lake" component.

[18]https://pysimpleautomata.readthedocs.io/en/latest/index.html
[19]https://github.com/ryankurte/jfsm
[20]https://www.cockroachlabs.com/docs/v20.2/third-party-database-tools
[21]http://lipn.univ-paris13.fr/~cerin/HDU/Cockroach.html

5 DISCUSSION

5.1 Research questions and issues

Questions and issues 1: It should not be difficult to measure the emulator's realism and durability of such an emulator. It must be based on data sets that reflect the variety and mass of data from smart buildings. It is, therefore, necessary to describe how the emulator would be powered. It is, therefore, likely that a large consortium of actors would have to be mobilized to provide this mass and variety of data. The methodology must be clear on the validation of the results obtained.

Answers. The methodology we propose to use is through case studies. Later on in the paper (see subsection 5.4), we give details of one of them and the data provenance. Another simple use case capitalizes on our tool in the context of Covid-19. The objective is to propose efficient ventilation of the room(s) in a building (professional offices, hospital services, retirement homes, nurseries, teaching premises - from kindergarten to University -, individual apartments). The ventilation must be carried out in good understanding with the different users and fulfill sanitary security and sober energy consumption requirements. The university Sorbonne Paris Nord is currently deploying CO_2 sensors in March 2021. We could reuse the physical infrastructure for our use case. We also point out, later on, that an effort should be made on the standardization point of view, both in terms of ontologies and queries. There is no standardized benchmark for 'querying' a smart building data set to the best of our knowledge. We probably need expertise from major industrial actors in the field.

Questions and issues 2: It should not miss a precise positioning concerning existing ontologies or the work on data life cycle management that may have been carried out in or outside smart buildings' environments. Similarly, it should not be difficult to understand whether the proposed work is of a development nature or whether significant design work is expected (on ontologies, data life-cycle management...).

Answers. Our approach goes beyond the smart building approach as carried out by the CSTB institute and the SmartBuilding Alliance[22] because we are not limited to the aspects of data modeling of a smart building. So, significant design work is needed. We propose, in this paper, some directions to fulfill some of the requirements.

Most ontology servers provide a user graphic interface for defining and selecting individual ontologies. They also include deductive classifiers that help check the consistency of various models of ontologies and infer new information from the analysis of a given ontology. The Web Ontology Language (OWL) allows authoring ontologies managed by

[22]see http://www.cstb.fr/en/ and https://www.smartbuildingsalliance.org/

servers. The first issue and strategy are to reuse, as much as possible, existing ontologies or to enrich some of them, if necessary. As the second issue, we also need dedicated action to study the interactions with the data life cycle manager, for instance, to check the consistency between the two representations (the sensor as an instance of one ontology and the sensor as an instance of a data life cycle). The scientific obstacle of this WP is in the inability to do a mapping between an object in one 'semantic world' with an object in the other 'semantic world,' thus breaking any possibility of global reasoning, for instance, for proof checking.

We also need to tackle the model checking of scenarios for data life cycle management, specified, for instance, with the Active Data programming model, based on using formal modeling approaches such as Petri nets. The scenarios specify how to store, transfer, index, trace, and query data. Scenarios also attach handlers to programming actions on data. Such handlers use various programming languages to carry out the intended activities. Thus, the complex behavioral model obtained, combining control and data flows, must thus exhibit correctness and safety, and liveness properties. The scientific obstacle that the ongoing work will overcome is the automated, continuous, and incremental verification of the data life cycle management programming model. The verification allows code execution at each stage of the data life cycle and handles highly dynamic data incrementally produced, modified, or temporarily unavailable.

Questions and issues 3. Beyond a proof of concept, it is essential to demonstrate that the approaches developed should meet the requirements, the functional requirements, the performance requirements.

Answers. The scientific lock is in specifying the interactions between the three components, and the technological lock is in choosing existing middleware or developing some extensions of existing ones. Since our rationale is in promoting the idea that the "smart building behaves like a datacenter", our tool for emulating smart buildings has the potential to allow, firstly, to learn and plan from data produced locally (first requirement). Based on our tool, the communities working in the field will improve building management systems and develop advanced applications for safety, health, gaming, efficient water or electricity management, and sustainability (functional requirements).

Our approach also has the potential for data scientists to optimize machine-learning kernels for the ecosystem of smart buildings, especially for the heterogeneous computing systems present in a building such as tablets, smartphones, washing machines, TV sets, fridges, boilers, heaters, all of them equipped with processors. To the best of our knowledge, there are not yet machine learning libraries to efficiently maximize the utilization of hardware on these devices (performance requirements). In short, the development will be guided simultaneously by the different types of requirements.

5.2 Further work

As introduced in Figure 1, the 'Data Lake' is not fully implemented. One technical problem is to select the most convenient system for the storage and retrieval of data modeled in our study. Since we manage different data types (ontologies, data life cycles), the choice between relational databases and NoSQL databases is not apparent.

Moreover, a new class of distributed database systems, namely the NewSQL systems [27–29], appears promising because it combines the scalability and availability of NoSQL with the consistency and usability of SQL. Further investigations are necessary to clarify the situation regarding our needs, in particular, into the direction of the development tool-chain.

5.3 Relevance

We believe that making computer scientists and domain-specific scientists work together will help solve big data problems from real life. For this purpose, we propose integrating ontologies and their concepts, on the one hand, and of data life cycles, on the other one, in a unique emulation tool. We assume that the challenging part for domain scientists will help understand the places and transitions that trigger actions to be performed on the data. Managing ontologies is probably more 'natural' and easier since ontology writing does not involve any programming style with loops, guards, conditionals. As computer scientists, we should work and describe our actions so that Active Data does not constitute a barrier to understanding and adopting our emulation tool by domain scientists.

From a computer scientist's point of view, the most challenging part is in the specifications of the modeling and in composing multiple Active Data specifications. At present, we hypothesize that all the Active Data specifications are independent one from the other. Such a hypothesis means that sensors and actuators behave independently and that they do not share any information. Nevertheless, in some situations, we think that all sensors may send information to a shared database. The database then becomes an element shared between all Active Data specifications. If the number of sensors increases a lot, we will need an automatic way to compose the specifications because it will be too complicated for human intelligence.

5.4 Translational research

The model we follow in this research is a variation of the translational research model in computer science [21]. The

first goal of translational research in computer science is to address the complexity of computer systems growing complexity and ensure that the emerging opportunities (proliferation of cloud services, novel storage facilities, the proliferation of accelerators, deployment of high-bandwidth/low-latency networks…) are translated into insights, discoveries, and innovations. The second goal is to go beyond the traditional basic and applied research paradigms. Although applied computer science may be application-driven, there is actually no requirement to apply it to real problems. In general, once the research project terminates, the translation is usually outside the original research plan. However, the whole research process must include the ability to test the ideas at scale with real-world constraints as part of the research plan, i.e., the application of new knowledge is an implicit part of the research plan.

We are carrying out two projects that will help to investigating and experiencing with our new tool. One project, related to smart buildings, has been selected this year with Rêve de Scènes Urbaines (RSU), an industrial demonstrator for sustainable cities located in the city of St Denis (department 93 in the north of Paris). This non-funded project is conducted in partnership with the industrial companies Qarnot Computing and Digital & Ethics, and will take part in the renovation of the Institute of Technology (IUT) building at St Denis. In terms of "demonstration," the project aims to deploy a multiple sensor infrastructure inside the IUT building and collect data using the Qarnot OASIS tool. Real data coming from the whole building will be available for the test of our emulation model. We were also asked to define an economic model for the re-use of data and will be helped with this issue by a network of outstanding companies that are members of the RSU initiative.

A second project has been submitted and recently accepted as a use case to the ISO/IEC JTC 1 SC 42 Artificial Intelligence – Working Group 4. It is related to the question "How can we build ontologies from and for data coming from smart buildings?". From our point of view, it is essential to also interact with standardization committees in AI. Indeed, standardization can maximize the compatibility, interoperability, safety, reproducibility, and general quality of the different individual ontologies under construction.

These two initiatives complement our initial research plan from the perspective of socio-technical system [33], considering not only technical issues but also human and community factors. In [22], Foster also notices this practice in his work in Grid computing. In other words, we organize our research as translational computer science because we work with domain scientists, application developers, providers of computing power in buildings (Qarnot), and evaluation through standardization.

TCS implies a need for a locale into which research results are to be translated. In our case, the locale is the IUT university building, under the renovation project framework. The renovation project permits researchers to test their ideas at scale with real-world constraints.

If the coupling between the intended research result and target translation locale is lacking, then a translation can fail. This observation is not always detrimental to the research project as it may only reflect an inability to achieve the translation by the means envisaged. In this case, one should consider other means. We have eliminated at least one of the possibilities.

Let us examine some projects related to smart living under the umbrella of the Gaia-X project[23]. GAIA-X is a project initiated by Europe for Europe and beyond. It aims to develop standard requirements for a European data infrastructure. To quote the project's rationale as it is available on the project's home page, we can say that *the objective is to build a secure, federated system that meets the highest standards of digital sovereignty while promoting innovation. The project is the cradle of an open, transparent digital ecosystem, where data and services can be made available, collated, and shared in an environment of trust.* In this context, trust means GDPR[24] and data infrastructure means a federation of clusters.

Three projects related to smart living have been submitted and accepted to be part of Gaia-X. A common goal is to leverage suitable standardization requirements for linking the growing volumes of data and thus promotes the emergence of further AI applications, mainly through cooperation between the digital economy, housing industry and electrical industry. We can also imagine that the evaluation metric is a trust metric (GDPR). The projects are related to energy efficiency, security, and ambient assisted living to help the person concerned maintain their independence and quality of life, despite health limitations.

Even if applicative motivations guide all these projects, we cannot say that a translational science approach guides them. Our analysis may be biased because it is only based on the online Gaia-X project page's information.

However, we do not see a schedule either for basic research associated with the envisaged applications or for 'clinical' studies. The Gaia-X infrastructure can undoubtedly help, but it is not sufficient to enter a translational approach. On the fundamental research side, we propose, in this article, to equip ourselves with an intelligent building emulator, as part of a system-level science, i.e., science that requires the integration of diverse sources of knowledge about the constituent parts of a complex system to understand the

[23]https://www.data-infrastructure.eu/GAIAX/Navigation/EN/Home/home.html

[24]https://ec.europa.eu/info/law/law-topic/data-protection_en

systems properties as a whole. Thus, experimental methods will be at the service of scientific activities.

6 CONCLUSION

In this paper, we propose experimentally studying the ecosystem of buildings, and we create a testbed for 'smart' buildings. The testbed aims to observe and measure the behavior of a building in conditions that can be configured and controlled. We propose as a testbed a new software architecture, i.e., an 'emulation tool', that considers and experimentally studies buildings as data centers. Data are produced in a given building by sensors and actuators and new devices such as the Qarnot heaters and boilers that possess computing power.

We intend to keep the data inside the building as much as possible rather than host them on external centralized GAFAM data centers. We promote in our emulation tool the integration of both ontology managers and data life cycle managers. Users will describe and query details on the building through ontologies and get precise information on the building data flow - what, when, how, how much - through the data life cycles.

We also promote translational research methodology, which we believe is better for our research than basic and applied research methodologies. Indeed, we wish to integrate the utility and application of the work into our research plan. We want to invite communities with their own high-level goals and objectives to join this effort by adhering to a common goal for studying smart buildings.

We are aware that introducing more examples about different areas of the building interacting with different sensors would better shape and illustrate the potential of our proposal. However, we have chosen to keep the examples as simple as possible not to complicate the guidelines. The given considerations touched upon several research and application domains, the integration of these domains is a good step towards solving the discussed problem.

From a technical point perspective, future work aims to fully integrate within the emulation tool the so-called 'data lake' component and the already designed ontology and data life cycle managers. We also want to become able to master multiple interconnected data life cycles. For this purpose, we need to define mechanisms of composition between the data life cycles. As previously said, we also need a coherency mechanism to ensure that every sensor managed by the ontology's manager, produces data, whose life cycle is also managed by the data life cycle manager. Last but not least, we want to use both our 'Rêve de Scènes Urbaines RSU' and 'ISO/IEC JTC 1 SC 42' Projects to 'feed' our new tool with data sets and validate it experimentally.

In the long term, we also plan to investigate the collaboration of buildings with the possibility of exchanging data. The objective will be to imagine a system to manage a federation of buildings; each one will run our tool. In this context, we also have legal issues (improved privacy and security; sovereignty; governance[25]; what and when to share) above the technical issues to devise an exchange protocol and to query distributed data. Standards are also important in the context of smart buildings, AI, and systems regarding data management. We plan to work according to work and best practices presented through the Standard Library Web site[26] as a source of documentation and for all levels of the software stack.

ACKNOWLEDGMENTS

We would like to thank the National Institute of Informatics (NII), the Centre National de la Recherche Scientifique (CNRS), the Université Sorbonne Paris Nord (UPSN) and Rêve de Scènes Urbaines (RSU) in supporting this work. We also thanks Abdullah Imtiaz from IUT Villetaneuse, Angan Mitra from Qarnot Computing, Tarek Menouer and Patrice Darmon from Umanis, for their suggestions.

REFERENCES

[1] T. Hoffmann, "Smart Buildings," Johnson Controls, Inc., pp. 1-8, October, 2009. Available at https://www.scribd.com/document/259029136/Smart-Buildings

[2] V.M. Larios, J.G. Robledo, L. Gómez,and R. Rincon, "IEEE-GDL CCD Smart Buildings Introduction", white paper of the working group of physical infrastructure available online at https://smartcities.ieee.org/images/files/pdf/whitepaper_phi_smart-buildingsv6.pdf

[3] A.H. Buckman M. Mayfield Stephen B.M. Beck, "What is a Smart Building?", Smart and Sustainable Built Environment, Vol. 3 Iss 2 pp. 92 - 109, (2014), 10.1108/SASBE-01-2014-0003

[4] Siemens, "Smart buildings - Striving for the perfect place", White paper avail online https://new.siemens.com/global/en/products/buildings/smart-buildings.html

[5] Intel, "Designing More Affordable Smart Buildings Solutions", white paper avail at https://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/iot-smart-building-solutions-brief.pdf

[6] R. Bernstein, "Building Automation Training and LonMark Certification Institute Programs" https://www.lonmark.org/connection/presentations/2017/AHR/-Session%202/Session%202%20-%20Ron%20Bernstien%20Smart%20-Buildings%20Course%20101%20-%20Key%20Concepts,%20Definitions%20and%20Elements.pdf

[7] Residential Building Systems Repository. Check the Website on https://homes.lbl.gov/publications

[8] R. Marroquin, J. Dubois, C. Nicolle, Multiple Ontology Binding in a Smart Building Environment, LDAC2017 - 5th Linked Data in Architecture and Construction Workshop (13 - 15 Nov. 2017), http://linkedbuildingdata.net/ldac2017/

[25] https://www.meti.go.jp/English/press/2021/0115_006.html

[26] https://www.standards.city/

[9] R. Marroquin, J. Dubois, and C. Nicolle. WiseNET-smart camera network interacting with a semantic model: PhD forum. In Proceedings of the 10th International Conference on Distributed Smart Camera, pages 224–225. ACM, 2016.

[10] L. Nussbaum, Testbeds in Computer Science, available online at https://github.com/alegrand/RR_webinars/blob/master/9_experimental_testbeds/slides.pdf

[11] T. R. Gruber. A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220, 1993. Avalaible online at http://tomgruber.org/writing/ontolingua-kaj-1993.htm

[12] T. Hey and A. Trefethen. The Data Deluge: An e-Science Perspective, Wiley, Book Editor(s): Francine Berman, Geoffrey Fox, Tony Hey, First published: 11 March 2003 https://doi.org/10.1002/0470867167.ch36

[13] D. Peng and F. Dabek. Large-scale incremental processing using distributed transactions and notifications. In Proc of the 9th USENIX conf. on Operating systems design and implementation, OSDI'10, pages 1–15, Berkeley, USA, 2010.

[14] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems, 26(2):4:1–4:26, 2008.

[15] S. Venkataraman, I. Roy, A. AuYoung, and R. S. Schreiber. Using r for iterative and incremental processing. In Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Ccomputing, HotCloud'12, pages 11–11, Berkeley, CA, USA, 2012.

[16] M.N. Vora. Hadoop-hbase for large-scale data. In Proceedings of the 2011 International Conference on Computer Science and Network Technology, volume 1 of ICCSNT, pages 601–605, Dec 2011.

[17] A. Simonet, G. Fedak, and M. Ripeanu. Active Data: A Programming Model to Manage Data Life Cycle Across Heterogeneous Systems and Infrastructures. Future Generation Computer Systems, page 49, 2015

[18] G. Antoniu, J. Bigot, C. Blanchet, L. Bougé, F. Briant, F. Cappello, A. Costan, F. Desprez, G. Fedak, S. Gault, K. Keahey, B. Nicolae, Christian Pérez, A. Simonet, F. Suter, B. Tang, and R. Terreux. Scalable data management for mapreduce-based data-intensive applications: a view for cloud and hybrid infrastructures. International Journal of Cloud Computing, 2(2):150–170, 2013

[19] A. Simonet, G. Fedak, and M. Ripeanu. Active Data: A Programming Model to Manage Data Life Cycle Across Heterogeneous Systems and Infrastructures. Research Report RR-8062, Inria - Research Centre Grenoble ; ENS Lyon ; University of British Columbia, 2015.

[20] C. A. Petri, and W. Reisig, (2008). "Petri net". Scholarpedia. 3 (4): 6477. doi:10.4249/scholarpedia.6477

[21] David Abramson, Manish Parashar: Translational Research in Computer Science. Computer 52(9): 16-23 (2019)

[22] Ian T. Foster, Carl Kesselman: Translating the Grid: How a Translational Approach Shaped the Development of Grid Computing. CoRR abs/2008.09591 (2020). URL: https://arxiv.org/abs/2008.09591

[23] Taylor Gilliland, Julia White, Barry Gee, Rosan Kreeftmeijer-Vegter, Florence Bietrix, Anton E. Ussi, Marian Hajduch, Petr Kocis, Nobuyoshi Chiba, Ryutaro Hirasawa, Makoto Suematsu, Justin Bryans, Stuart Newman, Matthew D. Hall, and Christopher P. Austin, The Fundamental Characteristics of a Translational Scientist, ACS Pharmacol. Transl. Sci. 2019, 2, 3, 213–216 Publication Date: May 2, 2019, https://doi.org/10.1021/acsptsci.9b00022

[24] L. G. Valiant, A theory of the learnable, Cmmunications of the ACM, November 1984 https://doi.org/10.1145/1968.1972

[25] OWL 2 Web Ontology Language Document Overview (Second Edition)". W3C. 11 December 2012. See https://www.w3.org/TR/owl2-overview/

[26] M.A. Musen, The Protégé project: A look back and a look forward. AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 1(4), June 2015. DOI: 10.1145/2557001.25757003.

[27] T. Özsu, P. Valduriez. Principles of Distributed Database Systems, 4th Edition, Springer, 2020.

[28] P. Valduriez, Ricardo Jimenez-Peris. NewSQL : principles, systems and current trends. IEEE Big Data Conference, Los Angeles, Dec. 2019.

[29] Gay Harrison, Next Generation Databases: NoSQL and Big Data, Apress, isbn: 9781484213292 1484213297, 2015

[30] Industry Foundation Classes (IFC). General Information available at: https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/

[31] A. Gangemi, Ontology:DOLCE+DnS ultralite homepage. http://ontologydesignpatterns.org/wiki/Ontology:DOLCE+-DnS_Ultralite. Latest update on 4 March 2010.

[32] Y. Raimond and S. Abdallah. The event ontology. Technical report, Citeseer, 2007

[33] Gordon D. Baxter, Ian Sommerville, Socio-technical systems: From design methods to systems engineering. Interact. Comput. 23(1): 4-17 (2011)

[34] Rebecca Taft, Irfan Sharif, Andrei Matei, Nathan VanBenschoten, Jordan Lewis, Tobias Grieger, Kai Niemi, Andy Woods, Anne Birzin, Raphael Poss, Paul Bardea, Amruta Ranade, Ben Darnell, Bram Gruneir, Justin Jaffray, Lucy Zhang, Peter Mattis: CockroachDB: The Resilient Geo-Distributed SQL Database. Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020, pages 1493-1509. DOI: https://doi.org/10.1145/3318464.3386134

[35] Garvita Bajaj, Rachit Agarwal, Pushpendra Singh, Nikolaos Georgantas, Valérie Issarny: 4W1H in IoT Semantics. IEEE Access 6: 65488-65506 (2018), https://doi.org/10.1109/ACCESS.2018.2878100