Ontologies for Simulating Smart Cities: the KnOCS Project

Chiara Alì[1], Domenico Cantone[1], Giorgia Leanza[1], Mirko Giuseppe Mangano[1], Alex Mattia[1], Marianna Nicolosi Asmundo[1] and Daniele Francesco Santamaria[1]

[1]*University of Catania, Catania, Italy*

Abstract

Developing smart cities is a multifaceted effort spanning technological challenges that require innovative tools to facilitate their effective and practical deployment. Among such tools, simulators play a pivotal role by enabling the testing, analysis, and optimization of urban systems in controlled, risk-free environments before real-world implementation. Despite their potential, however, the current literature reveals a gap in engineering simulators for modeling and analysing the behaviour of smart cities in a comprehensive and machine-understandable manner. This goal can be reached through ontologies, which offer expressiveness and interoperability, allow for more accurate representations of urban dynamics, ensure alignment with real-world semantics, and provide the flexibility required by urban scenarios.

The present contribution reports on the ongoing project KnOCS – Knowledge Oriented City Simulator, a novel framework that integrates discrete-event simulators with ontologies to support comprehensive modeling and analysis of smart city evolutions.

Keywords

Semantic Web, OWL, Smart City, IoT, Simulation, Discrete Event

1. Introduction

Smart cities reshape urban environments by integrating technologies such as Artificial Intelligence (AI), Internet of Things (IoT), and Big Data to enhance sustainability, efficiency, and quality of life for modern citizens. Smart cities aim at optimizing energy consumption, reducing traffic congestion through smart mobility solutions, improving public services with digital governance, and reducing waste and carbon footprints.

However, building and implementing a smart city is a complex undertaking that encompasses many challenges from infrastructural, economic, social, governance, and technological perspectives.

From a technological standpoint, the ability to digitally simulate complex urban environments before real-world deployment is crucial. Simulators provide a safe and cost-effective way to experiment with different configurations, technologies, and (cyber)security policies without disrupting existing infrastructure. They empower stakeholders to predict system behaviours, detect potential failures, optimize performance, reduce costs, and test new solutions.

However, building such simulators remains challenging. On the one hand, it requires software capable of managing urban scenarios, triggering, collecting, and logging events related to smart cities, including agent actions and interactions. On the other hand, the collected information must be represented in a formal, meaningful, and interoperable way. In this context, ontologies can play a substantial role.

Smart cities are inherently complex systems, shaped by the interaction of thousands of agents and assets of different natures and goals, operating in diverse infrastructures, technologies, and capabilities.

Therefore, ontologies from various domains must be integrated, with the modeled information seamlessly incorporated into the simulator allowing for continuous updates while maintaining coherence and computational efficiency. Moreover, the knowledge base must be designed to adapt and evolve over

Proceedings of the Joint Ontology Workshops (JOWO) - Episode XI: The Sicilian Summer under the Etna, co-located with the 15th International Conference on Formal Ontology in Information Systems (FOIS 2025), September 8-9, 2025, Catania, Italy

[†]These authors contributed equally.

chiara.ali@studium.unict.it (C. Alì); domenico.cantone@unict.it (D. Cantone); giorgia.leanza@studium.unict.it (G. Leanza); mirkogiuseppe.mangano@studium.unict.it (M. G. Mangano); alex.mattia@studium.unict.it (A. Mattia); marianna.nicolosiasmundo@unict.it (M. Nicolosi Asmundo); daniele.santamaria@unict.it (D. F. Santamaria)

time in response to changing urban dynamics, which requires the use of ontologies that evolve over time.

In conclusion, integrating smart city simulators with semantic knowledge bases presents significant challenges, both from an engineering and an ontological perspective.

The **Knowledge-Oriented City Simulator** (KnOCS) is an ongoing project for the definition of a framework for smart city simulation grounded on semantic knowledge bases. KnOCS is designed for modeling, simulation, and analysis of complex urban environments by integrating formal knowledge representations with event-driven computational models.

At the core of KnOCS lies a *Discrete Event Simulator* (DES), a powerful simulation paradigm for capturing the temporal dynamics of systems where changes occur at discrete points in time, particularly well-suited to modeling urban processes such as traffic flow, energy consumption, emergency response, and communication among Internet of Things (IoT) devices.

What sets KnOCS apart from conventional simulation platforms is its integration with semantic knowledge bases, achieved through the use of domain-specific and foundational ontologies which provide a formal machine-interpretable representation of the key entities, agents, and processes involved in the simulation of urban environments. These include: (a) ontologies for agents and their interactions, which define the characteristics, roles, and behaviours of urban agents (e.g., citizens, vehicles, sensors, administrators) and capture their interactions in various contexts such as mobility, communication, and service delivery; (b) evolving ontologies, which represent the way a system changes during a simulation. These ontologies allow the system to track changes, detect anomalies, analyse trends over time, and pave the way to temporal reasoning; (c) ontologies for IoT devices and smart cities stakeholders, which are essential for specifying the participants and assets of urban environments.

By leveraging the Semantic Web, KnOCS supports interoperability, reusability, and explainability of city simulations. This enables researchers, city planners, and policymakers to configure, test, and extend simulation scenarios with minimal effort, using high-level conceptual models rather than low-level code.

Finally, KnOCS is envisioned as a valuable tool for urban digital twin applications, policy impact assessment, smart infrastructure design, and adaptive IoT system testing. Its semantically grounded architecture ensures that the simulations remain consistent, transparent, and aligned with real-world knowledge frameworks.

The remainder of the paper is organized as follows. Section 2 reviews the main contributions from the state of the art. Section 3 outlines the architecture of the KnOCS project and highlights the key design choices, while Section 4 briefly presents a use case. Finally, Section 5 concludes the paper and discusses the next steps necessary to successfully advance the project.

2. Related Works

This section is devoted to the literature review concerning the exploitation of ontologies for simulation purposes, in particular in the context of IoT and smart cities, where semantic technologies are crucial to reach the desired interoperability [1].

IoT research has mainly focused on sensor modulation, with the Semantic Sensor Network (SSN) ontology [2] widely recognized as the standard in the field. A practical application is presented in [3], which describes the open-source platform *OpenIoT* and the associated ontology, while another significant contribution is offered in [4], where the authors introduce *SAREF*, a suite of ontologies that has since evolved into a key resource for ensuring semantic interoperability among smart appliances.

A comprehensive overview of the development of ontologies for IoT is provided in [5], which reviews key advances in the field between 2012 and 2017. Notably, two works deserve mention: *OntoSensor* [6] and its successor *MyOntoSens* [7], the latter of which extends the former by defining the semantic description of sensor observations.

FIESTA-IoT [8] extends this landscape by aiming to unify existing IoT-related ontologies, with a particular focus on test-bed environments such as Smart Santander, where real data from sensors and

smart buildings are semantically annotated. Similarly, the *VITAL* project [9] introduces an ontology designed to manage heterogeneous data streams from smart city devices, modeling sensors and their measurements to support improved service integration within IoT ecosystems.

An attempt to address interoperability through semantic modeling is presented in [10], where the authors propose a unified knowledge base that leverages multiple ontologies to model the dynamic environments in which IoT entities operate. This knowledge base incorporates several existing ontologies, including: (a) the SSN ontology, which is extended to model sensor resources by accounting for additional properties such as roles and events; (b) the *GeoNames* ontology, which is extended to provide a location model that supports the linkage between IoT resources and services; (c) a *Policy Ontology* and a *Service Ontology*, which complete the knowledge base by enabling the specification of rules and functionalities within the IoT context.

In [11], the authors propose a semantic framework based on ontologies to enhance interoperability and automation in IoT systems. The approach is built on a three-layer architecture comprising: (i) the semantic/ontological layer, (ii) the cloud/edge computing layer, and (iii) the IoT devices and communication layer. By transforming raw data into semantically structured formats, enabling efficient data processing at the edge, and supporting heterogeneous communication protocols, the framework is tested through simulations in real-world scenarios. The results demonstrate a 98% communication success rate between devices, a 65% reduction in latency, and 85% efficiency with up to 500 devices.

An overview of the use of ontologies in smart city applications, covering work up to 2021, is provided in [12]. Among more recent contributions, [13] introduces a *Unified Knowledge Model* (UKM) and a framework for semantic reasoning and data management, with a focus on connecting multiple scenarios and leveraging Digital Twins. The UKM is closely linked to the *Snap4City* framework, which in turn builds upon the *Km4City* ontology [14], originally developed to support the reuse of existing ontologies. The overarching goal is to process large volumes of data from both public and private sources, map them into the ontology, and enable the development of services for smart cities.

In the same direction, the *PRISMA* project [15] proposes an ontology that reuses WGS84, NeoGeo, and *Collections* ontologies to integrate heterogeneous data related to urban infrastructure. The ontology models geodata from GIS, as well as information on public transport lines and stops, lighting maintenance systems, road conditions, and historical waste collection data. Similarly, *SCOnt* [16] presents an ontology-based approach, which combines a population ontology, a geo-location ontology, and DBpedia to support a four-layer architecture.

SCOPE, a framework for modeling cybersecurity threats in smart cities using the *UCO* and *CASE* ontologies, is presented in [17], while *TrafCsOnto*, proposed in [18], is a solution aimed at managing traffic in smart cities. Another notable contribution is the *STAR-CITY* project [19], which develops ontologies to diagnose and predict traffic congestion by integrating heterogeneous data sources, including weather conditions, public transport, road events, and social media. Finally, [20] introduces *S2RICO*, a framework whose main objective is to provide a standard ontology for assessing and monitoring smart city performance.

With regard to the management of user consent in the processing of personal data, the authors of [21] propose a unified and consistent model that covers consent, contracts, sensor data, and their processing. The resulting ontology consists of 202 classes, 87 object properties, and 42 data properties, and it reuses nine existing ontologies: GConsent [22], DPV [23], FIBO, PROV-O [24], OntoSensor [6], schema.org [25], DCAT [26], CampaNeo [27], and LCC [28].

At the time of writing, many simulators are available for use in the context of smart cities, such as *SUMO* [29], an open-source traffic simulation designed to handle large road networks, in particular vehicle movement, including traffic lights, public transport, and custom routing algorithms. *SUMO* is widely used in academic and industrial research for evaluating traffic management strategies and intelligent transportation systems. Analogously, *CityFlow* [30] is a high-performance traffic simulator capable of simulating large-scale urban road networks in real time, supporting road topologies and machine learning algorithms for traffic signal controls. In the context of frameworks for simulating transportation systems, it is worth mentioning *MATSim* [31], which is used for multimodal transport systems, long-term planning, and policy evaluation in urban mobility.

Finally, OMNeT++ [32] is a modular, component-based C++ simulation framework, originally designed for building network simulators, but flexible enough to be extended to smart city applications, particularly those involving semantic knowledge bases.

Concerning IoT simulation, it is worth mentioning *IoTIFY* [33], which enables the simulation of thousands of virtual devices sending data over MQTT or HTTP, making it particularly useful for smart city scenarios involving sensors, environmental monitoring, and smart infrastructure.

3. The KnOCS Architecture

Discrete Event Simulation (DES) is a simulation paradigm used to simulate the behaviour and performance of a real-world system as it evolves over time. Unlike continuous simulations, where the system state changes continuously, DES-based frameworks update the system state only at specific points in time; namely, it changes when events happen at distinct time—specific points.

In DES, each event is an instantaneous occurrence that may alter the state of the system. The simulation maintains a clock that progresses from one event to the next, and an event queue that stores all scheduled events, sorted by their execution time. At the core of a DES is an event loop that repeatedly removes the next event from the queue, advances the simulation clock, updates the system state, and potentially schedules new events. Events are scheduled dynamically based on the user's configuration, which specifies a set of dependencies between events as well as constraints on their execution order. In addition to deterministic rules, the configuration may include probabilistic elements that govern the random selection of certain events or event sequences, allowing for varied and non-repetitive behaviour across different runs. This approach ultimately enables the analysis and understanding of complex systems with asynchronous and time-dependent behaviour.

OMNeT++ is the most popular, open-source discrete event simulation framework, primarily used for modeling and simulating communication networks, but flexible enough to support a wide range of systems, including smart cities. The main challenges of simulating smart cities through DES stem from the need for scalability, interoperability, and adaptability to the complexity of urban environments. To address these challenges, the Knowledge Oriented City Simulator (KnOCS) – an ongoing project at the University of Catania– leverages ontological models to: (a) ensure consistency and unambiguity through structured, formal, and shared representations of urban stakeholders; (b) provide a semantic bridge across simulation tasks; (c) enable modularity and reusability within the simulator architecture; (d) enhance simulation with logical consistency, explainability, and traceability.

To integrate ontologies and DES systems, KnOCS provides a framework composed of two main modules. The first, called KnOCS-*Discrete Event* (KnOCS-*DE*), is responsible for managing discrete events related to smart city operations. The second, KnOCS-*Knowledge Base* (KnOCS-*KB*), maintains the semantic knowledge bases and incorporates a *Graph Database Management System* (GDBMS) for efficient storage and retrieval. It is also responsible for storing smart city stakeholder behaviours, the events resulting from their actions and interactions, and any other information used by KnOCS-*DE* to carry out the simulation.

As shown in Figure 1, KnOCS-*DE* consists of three main modules:

- The core module, called *Smart City Emulation Core* (SimCore) and extending OMNeT++, is devoted to the generation and triggering of the discrete events related to smart cities, in particular, agent actions and interactions. It also includes the software required to keep the knowledge base updated and synchronised with the simulator.
- The *Software Development Kit* (SDK) is a collection of software development tools that enables Smart City stakeholders to develop and deploy new modules for KnOCS-*Discrete Event*, thereby extending the simulator's functionalities and capabilities.
- The *User-API* includes the programming facilities that can be adopted to programmatically interact with the simulator.

The KnOCS-*KB* consists of two modules:

- the *GDBMS*, which manages the storing and retrieving of the information from the knowledge base. The current version of KnOCS adopts OpenLink Virtuoso [34] as GDBMS;
- the *Knowledge Base-API* (*KB-API*), the access point of the simulator to the knowledge base, is responsible for integrating the event simulator with the GDBMS and the related knowledge base. The *KB-API* connects the KnOCS-*DE* component with the GDBMS, managing the correct encoding and decoding of smart city events generated by *SimCore* in the knowledge base, their permanent storage, retrieval, and querying. Within the *KB-API*, the *Synchronisation Service* guarantees that changes are correctly applied by persistently recording the precise configuration and state of the city at each simulation step. This mechanism maintains a dynamic, semantically grounded trace of the city's evolution. The *Synchronisation Service* is also responsible for synchronizing the reasoner, ensuring the consistency and alignment of the evolution of the knowledge base throughout the simulation.

OMNeT++ is based on Event-Driven Architectures (EDAs), a foundational paradigm for modeling complex and dynamic systems such as smart cities. In an EDA, changes in the state of the system are captured as discrete events that trigger specific reactions from components of the system. This reactive logic supports scalable and asynchronous management of information flows generated by distributed agents, an essential feature in IoT-based environments.

Our architecture embraces this approach by explicitly modeling events as ontological entities, raised by the actions of the smart city stakeholders. These stakeholders are introduced as agents and their commitments, enriched through the notion of the roles they play in various urban contexts. OMNeT++ is responsible for triggering events thanks to the KnOCS-*KB*, whose *Terminological Box* (T-Box) describes how agents interact and how events are induced by those actions. Ontological representations of simulation events are collected in the *Assertion Box* (A-Box) that captures the evolution of the emulated smart city. The *SimCore* module is responsible for verifying which events can be generated and determining the dependencies among them by querying the T-Box of the KnOCS-*KB*. Using the OMNeT++ module, these events are then generated according to user-defined system presets as the simulation clock advances. When a set of events occurs in a time t_i, the state of the simulation changes, which means that the smart city evolved, and the A-Box is updated accordingly by means of the *KB-API*. When the simulation clock progresses to time t_{i+1}, another set of events occurs, causing the smart city to evolve to its subsequent state.

Continuous update of the A-Box requires appropriate strategies to ensure its consistency and alignment with the progression of the simulation. Although one can devise ontological models to represent the A-Box's evolution, this approach becomes impractical when OWL restrictions are involved or when it is necessary to reconstruct the temporal evolution of the smart city or querying across several states.

To enable this dynamic evolution, KnOCS relies on *evolving ontologies* that lie at the core of the *Synchronisation Service*. Evolving ontologies allow for the formal representation not only of the involved entities, but also of the entire lifecycle of events, including their propagation effects,and the management of state changes over time. Unlike static models, evolving ontologies can be incrementally updated to reflect changes in real-world contexts, while preserving semantic coherence across versions and enabling interoperability among heterogeneous systems. Within this context, the *Synchronisation Service* acts as an orchestrator that updates the knowledge base and ensures the consistency of the evolving knowledge base by leveraging automated reasoners. This synergy between EDA systems and evolving ontologies enables precise, traceable, and semantically grounded simulations of smart city states and their evolution, allowing the system to respond flexibly to dynamic and even unpredictable scenarios. For these reasons, the knowledge base is shaped as described below.

Agents, actions, and events for smart cities. To simulate events in smart cities, it is first necessary to have ontological tools for describing agents, particularly those within the *Internet of Things* (IoT)

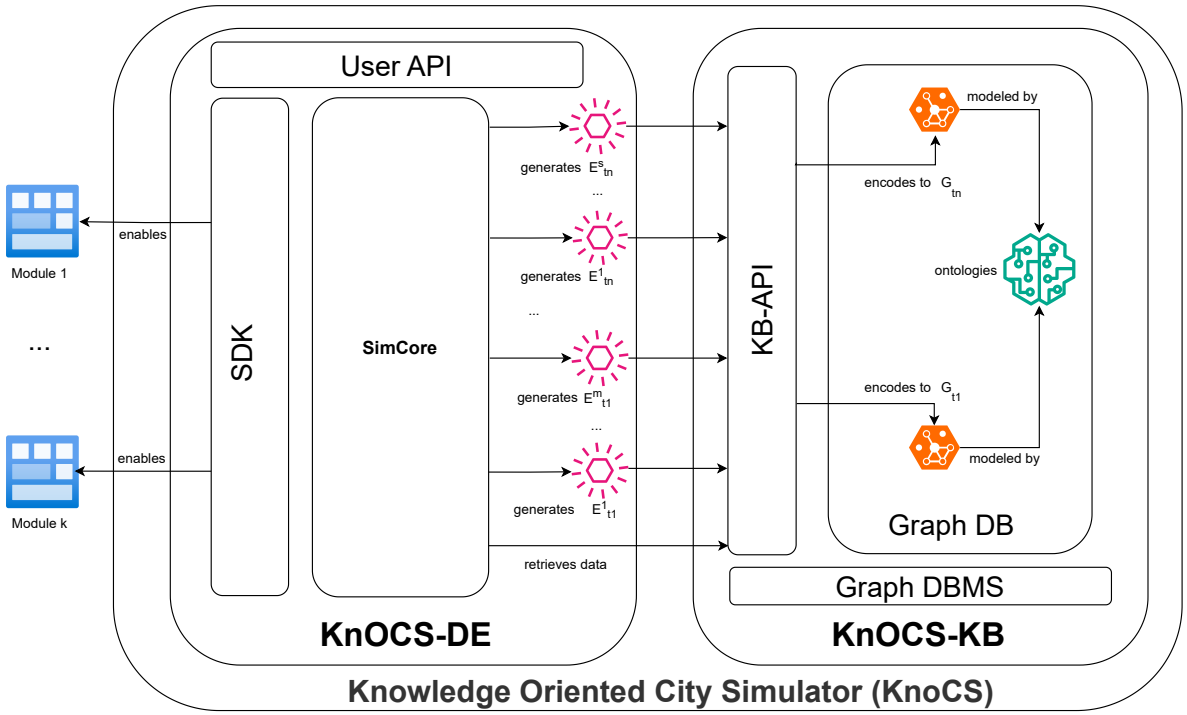

Figure 1: The architecture of KnOCS

domain, and their actions. For this purpose, we adopt OASIS 2 [35, 36], a foundational OWL 2 ontology based on the behaviouristic approach inspired by the *Theory of Agents* and its associated mentalistic notions. OASIS 2 effectively characterizes agents in terms of their capabilities.

Inspired by the *Tropos* methodology [37] which is devised from Agent Oriented Programming (AOP), OASIS 2 represents agents through three essential and publicly shared mental states, namely (expected) *behaviours*, *goals* and *tasks*. Behaviours represent the mental state of the agent associated with its ability to modify its environment or, in general, act or do something. Goals describe mental attitudes representing preferred progressions of a particular system that the agent has chosen to put effort into bringing about [38]. Tasks depict how to carry on such progressions and describe atomic operations that agents perform. Agents and their interactions are represented by carrying out three main steps, namely: (a) an optional step that consists of modelling descriptions of general abstract behaviours, called *templates*, conceptual characterization of behaviours from which concrete agent behaviours are drawn; (b) modelling concrete agent behaviours, possibly drawn by agent templates; (c) modelling actions and associating them with the corresponding behaviours. The first step, not mandatory, consists in defining the agent's behaviour template, namely a higher-level description of the behaviour of abstract agents that can be leveraged to define concrete behaviours of real agents; for example, a template is designed to describe the abstract behaviour consisting in activating a signal. Additionally, templates are useful to guide developers in the definition of the behaviours of their specific agents. The second step consists of representing concrete agent behaviours either by relying on a template or by defining it from scratch. Concrete behaviours are modelled analogously to those of templates, where the models of outstanding features are replaced with actual characteristics. Behaviours drawn by shared templates are associated with them in order to depict the behaviour inheritance relationship.

As stated above, the description of agents comprises three main elements, namely *behaviour*, *goal*, and *task*. Agent tasks, in their turn, describe atomic operations that agents perform, including possibly input and output parameters required to accomplish them. Those elements in OASIS 2 are introduced by way of the following OWL classes:

- *Agent*. This class comprises all the individuals representing agents. Instances of such a class are connected with one or more instances of the class *Behaviour* using the OWL object-property

hasBehaviour.

- *Behaviour*. Behaviours can be seen as collectors comprising all the goals that an agent may achieve. Instances of *Behaviour* are connected with one or more instances of the class *GoalDescription* by means of the object-property *consistsOfGoalDescription*.
- *GoalDescription*. Goals represent containers embedding all the tasks that the agent can achieve. Instances of *GoalDescription* comprised by a behaviour may also satisfy dependency relationships introduced by the object-property *dependsOn*. Goals are connected with the tasks that form them and are represented by instances of the class *TaskDescription* through the object-property *consistsOfTaskDescription*.
- *TaskDescription*. This class describes atomic operations that agents perform. Atomic operations are the simplest actions that agents are able to execute and, hence, they represent what agents can do within their environment. Atomic operations may depend on other atomic operations when the object-property *dependsOn* is specified. Atomic operations whose dependencies are not explicitly expressed are intended to be performed in any order.

In the last step, actions performed by agents are described as direct consequences of some behaviours and are associated with the behaviours of the agent that performed them. To describe such an association, OASIS 2 introduces *plan executions*. Plan executions describe the actions performed by an agent, associating them with one of its behaviours. Associations are carried out by connecting the description of the performed action to the behaviour from which the action has been drawn: actions are hence described by suitable graphs that retrace the model of the agent's behaviour.

OASIS 2 enables agents to declare the activities they can perform, the information required to execute them, and the expected outputs – thus formally specifying their behaviours. Technical details are abstracted away, allowing agents to automatically discover each other without needing to know how the underlying system architecture or the technologies involved. As a result, agent commitments are clearly described, and the evolution of the environment can be unambiguously represented, queried, and accessed. OASIS 2 has already been successfully applied in other domains, including blockchains [39], and is leveraged in KnOCS to answer the *4W1H* of IoT context: What, When, Who, Where, and How [40].

Recently, OASIS 2 has been extended to support the general specifications of processes and procedures executed by agents [35], drawing inspiration from the concept of *Abstract State Machines* [41]. Although the literature provides many modeling approaches to events [42], this extension introduces notions of events and agent roles, particularly suitable for modeling complex scenarios such as those involving smart cities and aligned with OMNeT++, where events correspond to messages sent from one agent to another. In KnOCS, event dependencies modeled through this extended version of OASIS 2 are used to generate cascade events associated with smart city activities. These activities are performed by agents and their related actuators, both of which are described according to the behaviouristic approach adopted by OASIS 2. The modeled actions may include protocols and algorithms that are subject to simulation.

Concerning the ontologies for representing IoT stakeholders, in particular in the context of smart cities, KnOCS requires a vocabulary capable of describing all relevant entities, subjects, objects, and assets. As an initial test-bench, KnOCS adopts an agent-oriented extension of *TrafCsOnto* [18] to simulate traffic within smart cities.

Evolving ontologies. Evolving ontologies are designed to remain up to date as the domains they represent evolve over time. Despite the growing importance of adaptive ontologies, the research community has not yet reached a consensus on how to standardize processes or design patterns for implementing them [43]. A recent proposal by Pietranik and Kozierkiewicz [44] introduces a framework for ontology evolution and alignment maintenance that preserves the validity of ontology alignments by analysing only the changes introduced to the maintained ontologies. Another notable approach [45] focuses on deriving expressive and invertible differential evolution mappings between different versions

of the ontology to support controlled evolution. This mechanism effectively enables ontologies to maintain a dynamic record of the state transitions triggered by events.

The adopted approach to simulating smart cities treats ontologies as active components for tracking the evolving state of the city. This is realized through the *Synchronisation Service* (implemented in Python) that orchestrates interactions both among and within the ontological models in the knowledge base. The service treats the ontologies as mutable data stores, continuously querying the current state and applying updates to reflect changes as the simulation progresses from interval t_i to interval t_{i+1}. These updates specifically target the A-Box, modifying instance-level data by updating class, object, and data property assertions involving individuals that represent city elements. Conversely, the T-Box – that is, the classes and properties characterizing the smart city domain – remain mostly stable throughout state transitions.

From [46], an evolving ontology is defined as a sequence of ontology states arranged along a timeline, namely, a specific version of an ontology at a given point in time, including its explicit axioms and any entailed (inferred) knowledge that can be computed from that version using a reasoner. Let $\mathcal{O}$ denote the set of all possible ontology states. An evolving ontology $\mathcal{E}$ is defined as $\mathcal{E} = \langle o_0, o_1, o_2, \ldots, o_n \rangle$, where o_i denotes the ontology state at time i, with o_0 representing the *initial state*. Each state is obtained by applying a modification operation c_i to the previous state, that is $o_{i+1} = c_i(o_i)$. A modification operation c_i is defined as a composition of fundamental changes, such as adding or removing a class assertion, an object-property assertion, or a data-property assertion.
An evolving ontology is constructed by way of the following steps:

- **Pre-Change State (o_t)**. The process begins by loading the current, stable version of the ontology o at time t into the memory. This condition serves as a crucial reference point for identifying subsequent variations.
- **Transaction Execution**. The activity that triggers the alteration occurs on a temporary version of the ontology. This transaction implements a collection of logical modifications, resulting in a temporary condition in memory.
- **Delta Calculation (Δ)**. Once the transaction is finished, a comparison is carried out between the state after the change (o_{t+1}) and the state before the change (o_t). This process isolates the set of added RDF triples ($\Delta+$) and those that were removed ($\Delta-$). Formally, $\Delta = (\Delta+, \Delta-)$, where $\Delta+ = o_{t+1}\ o_t$ and $\Delta- = o_t\ o_{t+1}$. This fundamental delta accurately captures the precise meaning of the change that took place.
- **Logging and Archiving**. The calculated delta (Δ) is converted into a standard format and stored permanently. At the same time, a metadata entry is created in an evolution log, linking the delta to a timestamp, a semantic description of the operation, and references to the physical delta files. This log serves as a verifiable record of the ontologies.
- **Consolidation of the New State (o_{t+1})**. The new state (o_{t+1}) is only finalized after the delta has been successfully logged, replacing the earlier version (o_t) and establishing the new stable baseline for upcoming transactions.

This method not only guarantees that changes are preserved but also offers complete traceability and the option to carry out rollback processes by sequentially applying reserved deltas or retrieving earlier ontology versions.

4. A use case on V2I communication

We present a simple use case in *Vehicle-to-Infrastructure* (V2I) use case to illustrate how a simulation in KnOCS operates. For simplicity, we consider two agents: a vehicle v and a smart traffic light ℓ. As the vehicle approaches ℓ, the traffic light issues a stop signal to indicate that it has turned red. Due to a malfunction, however, the vehicle ignores the signal and continues its trajectory.

We assume that, in the simulation, each vehicle movement results in an update of its GPS coordinates. Additionally, suitable OASIS behaviours modeling the capabilities of both vehicles and traffic lights, along with the corresponding event models, are encoded in the knowledge base's T-Box. We can summarize these behaviours as follows:

B_1. move(p_i, p_{i+1}): models the capability of a vehicle v to move from a point p_i to a point p_{i+1};

B_2. send_message(s, r, m): models the capability of any agent s to send a message m to an agent r;

B_3. activate_signal(c): models the capability of a traffic light ℓ to set its signal to a colour c.

The *SimCore* module retrieves such information from the T-Box to instruct the OMNeT++ module on how to schedule the events. For such purpose, the module leverages user-defined presets that regulate the event generation. In our example, the vehicle can randomly ignore the stop signal to emulate a communication error or a malfunction of the vehicle brakes. Then, the OMNeT++ scheduler progresses through four intervals t_1–t_4, during which the following events are triggered in order:

E_1. vehicle_move(v, p_0): triggered by the *move* behaviour; at time t_1, the vehicle moves to GPS point p_0, which lies within the area monitored by the traffic light ℓ.

E_2. set_signal(ℓ, red): triggered by the *send_message* behaviour; at time t_2, the traffic light ℓ sets its signal to red. We assume that at time t_0 a set_signal(ℓ, yellow) event has already been triggered to change the state of the light from green to yellow.

E_3. c_send(ℓ, v, m): triggered by the *send_message* behavior; at time t_3, the traffic light ℓ sends the message m to vehicle v, indicating that the light is currently red.

E_4. vehicle_move(v, p_1): triggered by the *move* behavior; at time t_4, the vehicle moves to the GPS point p_1, which is located beyond the traffic light ℓ.

Figure 2 illustrates the ontology states $o_0, \ldots, o_4$ generated by the *synchronization servicee* at times $t_0, \ldots, t_4$, respectively,with the relevant assertions from each state explicitly shown. For conciseness, the OWL assertions are presented using a notation analogous to OMNeT++ constructs. The simulation ends after processing the final state.

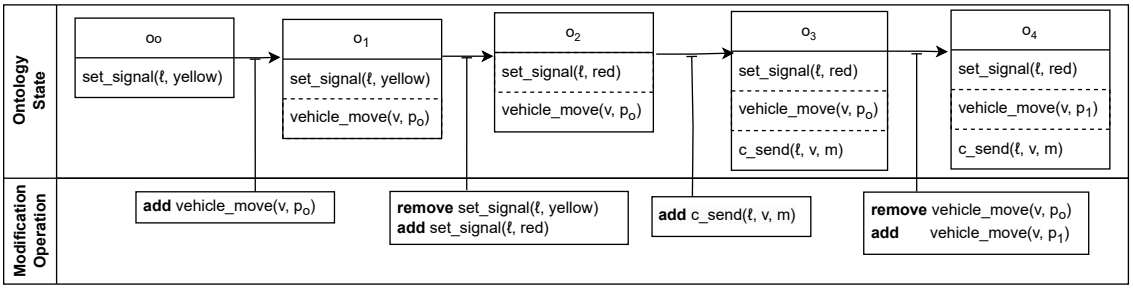

Figure 2: Graphical representation of the ontology states of the case study

5. Conclusions

In this contribution, we presented the *Knowledge Oriented City Simulator*, an ongoing project at the University of Catania aimed at establishing a framework for smart city simulators grounded in semantic knowledge bases. At its current stage, the development team is focusing on the core of KnOCS and on the integration of the modules described in the present contribution.

The next steps include a) the full development of the *KB-API* and of the *SDK*, which will enable semantic querying, reasoning, and ontology-based data manipulation; b) the implementation of a *USER API*, designed to support user-level interactions with the simulation environment; c) the introduction of the first real-world case study, focusing on traffic management. This will serve as the foundation for the first publicly accessible version of the simulator; d) the design of a visual dashboard for real-time visualisation and interaction with simulated city events; e) the integration of interactions among heterogeneous entities such as vehicles, pedestrians, and infrastructure components; and f) the gradual inclusion of additional complex event categories, such as intra-vehicle and extra-vehicle communications, environmental monitoring, and emergency response scenarios. Finally, we plan to replace OMNeT++ with a purpose-built DES specifically tailored to leverage the semantics of the underlying ontological knowledge bases. These steps are part of a broader effort to establish KnOCS as a flexible, extensible, and semantically grounded platform capable of supporting realistic and comprehensive simulations of smart city dynamics.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT in order to Grammar and spelling check, Paraphrase and reword. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] M. Ganzha, M. Paprzycki, W. Pawlowski, P. Szmeja, K. Wasielewska, Semantic technologies for the IoT - An Inter-IoT perspective, in: 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), IEEE, Berlin, Germany, 2016, pp. 271–276. doi:10.1109/IoTDI.2015.22.

[2] A. Haller, K. Janowicz, S. Cox, D. L. Phuoc, K. Taylor, M. Lefrançois, Semantic Sensor Network Ontology, https://www.w3.org/TR/vocab-ssn/, 2017. URL: https://www.w3.org/TR/vocab-ssn/, w3C Recommendation, 19 October 2017.

[3] J. Soldatos, N. Kefalakis, M. Nechifor, M. Serrano, M. Hauswirth, J. Lanza, D. de Miguel, OpenIoT: Open source Internet-of-Things in the cloud, in: I. P. Žarko, K. Pripužić, M. Serrano (Eds.), Interoperability and Open-Source Solutions for the Internet of Things, volume 9001 of *Lecture Notes in Computer Science*, Springer, Cham, 2015, pp. 13–25. doi:10.1007/978-3-319-16546-2_3.

[4] L. Daniele, F. den Hartog, J. Roes, Created in close interaction with the industry: The smart appliances reference (SAREF) ontology, in: R. Cuel, R. Young (Eds.), Formal Ontologies Meet Industry. FOMI 2015, volume 225 of *Lecture Notes in Business Information Processing*, Springer, Cham, 2015, pp. 100–112. doi:10.1007/978-3-319-21545-7_9.

[5] G. Bajaj, R. Agarwal, P. Singh, N. Georgantas, V. Issarny, A Study of Existing Ontologies in the IoT Domain, Technical Report hal-01556256, Inria, 2017.

[6] D. J. Russomanno, C. Kothari, O. Thomas, Sensor ontologies: from shallow to deep models, in: Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory, 2005. SSST '05, IEEE, Tuskegee, AL, USA, 2005, pp. 107–112. doi:10.1109/SSST.2005.1460887.

[7] L. Nachabe, M. Girod-Genet, B. E. Hassan, Unified data model for wireless sensor network, IEEE Sensors Journal 15 (2015) 3657–3667. doi:10.1109/JSEN.2015.2393951.

[8] R. Agarwal, D. G. Fernandez, T. Elsaleh, A. Gyrard, J. Lanza, L. Sanchez, N. Georgantas, V. Issarny, Unified IoT ontology to enable interoperability and federation of testbeds, in: 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), IEEE, 2016, pp. 70–75.

[9] A. Kazmi, Z. Jan, A. Zappa, M. Serrano, Overcoming the heterogeneity in the Internet of Things for Smart Cities, in: International workshop on interoperability and open-source solutions, Springer, 2016, pp. 20–35.

[10] S. N. A. U. Nambi, C. Sarkar, R. V. Prasad, A. Rahim, A unified semantic knowledge base for IoT,

in: 2014 IEEE World Forum on Internet of Things (WF-IoT), IEEE, Seoul, Korea (South), 2014, pp. 575–580. doi:10.1109/WF-IoT.2014.6803232.

[11] R. Ranpara, A semantic and ontology-based framework for enhancing interoperability and automation in iot systems, Discover Internet of Things 5 (2025). doi:10.1007/s43926-025-00122-8.

[12] A. D. Nicola, M. L. Villani, Smart city ontologies and their applications: A systematic literature review, Sustainability 13 (2021) 5578. doi:10.3390/su13105578.

[13] P. Bellini, D. Bologna, P. Nesi, G. Pantaleo, A unified knowledge model for managing smart city/IoT platform entities for multitenant scenarios, Smart Cities 7 (2024) 2339–2365. doi:10.3390/smartcities7050092.

[14] P. Bellini, M. Benigni, R. Billero, P. Nesi, N. Rauch, Km4City ontology building vs data harvesting and cleaning for smart-city services, Journal of Visual Languages & Computing 25 (2014) 827–839. doi:10.1016/j.jvlc.2014.10.023.

[15] S. Consoli, M. Mongiovì, A. G. Nuzzolese, S. Peroni, V. Presutti, D. Reforgiato Recupero, D. Spampinato, A smart city data model based on semantics best practice and principles, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 1395–1400.

[16] M. Beseiso, A. Al-Alwani, A. Altameem, An interoperable data framework to manipulate the smart city data using semantic technologies, International Journal of Advanced Computer Science and Applications 8 (2017) 68–72.

[17] Y. C. Tok, D. Y. Zheng, S. Chattopadhyay, A smart city infrastructure ontology for threats, cybercrime, and digital forensic investigation, Forensic Science International: Digital Investigation 52 (2025) 301883. doi:10.1016/j.fsidi.2024.301883.

[18] N. Saafi, K. Dhouib, An ontological model to enhance traffic conditions in smart city domain, Spectrum of Engineering and Management Sciences 2 (2024) 70–84. doi:10.31181/sems1120246m.

[19] F. Lécué, R. Tucker, S. Tallevi-Diotallevi, R. Nair, Y. Gkoufas, G. Liguori, M. Borioni, A. Rademaker, L. Barbosa, Semantic traffic diagnosis with star-city: Architecture and lessons learned from deployment in Dublin, Bologna, Miami and Rio, in: The Semantic Web – ISWC 2014, volume 8797 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 292–307.

[20] M. Panagiotopoulou, A. Stratigea, M. Kokla, Smart, sustainable, resilient, and inclusive cities: Integrating performance assessment indicators into an ontology-oriented scheme in support of the urban planning practice, Urban Science 9 (2025) 33. doi:10.3390/urbansci9020033.

[21] A. Kurteva, T. R. Chhetri, A. Tauqeer, R. Hilscher, A. Fensel, K. Nagorny, A. Correia, A. Zilverberg, S. Schestakov, T. Funke, et al., The smashHitCore ontology for GDPR-compliant sensor data sharing in smart cities, Sensors 23 (2023) Article 6188.

[22] H. J. Pandit, C. Debruyne, D. O'Sullivan, D. Lewis, GConsent - A consent ontology based on the GDPR, in: P. Hitzler, M. Fernández, K. Janowicz, A. Zaveri, A. J. Gray, V. Lopez, A. Haller, K. Hammar (Eds.), The Semantic Web, volume 11503 of *Lecture Notes in Computer Science*, Springer International Publishing, Cham, 2019, pp. 270–282.

[23] H. J. Pandit, B. Esteves, G. P. Krog, P. Ryan, D. Golpayegani, J. Flake, Data privacy vocabulary (DPV) – Version 2.0, in: The Semantic Web - ISWC 2024 - 23rd International Semantic Web Conference, ISWC 2024, Athens, Greece, October 2024, Proceedings, Part III, volume 15233 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 171–193. doi:10.1007/978-3-031-77847-6_10.

[24] K. Belhajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik, J. Zhao, PROV-O: The PROV Ontology, Technical Report, World Wide Web Consortium (W3C), 2012. URL: http://www.w3.org/TR/prov-o/.

[25] R. Guha, D. Brickley, S. MacBeth, Schema.org: Evolution of structured data on the web; big data makes common schemas even more necessary, Queue 13 (2015) 10–37. doi:10.1145/2857274.2857276.

[26] R. Albertoni, D. Browning, S. Cox, A. N. Gonzalez-Beltran, A. Perego, P. Winstanley, The W3C data catalog vocabulary, Version 2: Rationale, design principles, and uptake, arXiv preprint arXiv:2303.08883 (2023).

[27] S. C. Rasmusen, M. Penz, S. Widauer, P. Nako, A. Kurteva, A. Roa-Valverde, A. Fensel, Raising consent awareness with gamification and knowledge graphs: An automotive use case, International

Journal on Semantic Web and Information Systems 18 (2022) 1–22. doi:10.4018/IJSWIS.300820.
[28] Languages, Countries, and Codes (LCC) Specification Version 1.2, Technical Report, Object Management Group (OMG), 2021. URL: https://www.omg.org/spec/LCC/1.2/About-LCC/.
[29] M. Behrisch, L. Bieker, J. Erdmann, D. Krajzewicz, SUMO – simulation of urban mobility: An overview, in: Proceedings of the Third International Conference on Advances in System Simulation (SIMUL), 2011, pp. 63–68. DOI: 10.5281/zenodo.13907886.
[30] Y. Zhang, et al., Cityflower: An efficient and realistic traffic simulator with embedded machine learning models, 2024. URL: https://arxiv.org/abs/2402.06127. arXiv:2402.06127.
[31] A. Horni, K. Nagel, K. W. Axhausen (Eds.), The Multi-Agent Transport Simulation MATSim, Ubiquity Press, 2016. doi:10.5334/baw.
[32] A. Varga, R. Hornig, An overview of the OMNeT++ simulation environment, in: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops (SIMUTools), ICST, 2008, pp. 1–10. doi:10.4108/ICST.SIMUTOOLS2008.3027.
[33] IoTIFY Team, IoTIFY Documentation, 2024. URL: https://docs.iotify.io/, online documentation.
[34] O. Erling, Virtuoso universal server: A platform for linked data and semantic web applications, 2019. URL: https://virtuoso.openlinksw.com/, accessed: 2025-05-21.
[35] G. Bella, G. Castiglione, D. F. Santamaria, A behaviouristic approach to representing processes and procedures in the oasis 2 ontology, in: Proceedings of the Joint Ontology Workshops 2023, Episode IX: The Quebec Summer of Ontology, co-located with the 13th International Conference on Formal Ontology in Information Systems (FOIS 2023), Sherbrooke, Québec, Canada, July 19–20, 2023, volume 3637, CEUR Workshop Proceedings, 2023.
[36] G. Bella, D. Cantone, C. F. Longo, M. Nicolosi-Asmundo, D. F. Santamaria, The Ontology for Agents, Systems and Integration of Services: OASIS version 2, Intelligenza Artificiale, Vol. 17, no 1 (2023) 51–62.
[37] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos, Tropos: An agent-oriented software development methodology, in: Autonomous Agents Multi Agent Systems, volume 8:3, 2004, pp. 203–236.
[38] M. B. van Riemsdijk, M. Dastani, M. Winikoff, Goals in agent systems: A unifying framework, in: Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS), AAMAS 08, International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 713–720.
[39] G. Bella, D. Cantone, M. Nicolosi Asmundo, D. F. Santamaria, Towards a semantic blockchain: A behaviouristic approach to modelling Ethereum, Applied Ontology 19 (2024) 143 – 180. doi:10.3233/AO-230010.
[40] G. Bajaj, R. Agarwal, P. Singh, N. Georgantas, V. Issarny, 4W1H in IoT Semantics, IEEE Access 6 (2018) 65488–65506. doi:10.1109/ACCESS.2018.2878100.
[41] E. Börger, R. F. Stärk, Abstract State Machines. A Method for High-Level System Design and Analysis, Springer, 2003. URL: http://www.springer.com/computer/swe/book/978-3-540-00702-9.
[42] F. H. Rodrigues, M. Abel, What to consider about events: A survey on the ontology of occurrents, Applied Ontology 14 (2019) 387–422. doi:10.3233/AO-190217.
[43] F. Zablith, G. Antoniou, M. d'Aquin, G. Flouris, H. Kondylakis, E. Motta, D. Plexousakis, M. Sabou, Ontology evolution: A process-centric survey, The Knowledge Engineering Review 30 (2015) 45–75. doi:10.1017/S0269888913000349.
[44] M. Pietranik, A. Kozierkiewicz, Methods of managing the evolution of ontologies and their alignments, Applied Intelligence 53 (2023) 20382–20401. URL: https://doi.org/10.1007/s10489-023-04545-0. doi:10.1007/s10489-023-04545-0.
[45] M. Hartung, A. Groß, E. Rahm, COnto-Diff: Generation of complex evolution mappings for life science ontologies, Journal of Biomedical Informatics 46 (2013) 15–32.
[46] R. Pernisch, D. Dell'Aglio, A. Bernstein, Beware of the hierarchy – an analysis of ontology evolution and the materialisation impact for biomedical ontologies, Journal of Web Semantics 70 (2021) 100658. doi:10.1016/j.websem.2021.100658.