Preprint Cambridge Centre for Computational Chemical Engineering ISSN 1473 – 4273

Semantic Agent Framework for Automated Flood Assessment Using Dynamic Knowledge Graphs

Markus Hofmeister[1,2,3], Jiaru Bai[1], George Brownbridge[3], Sebastian Mosbach[1,2,3], Kok Foong Lee[2], Feroz Farazi[1], Michael Hillman[3], Mehal Agarwal[2], Srishti Ganguly[2], Jethro Akroyd[1,2,3], Markus Kraft[1,2,3,4,5]

released: June 20, 2023

[1] Department of Chemical Engineering and Biotechnology
University of Cambridge
Philippa Fawcett Drive
Cambridge, CB3 0AS
United Kingdom

[2] CARES
Cambridge Centre for Advanced Research and Education in Singapore
1 Create Way
CREATE Tower, #05-05
Singapore, 138602

[3] CMCL Innovations
Sheraton House
Cambridge
CB3 0AX
United Kingdom

[4] School of Chemical and Biomedical Engineering
Nanyang Technological University
62 Nanyang Drive
Singapore, 637459

[5] The Alan Turing Institute
2QR, John Dodson House
96 Euston Road
London, NW1 2DB
United Kingdom

Preprint No. 309

UNIVERSITY OF CAMBRIDGE

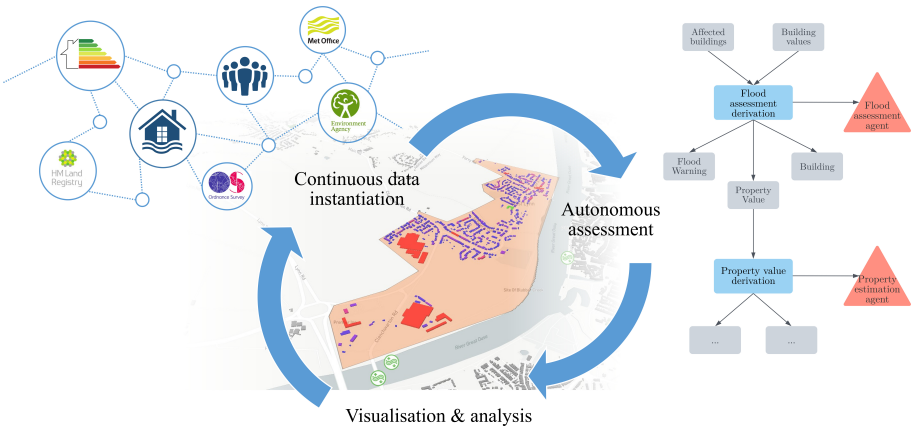
Keywords: knowledge graph, digital twin, derivation framework, ontology, flood assessment

Edited by
Computational Modelling Group
Department of Chemical Engineering and Biotechnology
University of Cambridge
Philippa Fawcett Drive
Cambridge, CB3 0AS
United Kingdom

E-Mail: mk306@cam.ac.uk
World Wide Web: https://como.ceb.cam.ac.uk/

Abstract

This paper proposes a framework of linked software agents that continuously interact with an underlying knowledge graph to automatically assess impacts of potential flooding events. It builds on the idea of connected digital twins based on the World Avatar dynamic knowledge graph to create a semantically rich asset of data, knowledge, and computational capabilities accessible for humans, applications, and artificial intelligence. We develop three new ontologies to describe and link environmental measurements and their respective reporting stations, flood events and their potential impact on population and built infrastructure as well as the built environment of a city itself. These coupled ontologies can dynamically instantiate near real-time data from multiple fragmented sources into the Base World of the World Avatar. Sequences of autonomous agents based on a previously developed derived information framework automatically assess consequences of newly instantiated data, such as newly raised flood warnings, and cascade respective updates through the graph to ensure up-to-date insights into the number of people and building stock value at risk. Although we showcase the strength of this technology in the context of flooding, our findings suggest that this system-of-systems approach is a promising solution to build holistic digital twins for various other contexts and use cases to support truly interoperable smart cities.



Highlights

- Developed three coupled domain ontologies to instantiate data relevant to flooding.
- Implemented semantic agent framework for autonomous data ingestion and assessment.
- Utilised derived information framework to automatically update dependent information.
- Deployed sequences of connected derivation agents to assess impacts of flood hazards.

Contents

1 Introduction

Cities have an incomparable amount of urban data that can boost innovation and provide decision support from strategic planning to day-to-day operations. Despite the abundance of data, it often remains fragmented in isolated silos. Numerous smart city applications have emerged to leverage those ever-increasing amounts of data; however, most of these solutions either focus on individual domains or provide tailored platform solutions to combine datasets for specific use cases using proprietary data models. On the contrary, FAIR data principles [95] promote open findability, accessibility, interoperability, and reusability of available data to maximise the value of individual pieces of information, foster collaboration, and promote more holistic perspectives. Connected and enriched data can be seen as a strategic asset, providing valuable insights, allowing timely and evidence-based decision-making, and enabling myriad automation opportunities. Real interoperability, however, requires not just merging data from different domains, sources, and temporal dimensions, but creating and instantiating the underlying knowledge models, allowing for comprehensive assessments of cross-domain effects.

Ontologies play a crucial role in enabling intelligent systems to comprehend both the conceptual and causal aspects of real-world phenomena [88], thereby facilitating effective knowledge inference from ever-increasing sensor data in the environmental and urban domain. The use of semantics in smart city modelling enables the discovery and analysis of data based on spatial, temporal, and thematic context [94] and enhances both quantity and quality of information available from large scale sensor networks. While numerous ontologies exist to help disambiguate heterogeneous urban information [67, 94], most of the available ontologies focus on conceptualising static domains instead of representing actual data. Depending on their scope, they provide more or less detailed representations of certain smart city aspects, but are hardly linked or utilised to instantiate actual live data feeds; however, exactly this dynamism would be required to create truly interoperable smart city ecosystems remaining current in time.

Floods are among the most devastating natural disasters, causing extensive damage to people, properties, and the environment [67, 78]. Managing flood risk effectively requires an accurate and timely assessment of potential impacts on human lives, infrastructure, and the economy [87]. Hence, identifying and extracting relevant pieces of information as well as making accurate inferences from various data sources rapidly is critical. Although several publicly available flood assessment tools support some consolidated insights, all of them remain limited to individual domains. For example, a live flood map for the UK [66] shows current flood warnings and alerts, together with current readings for river, sea, groundwater, and rainfall levels, and the expected flood risk over the next 5 days; however, no indication of potential impacts in terms of people and built infrastructure at risk is provided. Nevertheless, such cross-domain awareness has been shown to have a positive impact throughout all phases of the disaster management cycle [74], especially as floods are very complex phenomena involving a large number of stakeholders and domain experts to collaborate seamlessly [97]. Ontology-driven systems have been proposed to create a universal understanding across the different stakeholders to enable semantic interoperability, flexibility, and reasoning support [89].

Knowledge graph technology can be used to instantiate ontologies and connect data about

various aspects of urban environments into a network of entities and their relationships. The use of knowledge graphs has gained traction as a vital technology for offering machine-interpretable, semantic information about real-world entities on a large scale. For example, the recently developed geographic knowledge graph WorldKG [34] offers an ontological instantiation of geographic entities in the OpenStreetMap dataset to promote the use of semantic geographic knowledge across various real-world applications, such as event-centric and geospatial question answering as well as geographic information retrieval. Although this work significantly enhances previous efforts, such as LinkedGeoData [90] or YAGO2geo [54], which solely focused on instantiating entities instead of also providing respective class definitions, the proposed ontology seems too light-weight to represent comprehensive connections between related entities. Johnson et al. [52] have presented a scalable workflow for merging multiple geospatial datasets to create a comprehensive knowledge graph of urban infrastructure data. Furthermore, machine learning models are applied to the graph to infer and populate missing data in order to ensure availability of important information for emergency responders during flood events. Buildings and demographics at risk of flooding can be queried; however, no dynamic data assimilation is supported and a new graph needs to be created on demand for each new analysis, lacking continuous insights into real-world situations.

The World Avatar dynamic knowledge graph is designed as extensible semantic system to foster interoperability and effectively address cross-domain questions [3]. It combines ontologies (*i.e.*, data definitions) with actual data instances (*i.e.*, from (open) APIs), and computational services operating on the instantiated data (*i.e.*, so-called agents). Autonomous computational agents accomplish tasks such as updating the knowledge graph, simulating systems, or transmitting responses to the physical world. Based on Semantic Web technologies, the World Avatar intends to overcome limitations of previous platform approaches, such as cumbersome data ingestion pipelines or potential lock-in effects. The effectiveness of this approach has been demonstrated in its ability to create ecosystems of connected digital twins that provide real-time information about the world's state, intelligently control complex systems, or support system design through elaborate what-if analyses. The World Avatar constitutes a versatile system to conduct geospatial [4], temporal [86], as well as cross-domain scenario analyses [38]. The derived information framework [10] can be deployed to track data provenance within the knowledge graph and ensures that newly instantiated data automatically traverses through the graph, including changes to all dependent information.

The **purpose of this paper** is to address the identified lack of dynamism and cross-domain interoperability when assessing potential flooding events. Three coupled ontologies are developed as foundational knowledge models to dynamically assimilate and connect real-world data feeds related to flooding. A sequence of connected autonomous software agents continuously monitors the knowledge graph and re-evaluates the likely impact of imminent floods with regards to people and buildings at risk whenever relevant inputs become updated, creating an up-to-date world view evolving in time. The system is hosted in the cloud using a containerised implementation approach to ensure collaborative deployment together with a unified visualisation interface.

The structure of this paper is as follows: Section 2 summarises previous technical works and provides a review of relevant ontologies as well as used data sources; section 3 intro-

duces the target use case and details both the ontology and agent development to create a dynamic knowledge graph-based digital twin to address it; section 4 describes the deployment of the developed system and highlights its results; and section 5 concludes the work.

2 Background

The Semantic Web [12] is an extension of the World Wide Web with the aim of making web content machine-readable and interoperable by adding structured metadata. It involves the use of ontologies and the Resource Description Framework (RDF) [58] as standard for representing such metadata. The Semantic Web aims to enable a "Web of Data" that can be easily understood and processed by machines, thus facilitating the development of more sophisticated and intelligent web-based and automated applications.

2.1 Ontologies and Knowledge Graphs

An ontology provides a conceptual description of a certain domain of interest. It describes relevant *concepts* (also known as classes), relationships between these concepts (referred to as *properties*), and restrictions and rules describing these relations explicitly. Properties are relationships whose domain and range are defined in terms of concepts, where *object properties* connect an instance of one class (the domain of the property) to another instance of a class (the range of the property) and *data properties* link an instance of a class to an actual data element, a so-called Literal. A so-called Terminological Component (TBox) defines the classes and properties that can exist within the ontology, while the Assertion Component (ABox) contains specific instances of those classes as well as their object and data properties. Ontologies can be represented in various forms, such as formal representations using Description Logic (DL) [8] or the widely accepted Web Ontology Language (OWL) [92], which is currently considered the standard by the World Wide Web Consortium (W3C) [5].

Ontologies can be designed in a modular fashion: upper ontologies (also known as top-level ontologies) capture general knowledge about a certain field by providing basic notions and concepts, and domain ontologies further extend and refine these notions to include detailed knowledge valid for a particular application or domain [89]. Users of a common ontology commit themselves to ask queries and make assertions in a way that is consistent, but not complete, with respect to the knowledge model specified by the ontology (*i.e.*, supporting the open world assumption). Hence, ontologies ease knowledge sharing and reuse without sharing actual (and potentially proprietary) data and enable knowledge to be machine-processable for better information retrieval [89]. Strict formalisation allows for additional automation as well as knowledge discovery, reasoning, or the inference of new and implicit information. Reasoners such as HermiT [29] can be used to check the consistency of an ontology and to deduce indirect subclass relationships. Ontology-Based Data Access (OBDA) [16] using tools like Ontop [98] enables access to data from a variety of structured sources (*e.g.*, relational databases) using ontologies, allowing for more efficient and effective data integration and management.

Representing data using ontologies results in the formation of directed graphs, so-called knowledge graphs (KGs), where nodes define concepts, instances, or data, and edges denote their relationships (*i.e.*, properties). KGs provide an extensible data structure that is well suited to represent arbitrarily structured data and which can be hosted decentralised (*i.e.*, distributed over the internet) using Semantic Web technology in the form of Linked Data [11, 13]. As KG resources can be uniquely identified via Internationalised Resource Identifiers (IRIs), Semantic Web data can unambiguously be linked to one another across the web to allow for identification of related information as well as the creation of a collaborative knowledge graph, where every concept and relation can be referenced back to its original definition. Linked Data fosters FAIR data principles [95], improves machine readability, enhances clarity by identifying and dissolving inconsistencies, provides additional context information, and increases discoverability of information across isolated data sources.

Knowledge graphs can be stored in graph databases, also known as triple stores, such as RDF4J or Blazegraph [14]. Graph databases are designed to host RDF data (and thus KGs) in the form of subject-predicate-object triples and can be queried and updated using SPARQL [6], a query language designed to interact with semantic information.

2.2 The World Avatar

The World Avatar (TWA) project intends to create an all-encompassing model of our world, with a current emphasis on automation and decarbonisation in chemistry [9, 42, 60], process and energy industry [7, 33], and smart cities and city planning [22–24]. TWA aims to provide a technology agnostic and distributed architecture based on open standards and protocols to create a collaborative knowledge-model based system to foster

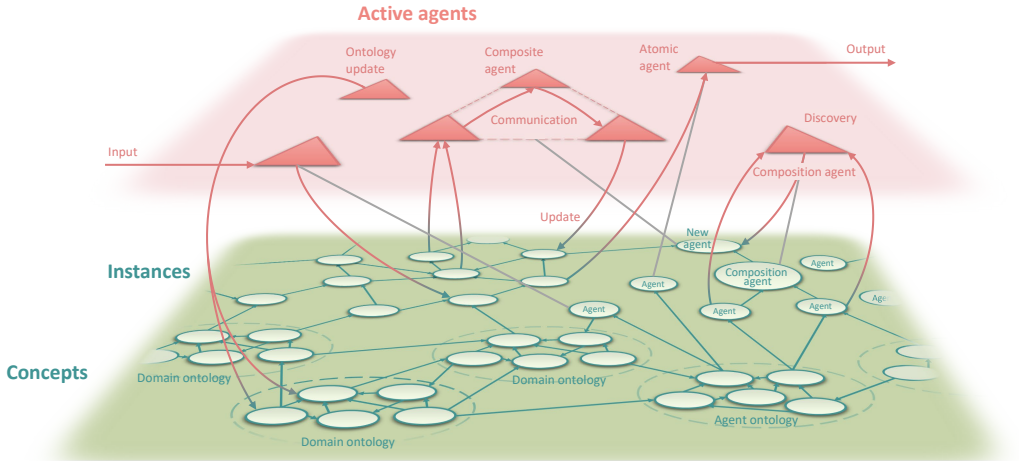


Figure 1: *The design of the World Avatar dynamic knowledge graph. The World Avatar consists of three principal components: 1) ontological description of relevant domains (i.e., concepts), 2) actual data instantiated based on those ontologies (i.e., instances), and 3) automated computational agents to operate on the knowledge graph. Image reproduced from [3] under a CC BY 4.0 licence*

interoperability along three themes [3]: 1) providing cross-domain insights into the current *Base World* (*i.e.*, the actual state of physical assets in the real world), 2) controlling real-world entities, and 3) facilitating complex what-if scenario analyses via so-called *Parallel Worlds*. TWA builds on Semantic Web technologies and is implemented as dynamic knowledge graph which can be distributed across the web, combining the ontological descriptions of relevant concepts and instances with semantically annotated computational agents to operate upon them. The combination of ontological descriptions, instantiated data, and automated agents makes TWA a powerful, extensible, and FAIR-compliant system for representing and reasoning about complex domains of knowledge. The overall design idea is depicted in **Fig.** 1.

As a dynamic KG, computational agents are an integral part of TWA and provide versatility and dynamism to the overall system. Similar to a micro-service architecture, TWA follows a system of systems approach, trying to avoid large monolithic models and breaking them down into sequences of individual agents with tailored capabilities, such as ingesting real-world data, interacting (autonomously) with instantiated information, restructuring the KG, or assembling new composite agents to accomplish more complex tasks [100]. The design of TWA as dynamic KG has been demonstrated as suitable approach to implement a comprehensive ecosystem of connected digital twins to describe the behaviour of complex systems [3].

2.3 Derived Information Framework

The derived information framework provides a knowledge graph native solution to track data provenance and dependencies within a dynamic KG [10] by semantically annotating how certain information can or has been derived from other data instantiated within the KG. Another feature provided by the framework is its ability to automatically detect outdated agent calculations as well as the subsequent re-evaluation of affected outputs. The underlying idea is that all active agents share the same worldview and can be chained together by using the same instance IRIs: by declaring the output instances of one agent operation as input instances of another agent operation, entire cascades of information updates can be executed automatically. This design removes the need for direct agent-to-agent communication and allows for a distributed ecosystem of agents solely connected via their input/output resources within the KG. A short introduction to the framework is provided below, while the interested reader is referred to Bai et al. [10] for more details.

The term *derivation* denotes the fact that a specific piece of information (*i.e.*, a specific instance) has been obtained or computed from other instantiated information. OntoDerivation [10] has been proposed as light-weight ontological markup to denote such dependencies between related instances explicitly. OntoDerivation uses OntoAgent [100] to specify the agent instance responsible for a specific derivation task. While OntoAgent describes computational capabilities of agents conceptually, OntoDerivation concentrates on the instance level, *i.e.*, connects the actual instances processed and generated by a specific agent instance. Upon initialisation, each derivation as well as all its pure inputs are marked with a timestamp according to the W3C standard [26]. Throughout the lifespan of a derivation, these timestamps determine whether a derivation is still up to date or considered outdated. Whenever a derivation output is requested within the KG, the framework compares the

timestamp of the derivation instance with the timestamps of its inputs. In case any of the inputs has been amended after the last derivation calculation (*i.e.*, has a more recent timestamp than the derivation instance), an update of the output instance is conducted before returning the requested information.

The framework differentiates between two types of derivations, namely *synchronous* and *asynchronous* derivations, to cater for different response times upon receipt of a query: The *synchronous* mode utilises the agent's HTTP endpoint for communication, making it faster and suitable for applications that require immediate responses. Conversely, *asynchronous* mode communicates exclusively through the KG (*i.e.*, using specific status markup to track the progress of a derivation computation) to suit slower and computationally more expensive agent operations. Regardless of the communication protocol, all derivations are instantiated consistently: Outputs (*i.e.*, the actually derived information) are connected to respective derivation instance via a `belongsTo` relationship. Inputs (*i.e.*, a set of source information instances) are related to the derivation instances via an `isDerivedFrom` relationship. The connection between a derivation instance and the respective agent is denoted with an `isDerivedUsing` relationship. Although there is no limit to the number of inputs or outputs of a single derivation, any output instance cannot be associated with more than one derivation instance. Chains of derivations can be formed if an instance `belongsTo` to a certain derivation (*i.e.*, as one of its output entities), but also has an `isDerivedFrom` relationship with another derivation instance (*i.e.*, representing one of its input entities). The framework supports derivation dependencies from basic linear chain to non-linear polytree to generic directed acyclic graph.

2.4 Available Public Data Sources

A comprehensive search for available open data sources and existing ontologies concerning built-infrastructure as well as environmental measurement and flood data has been conducted, with primary focus on the UK. Relevant static sources and application programming interfaces (APIs) are summarised below and the interested reader is referred to Hofmeister et al. [50] as well as Appendix A.3 for more details.

Environmental observations data The Met Office DataPoint API provides open access to both weather observations and forecasts (*e.g.*, temperature, wind speed and direction, humidity, *etc.*) for thousands of stations across the UK [65]. While forecasts cover a five-day period, actual observation values are provided for the previous 24 hours. Similarly, the UK Air Information Resource (UK-AIR) provides real-time air quality data for various pollutants via a machine readable Sensor Observation Service [91], including nitrogen dioxide, particulate matter, and ozone. The Environment Agency (EA) provides several API endpoints with (near) real-time information related to flooding and flood risk: The Real Time flood-monitoring API [30] provides a listing of all current flood alerts and warnings with applicable flood areas as well as further meta information (*e.g.*, severity and associated water bodies) and is updated every 15 minutes. Flood alerts and warnings refer to different warning stages: *flood alerts* express that flooding is possible and it is important to stay alert and vigilant, while *flood warnings* refer to situations when flooding is expected and immediate action is required to protect people and properties. The same

API also provides an endpoint for live readings of water levels and flows recorded at various measuring stations along rivers and other water bodies. Furthermore, precipitation and hydrological data about river levels, river flows, groundwater levels, and water quality is provided via EA's Hydrology API [31]. The Flood Forecasting Centre generates a daily flood risk forecast, assessing the likelihood of flooding 5 days into the future [43]. Compared to immediate flood alerts and warnings, this information is associated with higher uncertainty, both with regards to areal extent and anticipated severity.

Building data The Ordnance Survey (OS) provides several open (*e.g.*, OpenMap Local) and premium (*i.e.*, Building Height Attribute) datasets describing the physical characteristics of the built environment in various levels of detail [77]. While the Building Height Attribute (BHA) data represents the most granular data about individual buildings, including their base polygon, building height, and ground elevation, the OpenMap Local contains building data on a more aggregated level and lacks information about building heights. Premium datasets are generally license restricted; however, made available via Digimap [37] for educational and research purposes. The Unique Property Reference Number (UPRN), which constitutes a unique and officially maintained identifier assigned to every addressable location in the UK, can be used to cross-links building information across datasets. The Department for Levelling Up, Housing & Communities offers open Energy Performance Certificate (EPC) data [32] via three dedicated APIs for domestic, non-domestic and display (*i.e.*, mostly public buildings) certificates. This data contains property-level information about energy efficiency, key construction characteristics (*i.e.*, number of rooms, total floor area, building type, *etc.*), high-level usage classification as well as address and location details and is updated every four to six months. His Majesty's Land Registry publishes several public datasets related to residential property sales on a monthly basis: The UK House Price Index (UKHPI) [48] captures the monthly change in the value of residential properties on different levels of geospatial granularity. And the Price Paid Data (PPD) [47] contains information about actual prices and dates of residential property sales, including address and property type.

Population data The OpenPopGrid [68] provides an open gridded population dataset for England and Wales based on the Office for National Statistics (ONS) 2011 Census as well as OS OpenData. It aims to enhance the spatial accuracy of the ONS population dataset by redistributing population to actual residential areas.

2.5 Existing Domain Ontologies

Knowledge model-based systems require structured and curated data to be represented in the form of ontologies. The following section introduces several relevant existing ontologies together with their limitations, with further details provided in Appendix A.4.

2.5.1 Sensor and Measurement Ontologies

Several sensor ontologies have been proposed in the literature, each with its own design principles, coverage, and target applications: The Sensor, Observation, Sample, and Actuator (SOSA) ontology [51] provides a light-weight, modular, and self-contained core ontology for describing basic concepts related to sensors, observations, and actuators. While its simplicity compared to other ontologies fosters re-use, its limited coverage is not suitable for all sensor-related applications. The Semantic Sensor Network (SSN) ontology [96] extends SOSA with more specialised and domain-specific concepts to provide a more expressive ontology to describe sensors and their observations, samples and procedures used, observed properties, and actuators. While its comprehensive coverage of standardisation is one of the key advantages, some aspects of the ontology may be overly complex for certain use cases. The Modular Environmental Monitoring (MEMOn) ontology [63] defines a set of concepts and relationships for describing environmental monitoring equipment, including sensors, together with clear guidelines for mapping sensor data to the ontology. While MEMOn's domain specificity ensures an accurate representation of environmental monitoring data, it may be overly specific for various applications. The Smart Appliances REFerence (SAREF) ontology [39] aims to provide a standardised framework for representing smart appliances and associated data and is designed in a modular fashion. Compared to SSN and SOSA, SAREF has a narrower scope, focusing specifically on smart appliances rather than sensors and observations more broadly; however, it can easily be extended.

Several ontologies have been proposed to describe environmental water resources and associated sensor readings; however, the focus has mainly been on either aligning heterogeneous data from various sensor web resources [35, 62, 93, 94] or supporting water quality monitoring [99]. A hydrological sensor web ontology-based on the SSN ontology has been developed to align semantics and support collaboration between individual water related sensor networks [93], primarily in response to natural disasters such as floods. It introduces hydrological domain classes and establishes relevant reasoning rules.

2.5.2 Flood Ontologies

Various ontologies have been proposed for natural disaster management in general [57, 59] as well as flooding in particular [2, 56, 67, 97]. Recent systematic reviews of existing flood ontologies confirm the increasing usage of ontological approaches for flood knowledge and disaster management [67, 89]. Ontologies have been developed for different aspects and types of flood disasters, including urban flooding, flood risk and environmental assessment, and stakeholder and response management. The majority of the 14 flood ontologies reviewed by Sinha and Dutta [89] are formal application ontologies built around small tasks performed during the response phase of flood disasters [89]. Among these, very few intend to conceptualise the broader domain knowledge of flood disasters, but are mostly scenario and/or task-specific. The number of classes varies significantly between 4 and 410; however, most of the ontologies are modular to simplify the complexity of conceptualising large-scale spatio-temporal systems, such as floods [67, 89]. The study reveals that there is no standard flood ontology available in the field. The most commonly

reused ontologies comprise the Semantic Web for Earth and Environmental Terminology (SWEET) ontologies [18] and the MONITOR risk management ontology [59].

SWEET ontologies define a hierarchy of many flood risk related terms and are often referenced for a variety of environmental concepts [18]. The Environmental Ontology (ENVO) is a domain ontology describing various environment types across several levels of granularity. Its *environmental hazard* module contains suitable concepts to represent a flood and flooding in general, including more detailed descriptions of typical kinds of flooding; however, no concepts to describe social or economic damage of floods are considered. Several domain ontologies have been proposed to resolve ambiguity in information exchange and enable seamless collaboration between various stakeholders during flood disaster management [56, 67]. The focus of these ontologies is mainly on conceptualising the structure and sequence of relevant information, involved organisations, roles as well as the interplay between them rather than representing the actual data itself. Kollarits et al. [59] have developed MONITOR as formalised knowledge model to describe the relations between natural, social and built environments, potentially hazardous events and several risk assessment and management terms. The proposed risk model has been refined by Scheuer et al. [87] proposing an tailored ontology for multi-criteria flood risk assessment.

Although most of the proposed ontologies are claimed to be available in OWL, only ENVO could be found in coded form online. This matches previous findings by Sinha and Dutta [89], which significantly hinders the re-usability of previous efforts and partially undermines the fundamental idea of common ontologies to enable machine-interpretability.

2.5.3 Building Ontologies

OntoCityGML has been proposed as comprehensive ontology [21] for three-dimensional geometrical city objects based on the CityGML 2.0 standard [45]. CityGML is an open data model and XML-based format developed by OGC to describe urban environments by providing a common definition of the basic entities, attributes, and relations within a 3D city model. OntoCityGML provides a multi-scale model with five consecutive Levels of Detail (LoD), where the geometric representation of any building is successively refined from LoD0 to LoD4. A single building can have multiple spatial representations in different LoDs at the same time, *e.g.*, a simple 2D footprint polygon as LoD0 and a set of 3D surfaces confining the building volume as LoD1 representation. OntoCityGML primarily focuses on the geospatial representation of buildings to support spatial analyses and city planning. Further information about a building (*e.g.*, building usage, year of construction, energy rating, *etc.*) can be encoded using predefined codelists or so-called `genericAttributes`. Although this approach provides further information about a building, it does not fulfil the requirements for semantically Linked Data.

A review of 40 metadata schemas for different phases of the building life cycle is provided by Pritoni et al. [79]. Five popular ontologies/schemas have been scrutinised in detail with regards to suitability and potential gaps in building modelling, namely the Brick Schema [17], the BOT ontology [82], the RealEstateCore ontology [83], SAREF including extensions [40] as well as SSN/SOSA [51, 96]. The ontologies differ in perspective of how they represent building information, from focus on topologies of buildings and their sub-components (BOT) to sensors (SSN/SOSA) or devices (SAREF) and

assets (Brick Schema). General-purpose ontologies such as BOT, SSN/SOSA, and the core of SAREF tend to leave gaps that require a modeler to supplement them with extensions or external schemas. Conversely, application ontologies like Brick, RealEstateCore, and SAREF4BLDG may not be as broadly applicable but offer domain-specific features. While the Brick Schema seems most suited for large, complex building automation, the Haystack ontology suits smaller-scale systems . Key ontologies are briefly discussed in Appendix A.4.3, and the interested reader is referred to Pritoni et al. [79] for more details.

Despite the fact that the proposed ontologies differ in how exactly they represent building sub-components and their relationships, most of them are extensible and compatible with one another. Further perspectives on buildings are provides by specific domain ontologies, such as the iCity Building ontology [55] providing a foundational vocabulary to represent building-related data within the urban context or the Domain Analysis-Based Global Energy Ontology (DABGEO) [28] fostering interoperability across the energy domain, with a focus on smart home and smart city energy management [28]. The DABGEO ontology [27] contains a tailored sub-module for knowledge about infrastructure and buildings, containing classes, properties and axioms to represent static building features (*e.g.*, surface, material), geometrical details (rooms, floors) as well as internal and external environmental conditions (*e.g.*, room temperature).

2.6 Representation of Geospatial and Time Series Data

Common graph databases are optimised to handle large-scale RDF data and perform complex queries efficiently; however, native support for geospatial capabilities is limited in most off-the-shelf triple stores [4]. Furthermore, the representation of large quantities of time series data where each data point is instantiated as individual triple with full semantic markup can pose performance issues due to the vast amount of (partially redundant) statements.

Numerous guidelines for encoding geospatial data in RDF have been published, including the GeoSPARQL [72] as well as CityGML [73] standards, both created by the OGC. GeoSPARQL forms the de-facto standard for representing and querying geospatial data on the Semantic Web, including an extension to the SPARQL query language for processing geospatial data. The level of support for GeoSPARQL varies among different triple stores, but remains limited and inconsistent [21, 53]: While RDF4J, for example, provides "partial GeoSPARQL support" [36], Blazegraph (which is used in this study) does not support GeoSPARQL at all and instead offers a custom subsystem for simple geospatial queries [15].

3 Methodology Development

3.1 Use Case

The first key motivation behind this work is to enrich the *Base World* of TWA with multiple additional data sources, including information about the built environment as well

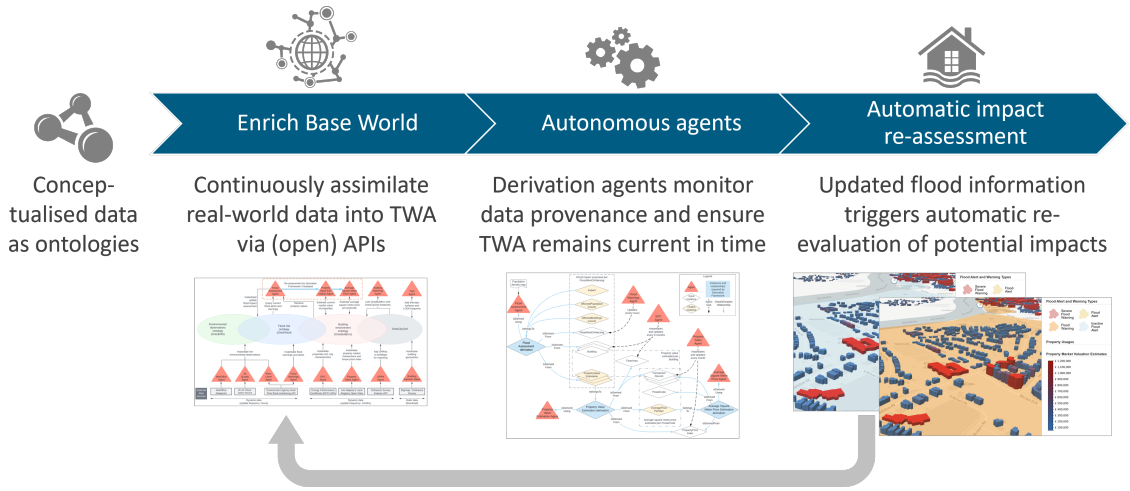


Figure 2: *Schematic of use case implementation. For details see Figs. 9, 15, and 18, respectively.*

as transient and live data feeds from physical entities in the real world. Connecting previously isolated data sources enables both humans and software agents to make better fact-based decisions, especially if the various data sources provide complementing perspectives on related domains, *e.g.*, bringing together data about potential flood events (*i.e.*, severity, areal extent) with data about dwellings (*i.e.*, building locations, building usages, property values). This requires the development of domain ontologies to represent relevant building, flood, and environmental observation data as well as multiple input agents to continuously ingest latest real-world data. Secondly, the agent-enabled autonomous cascading of information through the KG shall be demonstrated with an automatic re-assessment of potential flood impacts using the derived information framework [10]: whenever a new flood alert or warning gets instantiated or already instantiated input information gets updated, the potential impact with regards to the number of people and buildings as well as the estimated building stock value at risk gets re-assessed.

The use case is implemented for a mid-size coastal town in the UK, King's Lynn, and schematically illustrated in **Fig.** 2. While this work focuses on the technical details and implementation, gained insights and the added value for flood risk management are discussed elsewhere [50] in more detail.

3.2 Ontology Development

Typically, ontology development is not a goal in itself, but follows specific usage objectives, such as semantic search or allowing cross-domain interoperability. Competency questions are often used to assess whether an ontology provides a specific enough description of the domain of interest [70]. A good set of competency questions should target TBox, ABox, and the combination of both. A satisfactory response to predefined competency questions is one way to assess the suitability of a proposed ontology.

Ontology development can follow a "top-down" or "bottom-up" approach, or a combination of both. The former targets broad applicability by defining high-level concepts first

before adding any increasingly specific terminologies, while the latter focuses on introducing tailored concepts to represent specific applications or data sources first (*i.e.*, before any optional generalisation). A hybrid approach initially defines salient concepts before either generalising or specialising them as needed. In this paper, a hybrid approach is used guided by available data as well as the target use case, while ensuring a sufficient level of generality to foster re-usability. Ontology development is an iterative process [70], consisting of 1) defining the domain and scope of the ontology, 2) identifying core concepts and relationships for the intended use case, 3) reusing existing concepts and ontologies to the extent possible to foster integration between applications and leverage previous efforts by domain experts, and finally 4) encoding classes, properties, individuals, and restrictions (*i.e.*, value types, cardinality, domain and range restrictions) in OWL format.

The following approaches are used to overcome limitations in representing geospatial and time series data (see section 2.6):
To avoid using custom typed Literals to encode geospatial coordinates suitable for Blazegraph's geospatial engine (as done in previous studies [4, 21]), OBDA via Ontop [98] is used with the actual geospatial information being stored in a PostGIS database. Ontop provides a virtual semantic layer on top of relational databases to allow querying of the underlying structured data via SPARQL. It "hides" the actual data source and exposes the data in terms of the mapped ontological concepts and relationships based on pre-defined mappings between SQL column and ontology concept names. Although Ontop is not fully compliant with OGC GeoSPARQL 1.0, it supports the main geospatial properties and functions. Hence, geospatial information can be served as virtual triples according to provided OBDA mappings and combined with actually instantiated data in Blazegraph.
To overcome issues with instantiating myriad individual time stamps, a light-weight time series ontology [61] has been developed together with a dedicated *TimeSeriesClient*: upon creation, each time series gets instantiated within the KG with full semantic markup, while the actual tabular data (*i.e.*, time stamp and data value) get stored in an associated relational database (*e.g.*, PostgreSQL). The client provides a robust mapping between any data instance in the KG and the respective values stored in the relational database. Adding or deleting time series data only adds or deletes data points in the data base, while deleting an entire time series via the *TimeSeriesClient* also deletes all associated relationships from the KG.

The following subsections delineate the development of three domain-specific ontologies and their interconnections. Although the ontologies strongly reflect the data and structure of the resources detailed in section 2.4, they are kept as general as practicable to ease re-usability. Concepts from existing ontologies are reused where applicable and links to equivalent external concepts are introduced to enrich the meaning of instantiated data. The names of ontological classes and properties are written in typewriter font. All three ontologies rely on the established Ontology of units of measure (OM) [84] to represent measured or reported quantities, numerical values, and associated units. Furthermore, an aligned representation of geospatial information using GeoSPARQL [72] is used. All proposed ontologies have been checked for consistency using the HermiT reasoner in the Protege editor [69] to identify and eliminate logical contradictions. Sets of competency questions have been formulated to evaluate the ontologies' abilities to capture relevant information about their respective domain of interest, with subsets provided in Appendix A.5. A formal representation of the ontologies using description logic [8] is

provided in Appendix A.6, with the OWL versions are publicly available on Github.

3.2.1 Environmental Measurement Station Ontology (OntoEMS)

Several sensor ontologies have been proposed in the literature (see section 2.5.1); however, most of them focus on detailed representations of measurement devices and procedures together with actual measurement values. In contrast, the Environmental Measurement Station ontology aims to establish an aligned conceptualisation for environmental measurement observations from a variety of sources. OntoEMS disregards a comprehensive sensor and procedure representation, as such information is mainly unavailable from public APIs (such as [30, 65, 91]), and focuses on the nature of reported quantities, their geo-location, and time series data for historical readings and forecasts. It represents a light-weight top-level ontology designed to accommodate various data sources without imposing strict limitations on the required data coverage, while still providing an appropriate semantic representation of measured quantities. Given its modular design, OntoEMS can easily be extended with more detailed domain knowledge.

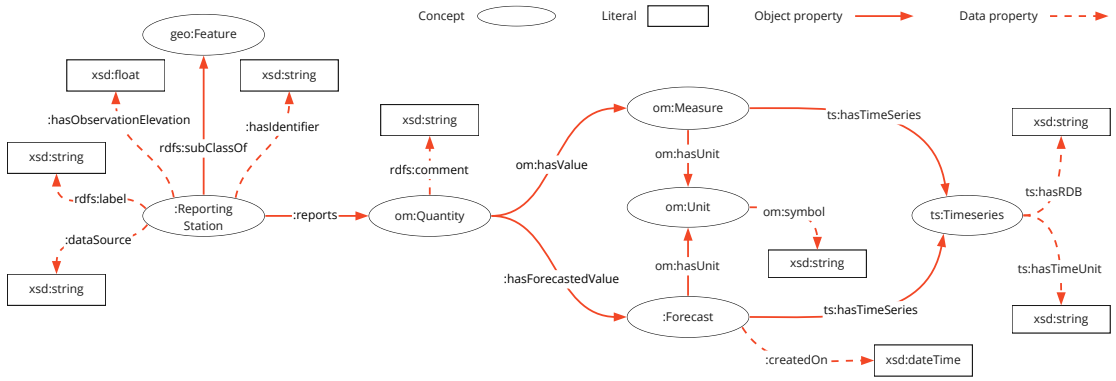


Figure 3: *OntoEMS top-level ontology. OntoEMS represents the top-level ontology to represent environmental reporting stations (e.g., measurement stations) and associated readings of environmental observations (incl. time series values). Further domain knowledge can be incorporated with domain ontologies enriching* `ontoems:ReportingStation`*,* `om:Quantity`*, and* `om:Measure`*. All referenced namespaces are declared in Appendix A.1, with missing explicit namespace declarations referring to* `ontoems`*.*

As depicted in **Fig.** 3, the central concept is a `ReportingStation`, which represents any entity reporting environmental observations and/or forecasts. This general term is used because most of the screened data feeds from public APIs do not contain detailed information about the actual measurement devices. Instead, only high-level metadata regarding the location and nature of the supplied data is given, along with the actual readings data. A `ReportingStation` can refer to a physical asset that houses one or more actual sensors/measurement devices or a virtual station that provides particular readings. To enable geospatial capabilities using GeoSPARQL, each `ReportingStation` is defined as a `rdfs:subClassOf` a `geo:Feature`. All reported quantities are defined as a

`rdfs:subClassOf` a `om:Quantity` and time series data is represented using the Time Series ontology [61]. **Fig.** 4 illustrates this definition for `WaterLevel` and `Rainfall` measurements, which are declared as subclasses of `om:Height`, which itself is a subclass of `om:Quantity`. References to other equivalent external concepts are captured using `owl:equivalentClass` or further `rdfs:subClassOf` relationships, depending on the actual definition of the concept, to enrich meaning and leverage Linked Data.

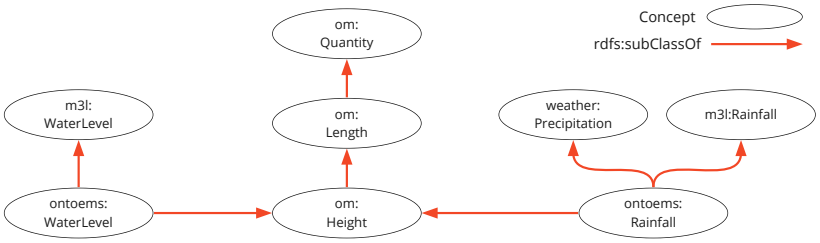


Figure 4: *Example definition of a reported quantity. OntoEMS design requires reported quantities to be subclasses of* `om:Quantity`*; exemplarily depicted for rainfall and river level measurements, incl. references to further ontologies to leverage the power of Linked Data.*

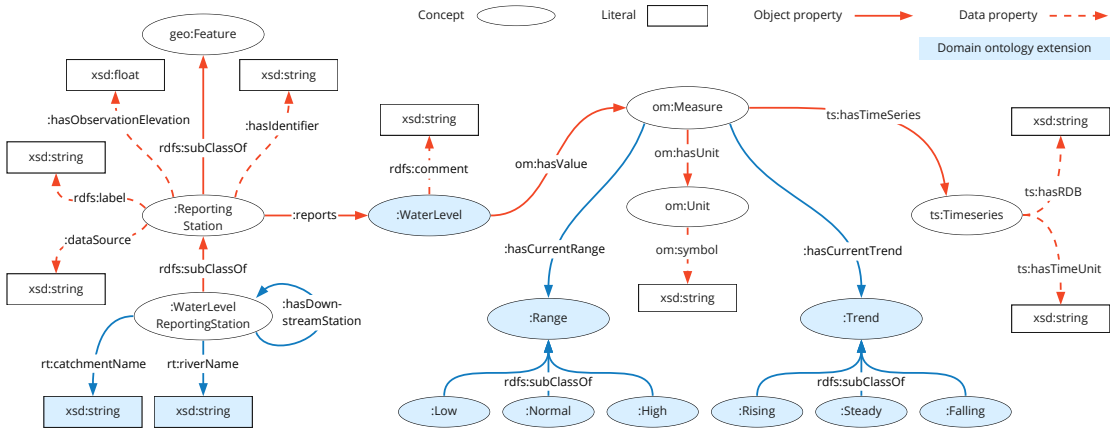


Figure 5: *Example OntoEMS domain ontology extension. The OntoEMS top-level ontology can be refined to suit needs for more detailed domain representations by elaborating* `ontoems:ReportingStation`*,* `om:Quantity`*, and* `om:Measure`*; exemplarily shown for water level reporting stations monitoring the water level in rivers. All referenced namespaces are declared in Appendix A.1, with missing explicit namespace declarations referring to* `ontoems`*.*

The OntoEMS top-level ontology can easily be refined to provide a more detailed description of specific domains, as illustrated in **Fig.** 5 for water level reporting stations. In this case, a `WaterLevelReportingStation` subclass is introduced to include additional information about attached water bodies, such as the catchment or river name, as well as

the relative location to other stations along the same river (*i.e.*, up- and downstream stations). A `Range` and `Trend` concept are included to provide additional context for current water level measurements. Such domain-specific extensions facilitate the direct import of application-specific ontologies, such as the RDF data model utilised by the EA Real Time flood-monitoring API [30]. Furthermore, additional ABox level assertions can be incorporated, *e.g.*, to link instantiated air pollutants with equivalent instances in the European General Multilingual Environmental Thesaurus (GEMET) [41] using `owl:sameAs` relationships.

Several amendments to the Ontology of units of measure have been proposed (*i.e.*, inclusion of new quantities and corresponding units), which are currently being tracked in a fork of the official ontology. It is intended that these amendments will be submitted as a pull request in the future.

3.2.2 Flood Risk Ontology (OntoFlood)

The aim of the proposed Flood Risk ontology is twofold: provide a reasonably general conceptualisation of the flood risk domain while providing a suitable description for the available flood warning and forecast data provided by the EA Real Time flood-monitoring and Flood Forecast APIs [30, 43]. Compared to many previous ontology efforts, the focus is less on establishing a full-fledge conceptual representation of the entire domain and more on providing a semantic description of relevant phenomenological data to instantiate and operationalise actual data feeds. The majority of previously identified ontologies (see section 2.5.2) is not publicly available and direct reuse of core concepts and relations is not possible. Hence, the majority of required concepts and relationships is created independently in OntoFlood and links to previously published ontologies are only included for a few classes where useful to enrich the semantic meaning (*i.e.*, ENVO [19], SWEET [18]). Nevertheless, multiple design choices in OntoFlood are based on previous conceptualisations to build on existing domain knowledge. Several concept are borrowed directly from the data model used by the EA Real Time flood-monitoring API [30] (prefixed with `rt`). Compared to previous flood ontologies, the proposed ontology represents the areal extent of a flood by any arbitrarily shaped polygon instead of linking a flood to a particular predefined region or district. This allows for more versatile geospatial representation, similar to the `flood hazard map` concepts developed by Scheuer et al. [87]. However, the primary purpose is an accurate assessment of affected buildings and people using geospatial queries, as compared to pure visualisation purposes.

OntoFlood is capable of describing floods and associated impacts in three different stages: actual flooding events, flood alerts and warnings (*i.e.*, expected flooding events), and flood forecasts (*i.e.*, potential flooding events, but less certain). The core concept to describe a flood is the `envo:flood` class, which is defined as a `rdfs:subClassOf` of `soph:Event`. Each event is associated with a certain time interval and a `Location` which provides information about the parent `AdministrativeDistrict` as well as the actual `ArealExtentPolygon` of the flood. The `ArealExtentPolygon` is defined as subclass of a `geo:Feature` and the encompassing `AdministrativeDistrict` shall either be the corresponding IRI from ONS Geography Linked Data [71] or refer to it via an `owl:sameAs` relationship. This enables geospatial capabilities, such as the identifi-

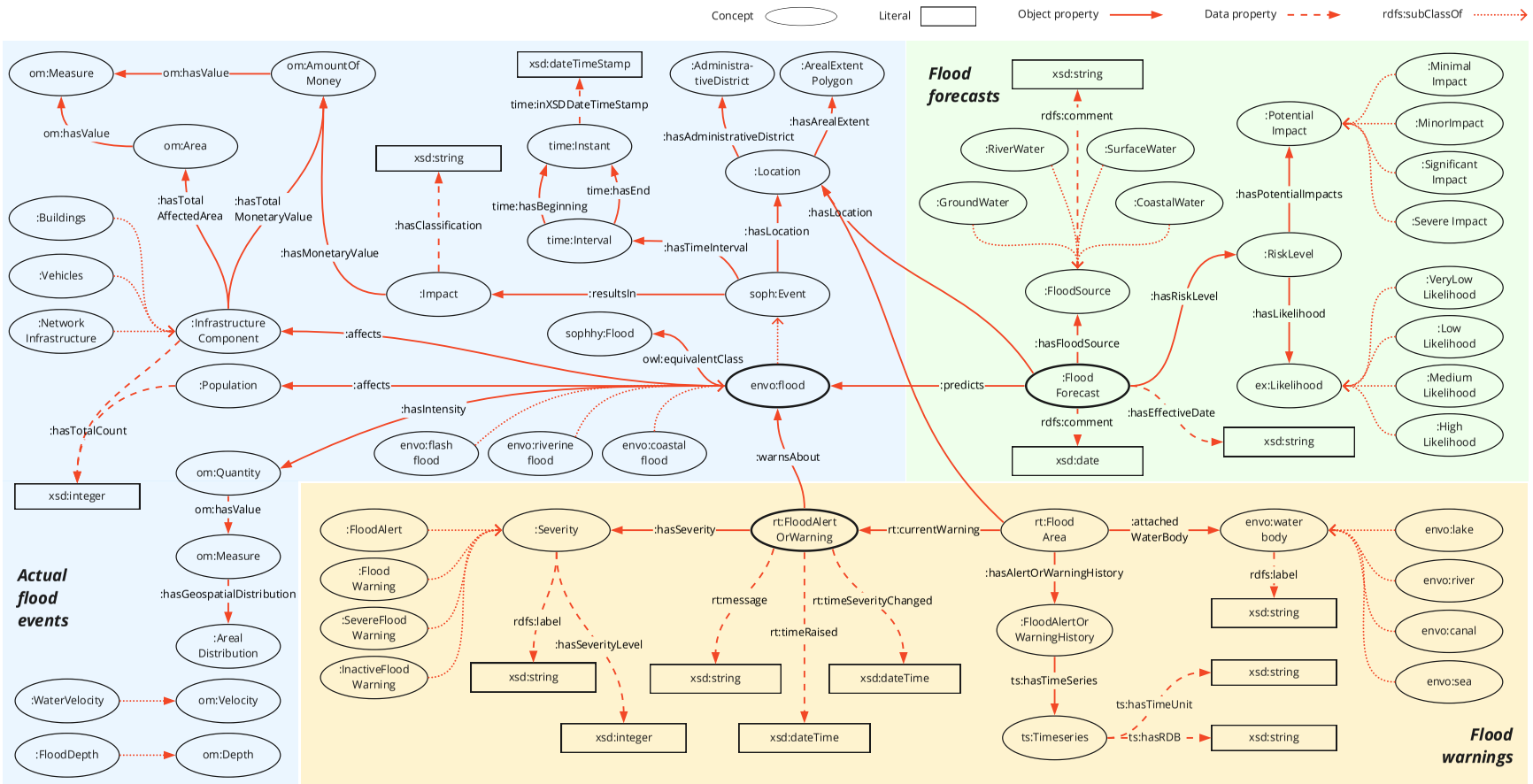


Figure 6: *OntoFlood ontology (extract only). The OntoFlood ontology describes flood events and their (potential) impacts in three stages: 1) actual flood events, 2) flood warnings, and 3) forecasted floods. The ontology has been designed to represent available data from the Environment Agency Real Time flood-monitoring API [30] and assess built infrastructure as well as population at risk. All referenced namespaces are declared in Appendix A.1, with missing explicit namespace declarations referring to* `ontoflood`.

cation of certain geo:Features within a flood polygon or insights into most relevant flood areas in a particular county. An event can resultIn a monetary Impact (positive or negative) describing the total monetary consequences of its occurrence. In the case of flooding, this value describes the total (potential) damage to all affected infrastructure assets. More detailed insights into the (potential) effects of a envo:flood are represented using the affects relationship, connecting to Population and infrastructure assets, such as Buildings, representing both the count and estimated monetary value.

The rt:FloodAlertOrWarning is the key concept for the flood warnings and alert stage, which represents the key data provided by the flood-monitoring API [30]. Each alert or warning warnsAbout a potential envo:flood event and is associated with a certain Severity and rt:FloodArea. Required levels of severity are defined and instantiated as ABox together with the Tbox of the ontology. The concept of a FloodAlertOrWarningHistory is introduced to store key information (*i.e.*, date of issue/change, severity, impacts) of previously issued flood alerts and warnings for a particular rt:FloodArea. This allows to investigate potential trends in both frequency and severity of flood warnings for a certain area and tailor efforts where to take preventive measures. The flood forecast stage is centered around the FloodForecast concept, which predicts a potential envo:flood event. As a flood forecast is less certain and, hence, less quantitative than a rt:FloodAlertOrWarning, it is mainly characterised by its RiskLevel expressing a hazard potential defined by expected likelihood and impact.

3.2.3 Building Environment Ontology (OntoBuiltEnv)

The Building Environment ontology aims to create a light-weight ontology to represent properties (*i.e.*, buildings and flats) suitable for a variety of use cases within TWA. It is designed to integrate with existing agents and knowledge representations, such as OntoCityGML and the City Energy Analyst (CEA) Agent [22]. Compatibility and interoperability with previous building instantiations based on OntoCityGML is ensured by introducing a link between corresponding building instances across both ontologies. While OntoCityGML represents all geospatial and geometric aspects of a building (*i.e.*, in various LoDs), OntoBuiltEnv provides a semantic representation of additional building information (*i.e.*, to replace previously used non-semantic ExternalReference and genericAttribute). The ontology is designed to represent publicly available building data based on the Energy Performance Certificate and HM Land Registry APIs [32, 49] and follows a use case specific approach considering only a required subset of all available data, *i.e.*, focusing on key construction properties as well as property market value relevant information. However, the design ensures extensibility to accommodate additional use cases in the future and already captures building usage as well as solar photovoltaic descriptions to support ongoing efforts to refine the CEA agent.

The ontology reuses existing concepts from multiple screened ontologies (see section 2.5.3), such as DABGEO or iCity Building [27, 55]. A few public concepts from the tailored HM Land Registry data model [47] are included to represent property market relevant data (prefixed with lrppi). A Property represents the central concept of the ontology, with dabgeo:Building and Flat as relevant subclasses. While each Property holds multiple relationships regarding its location, construction characteristics, and mar-

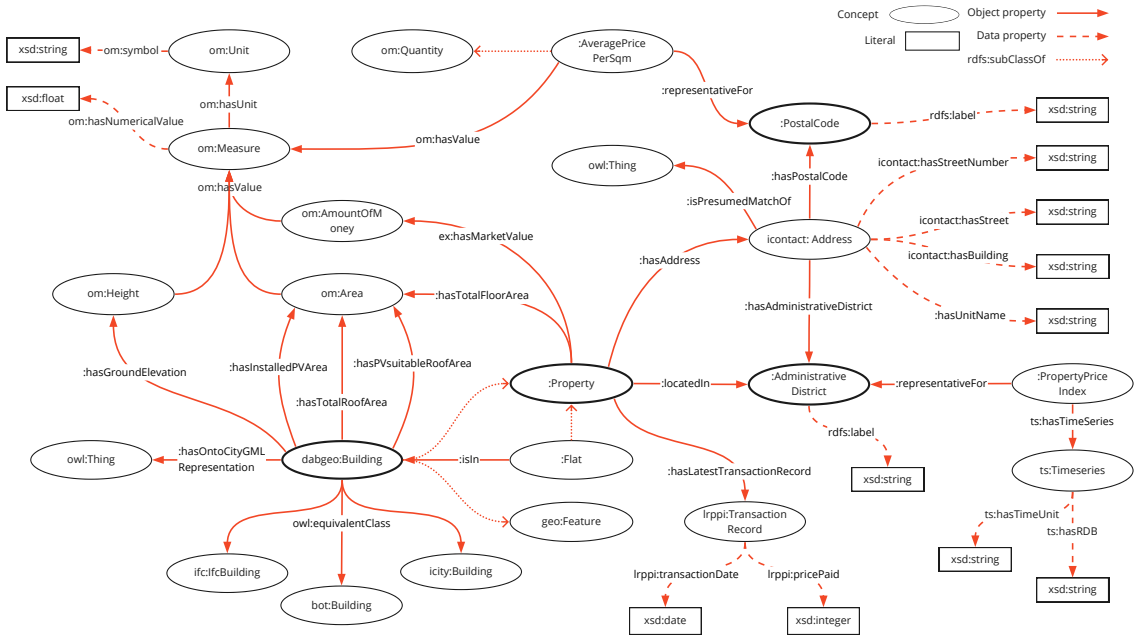


Figure 7: *OntoBuiltEnv ontology part 1 (extract only). The OntoBuiltEnv ontology provides a semantic description for dwellings (i.e., both buildings and flats), incl. location and address details as well as information about previous sales transactions and current market value estimates. The ontology has been designed to represent available data from both Energy Performance Certificates [32] and His Majesty's Land Registry [46], while seamlessly integrating with OntoCityGML [20] for the geospatial representation of buildings. All referenced namespaces are declared in Appendix A.1, with missing explicit namespace declarations referring to* `ontobuiltenv`.

ket valuation details, certain information is restricted to buildings only, *e.g.*, elevation and roof area specifications (see **Fig.** 7). The `dabgeo:Building` class is the key concept to represent buildings and the `hasOntoCityGMLRepresentation` relationship connects it with an external IRI representing the corresponding OntoCityGML instance containing all geospatial information. To enrich semantics, `owl:equivalentClass` relationships are included to refer to building definitions in further ontologies. This ensures linking and easier integration of potentially more elaborate building models in the future (*e.g.*, BIM representation). All buildings are again represented as `geo:Feature` to enable geospatial capabilities using GeoSPARQL (*e.g.*, to assess whether a building is located within a particular flood polygon). Each property is connected with an `icontact:Address` containing detailed address information, including `owl:sameAs` links to corresponding statistical ONS geography IRIs for instantiated `PostalCodes` and `AdministrativeDistricts` to enable further navigation and geospatial analyses .

Fig. 8 illustrates the current representation of `PropertyUsage` types, which has been aligned between available data from the API and requirements from the CEA agent. Furthermore, several subclasses have been introduced to formalise relevant property types (*e.g.*, maisonette or house), built forms (*e.g.*, terraced or detached), and key construction

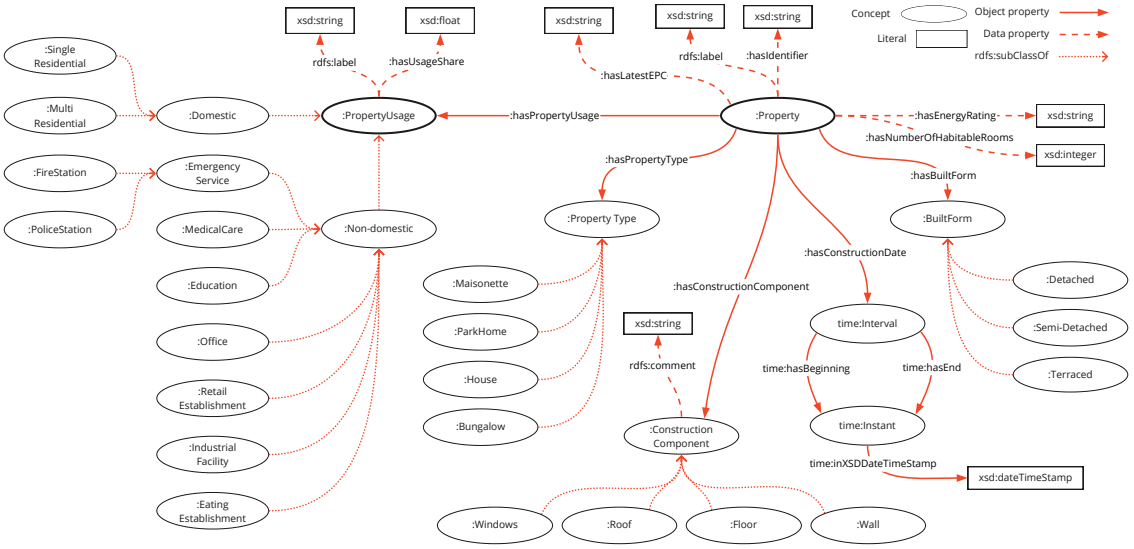


Figure 8: *OntoBuiltEnv ontology part 2 (extract only). The OntoBuiltEnv ontology provides relevant concepts to represent properties, incl. property usage classification, major construction components, property type and built form, and energetic characteristics.*

components. The `hasIdentifier` relationship is used to provide a unique identifier for each instantiated building, required for external referencing or instance matching. In UK context this corresponds to the UPRN of the property, which can be used to unambiguously identify each property and potentially assimilate further data in the future. The `hasLatestEPC` refers to the identifier of the underlying energy performance assessment of the instantiated data and can be used to assess whether this data still reflects latest information available from the API.

3.3 Agent Framework

The developed ontologies are used by a set of input agents to dynamically assimilate data from a variety of sources (see section 2.4 and Appendix A.3) and multiple autonomous agents operating on the instantiated data. All derivation agents (*i.e.*, agents deriving new information based on existing entities using the derived information framework) honour the OntoAgent [100] ontology to refer to associated inputs, outputs, and their respective service description. All agents are implemented in Python or Java and packaged as individual Docker containers. Details about agent deployment are provided in section 4.

The overall framework of interacting agents is schematically depicted in **Fig.** 9, with each agent being explained in detail below. Several dependencies exist between individual agents, since some rely on information maintained by other agents (details in **Tab.** 1).

Met Office Agent The *Met Office Agent* recurringly queries the Met Office DataPoint API [65] for latest weather observation and forecast data and instantiates it according to the OntoEMS ontology. This agent also serves as template for other OntoEMS input

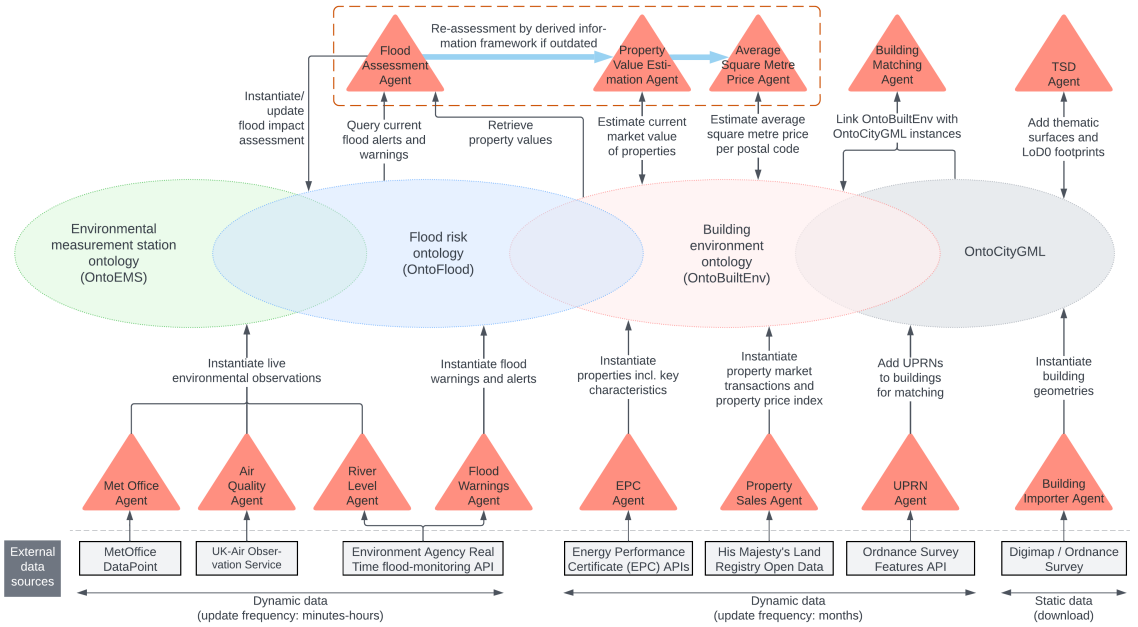


Figure 9: *Agent Framework. Schematic depiction of all agents involved in the flood assessment use case. Input agents (i.e., all agents in the bottom row of the figure) interact with external data sources to instantiate (or update) data within the KG, while other software agents operate (autonomously) on the instantiated data.*

agents to keep TWA's *Base World* in sync with the real world. A detailed Unified Modeling Language (UML) diagram is provided in **Fig.** 10. The agent offers dedicated endpoints to update 1) instantiated reporting stations, 2) reported quantities (*i.e.*, measured and/or forecasted parameters), and 3) the actual time series data of those quantities. Additionally, an endpoint to update all instantiated information is provided, performing tasks 1-3 sequentially: Upon invocation (either via HTTP request or scheduled background task), the agent queries all available reporting stations from both the API and TWA. In case stations are available from the API only, they get instantiated. Subsequently, all available stations and their associated readings are queried from both the API and TWA. Similarly, potentially missing quantities are instantiated and the associated time series instances to store actual readings get initialised. Lastly, the latest time series readings are queried from the API for all instantiated quantities and added to the respective time series instances. Potentially, duplicated timestamps are overwritten to ensure that latest information is available from TWA at all times.

Air Quality Agent The *Air Quality Agent* recurringly queries air pollutant concentration data from the UK-AIR Sensor Observation Service [91] and instantiates it according to the OntoEMS ontology. It follows the same implementation approach as the *Met Office Agent*, with consecutive updates of reporting stations, measured quantities, and actual time series data. Similarly, only not yet instantiated stations and readings are added to the KG on subsequent agent invocations. To leverage the power of Linked Data, each unambiguously identifiable air pollutant reading is linked to its equivalent concepts of the

Table 1: *Agent Dependencies. Overview of all agents and their required predecessors, i.e., agents which need to be executed prior to 1) instantiate required data within TWA or 2) instantiate required derivation markup to reflect information interdependencies.* Input agents *instantiate new data into TWA,* update agents *operate upon instantiated data to enrich/connect individual instances, and* derivation agents *deduce new information based on existing entities using the derived information framework.*

Agent	**Agent type**	**Required predecessor**
Met Office Agent	Input agent	
Air Quality Agent	Input agent	
River Level Agent	Input agent	
Building Importer Agent	Input agent	
TSD Agent	Update agent	Building Importer Agent
UPRN Agent	Input agent	TSD Agent
EPC Agent	Input agent	
Building Matching Agent	Update agent	UPRN Agent EPC Agent
Property Sales Agent	Input agent	EPC Agent
Average Square Metre Price Agent	Derivation agent	Property Sales Agent
Property Value Estimation Agent	Derivation agent	Property Sales Agent
Flood Warnings Agent	Input agent	
Flood Assessment Agent	Derivation agent	Flood Warnings Agent

European air quality e-Reporting initiative [41] via an `owl:sameAs` relationship. This provides additional information about relevant protection targets, (commonly used) units and measurement equipment, besides further metadata. Moreover, this eases alignment and interoperability with other data sources and stations potentially to be incorporated later.

River Level Agent The *River Level Agent* retrieves sensor observation data from the EA flood-monitoring API [30] and instantiates it according to the OntoEMS ontology. It downloads all available data in RDF format once per day and updates the KG by assimilating new station and readings information as well as adding new time series data. The agent instantiates all provided readings, not just river levels, including water flow rates, rainfall, and wind measurements. For `WaterLevelReportingStations` at rivers the agent enriches the RDF data with additional information about stage scales, such as the reference datum and typical low/high readings, and adds relationships to potential downstream stations. Data about the relative location of stations is obtained from another government service [66] through web scraping. The agent also derives and instantiates the current `Range` and `Trend` for each `WaterLevelReportingStation` with associated stage scale instance. The `Range` is determined by comparing the latest water level reading with typical low and high readings, while the `Trend` is based on an assessment of the readings over the last 12 hours. A rising or falling `Trend` is instantiated if the difference between the last and first value in that period exceeds $\pm 10\%$, respectively.

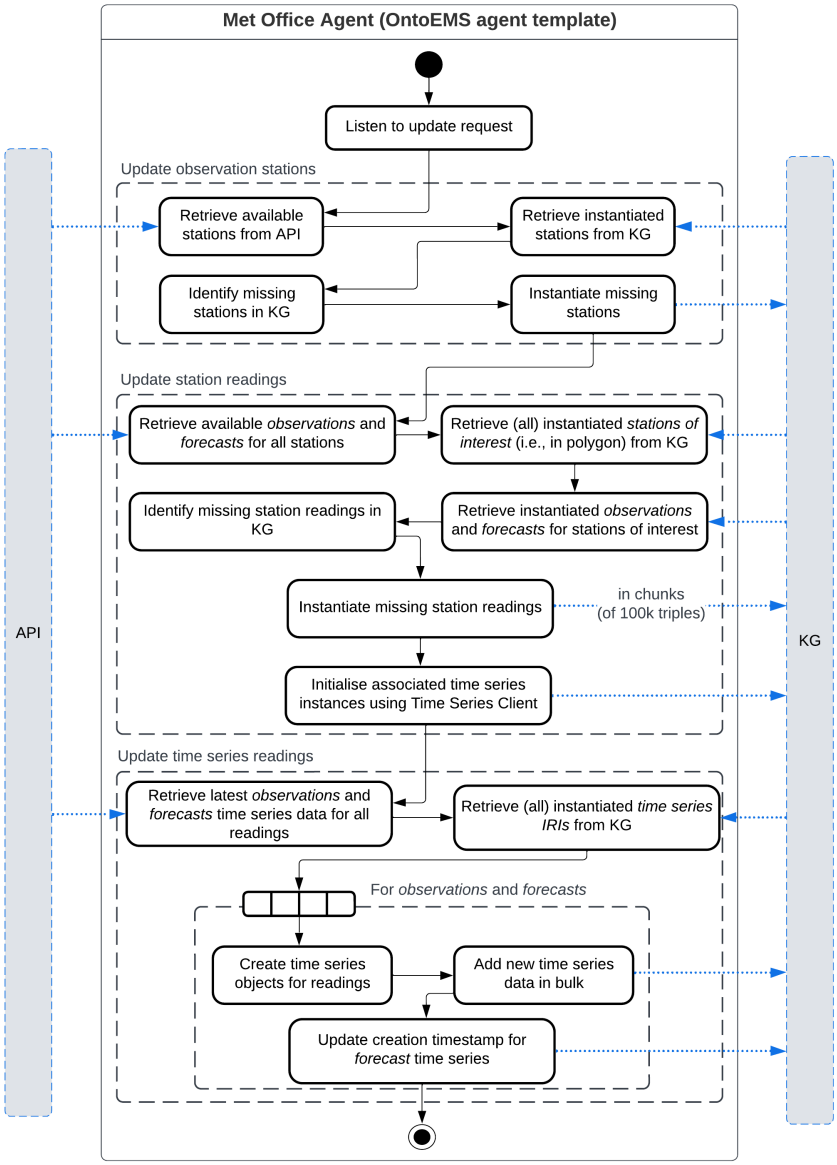


Figure 10: *Met Office Agent. The Met Office Agent recurringly queries the Met Office DataPoint API [65] for both latest weather observation and forecast data and instantiates it according to the OntoEMS ontology. This agent also serves as template for other OntoEMS input agents to keep TWA's Base World up-to-date.*

Building Importer Agent The *Building Importer Agent* imports entire CityGML files and instantiates respective buildings according to the OntoCityGML ontology [21, 22]. The agent supports building representations in multiple LoDs; however, given the available data from OS, this work relies on building instantiations in LoD1, with buildings represented as simple block models describing their overall shape and layout (*i.e.*, building footprints as polygon) Additional building data, such as building height or associated postcode, are also instantiated according to OntoCityGML, either as dedicated data prop-

erty (*e.g.*, building height) or as `genericAttribute` (*e.g.*, postcode).

Thematic Surface Discovery (TSD) Agent The *TSD Agent* enhances instantiated OntoCityGML LoD1 building representations to LoD2 [22]. The agent identifies wall, roof, and ground polygons and adds explicit semantic annotations to complement their pure geometrical perspective in an OntoCityGML compliant form. The agent also offers an endpoint to only identify and restructure the ground surface, *i.e.*, footprint, of a building. As this work focuses on LoD1 building representations, most of the thematic surface restructuring is not necessary; however, the ground surface classification is a pre-requisite for the instantiation of UPRN information by the *UPRN Agent*.

Unique Property Reference Number (UPRN) Agent The *UPRN Agent* enriches instantiated OntoCityGML building instances with a unique identifier, *i.e.*, their UPRN given the UK context of this work. Depending on the provided HTTP request parameters, the agent processes individual buildings (*i.e.*, single OntoCityGML city object with provided IRI) or targets all buildings instantiated in a provided triple store namespace. For each target building, the agent queries the OS Features API [77] with the building's bounding box to retrieve all enclosed UPRNs. Subsequently, all returned UPRNs are tested against the ground surface to exclude UPRNs not intersecting the building itself. Only intersecting UPRNs are then instantiated and linked to the respective building(s).

Energy Performance Certificate (EPC) Agent The *EPC Agent* retrieves data from all three EPC APIs [32] (*i.e.*, domestic, non-domestic, and display certificates) and instantiates relevant building data according to the OntoBuiltEnv ontology. To manage the size of individual API requests, the data is queried per postcode and for all three APIs subsequently. To instantiate all postcodes for a particular local authority, an initial request must be sent to the agent. New postcodes are only instantiated if the local authority is not already present in TWA. All instantiated postcodes get linked to corresponding ONS IRIs using the `owl:sameAs` relationship. A detailed UML diagram is provided in **Fig.** 19 in the Appendix, with key agent logic described below:
The agent requires an available SPARQL endpoint to retrieve instantiated OntoCityGML buildings, since EPC data is only assimilated for properties with available geospatial representation. Initially, all instantiated postcodes are queried from TWA, and the latest EPC data for all instantiated postcodes are retrieved from the respective API endpoint. Returned EPC data with a corresponding UPRN instantiated in OntoCityGML gets instantiated according to the OntoBuiltEnv ontology: either as standalone building or as property belonging to a parent building if the associated OntoCityGML building contains multiple UPRNs. Only outdated EPC data is instantiated, while already instantiated data are skipped. The agent determines information for parent buildings by aggregating data of contained child properties: by summing up actual values (*e.g.*, number of rooms and floor area), using the most common value (*e.g.*, EPC rating code, property type), or concatenating distinct values of child properties (*e.g.*, property usage, description of construction components). After ingesting new information and to ensure that updated information is detectable by the derived information framework, the IRIs of all (potential) pure inputs managed by the *EPC Agent* (*i.e.*, `Property`, `PostalCode`, and `FloorArea` instances)

are retrieved and associated timestamps are added or updated.
To enable visualisation using TWA's unified visualisation interface, the geospatial representation of all buildings (*i.e.*, their 2D footprints and elevations) must be uploaded to PostGIS. The agent provides all necessary functionality to do so; however, corresponding building instances in OntoBuiltEnv and OntoCityGML need to be matched prior using the *Building Matching Agent*.

Building Matching Agent The *Building Matching Agent* links buildings instantiated according to OntoBuiltEnv with their OntoCityGML equivalents. While OntoBuiltEnv provides the overall semantic description of properties, OntoCityGML is solely used for the geospatial representation of buildings. Linking corresponding instances allows the combination of both of these complementary perspectives. Matching is conducted using UPRNs as unique identifiers and realised by instantiating one additional relationship between matched IRIs, namely `hasOntoCityGMLRepresentation`. For details please see **Fig.** 20 in the Appendix.

Property Sales Agent The *Property Sales Agent* is an input agent which queries HM Land Registry Open Data and instantiates it according to the OntoBuiltEnv ontology. More precisely, it queries latest property sales transactions and index data from the Price Paid [47] and the UK House Price Index Linked Data [48], respectively, using Land Registry's publicly available SPARQL endpoint [49]. After instantiating new property sales data, the agent also instantiates the relevant derivation markups to allow for the automatic assessment of the `AveragePricePerSqm` per `PostalCode` as well as the `Property-ValueEstimation` of individual dwellings. A detailed UML diagram is provided in **Fig.** 21 in the Appendix, with key agent logic described below:
Initially, all instantiated properties are retrieved from TWA, including their full address information. Subsequently, for each postcode latest transaction records (*i.e.*, latest transaction date and price paid) are retrieved for all unique addresses from Land Registry's SPARQL endpoint. Since the PPD does not contain UPRN information, sales transactions are assimilated based on address matching. To attach a previous sales record to an instantiated property, both postcode and property type (*i.e.*, building or flat) must match. However, to account for minor discrepancies between instantiated address information (*i.e.*, based on EPC data) and addresses from HM Land Registry, fuzzy matching of the concatenated string of street name, street number, building name and unit name is used. If the fuzzy match exceeds a minimum confidence score (*i.e.*, 95) both addresses are considered equivalent and the retrieved `TransactionRecord` is instantiated for the respective property. The Python library Fuzzywuzzy [25] is used and two matching algorithms have been compared, *i.e.*, the Levenshtein distance and the Damerau-Levenshtein algorithm. Additionally, the performance of multiple scoring methods (*i.e.*, simple ratio, partial ratio, token sort ratio, and token set ratio) has been investigated. The token set ratio calculates the similarity score based on the ratio of the length of the longest common substring to the total length of the two strings being compared after breaking them into individual words and accounting for differences in word order and word frequency. Manual comparisons of the matching results suggests that the Levenshtein distance with the token set ratio performs best. While the scoring method has a significant influence, the difference between

both scoring algorithms is very minor.
Having assimilated all property sales transaction information, the PropertyPriceIndex (*i.e.*, the UKHPI) is instantiated and/or updated for all instantiated administrative districts (*i.e.*, local authorities, as they are the most granular regions for which the UKHPI is published). Finally, the agent also updates the timestamps of amended pure derivation inputs (*i.e.*, TransactionRecord and PropertyPriceIndex instances) and instantiates/updates the required markups for both the AveragePricePerSqm and the PropertyValueEstimation derivation. Both of these derivations are initialised as synchronous derivations to create new info immediately. As the PropertyValueEstimation depends on the AveragePricePerSqm, the latter one is marked up first to ensure availability of the required derivation input. Both markup methods are part of the *Property Sales Agent* and described in more detail in the following two paragraphs.

Average Square Metre Price Derivation Markup This method creates the semantic markup required by the derived information framework to enable the automatic assessment of the AveragePricePerSqm for each instantiated PostalCode: for each instantiated PostalCode an OntoDerivation instance is initialised with all necessary information about the responsible agent and connected with all actual input instances via an *isDerivedFrom* relationship. The method also handles postcodes without previous sales transaction records. In such cases, simply no transaction record instances get connected to the derivation instance as inputs, which ensures that the *Average Square Metre Price Agent* in turn queries data from nearby postcodes from the Land Registry SPARQL endpoint . A detailed UML diagram of the markup method is provided in **Fig.** 23 in the Appendix, with the key logic described below:
Initially, all unique PostalCodes are queried from TWA, followed by the retrieval of all associated TransactionRecords and the representative PropertyPriceIndex. If already instantiated (*i.e.*, due to previous derivation computation), also the AveragePricePerSqm instance for the postcode is retrieved. If no AveragePricePerSqm instance exists yet, a synchronous derivation for new information is initialised to connect all TransactionRecords of that PostalCode as well as the associated PropertyPriceIndex with a newly instantiated derivation instance and get an initial assessment computed immediately by the *Average Square Metre Price Agent*. Additionally, not yet instantiated timestamps are added to all input instances. Otherwise, TransactionRecord instances not yet connected with the existing AveragePricePerSqm instance of the respective postcode (*i.e.*, transaction record has been instantiated after previous average price assessment) are retrieved and added to the existing derivation instance. Finally, a derivation update is requested to ensure the instantiated AveragePricePerSqm is up-to-date.

Property Value Estimation Derivation Markup Similar to the *Average Square Metre Price Derivation Markup* method, this method maintains the required derivation markup for the automatic assessment of the PropertyValueEstimation of instantiated buildings. A detailed UML diagram is provided in **Fig.** 25 in the Appendix, with the key logic described below:
Initially, all instantiated properties are retrieved, together with their associated sales TransactionRecord and FloorArea as well as representative PropertyPriceIndex and AveragePricePerSqm (if available). If already instantiated (*i.e.*, due to previous derivation computation), also the current PropertyValueEstimation instance is queried. If

no market value estimate exists yet for a building, a synchronous derivation for new information is initialised based on all available inputs for that building and requested for immediate initial assessment by the *Property Value Estimation Agent*. As the `AveragePricePerSqm` derivation is marked up first, its value should already be instantiated when conducting the property value estimation markup. In the unlikely event that this is not the case, the derivation can also be initialised with the `AveragePricePerSqm` derivation instance as input. In case a property already has an instantiated `PropertyValueEstimation`, the need for a potential update is evaluated: In case the instantiated property value `isDerivedFrom` the representative `AveragePricePerSqm` and the `FloorArea`, but a sales `TransactionRecord` is actually instantiated for the property (*i.e.*, a new sales transaction has been made available in HM Land Registry after the last time the property value has been estimated), the respective `TransactionRecord` is added as additional input to the existing derivation and an update is requested to re-assess the current market value. Furthermore, not yet instantiated timestamps are added to all input instances.

Average Square Metre Price Agent The *Average Square Metre Price Agent* is a derivation agent to estimate the average square metre price of properties on postcode level. The required input data comprises total `FloorArea` and latest sales `TransactionRecord` for properties as well as the representative `PropertyPriceIndex`. A detailed UML diagram is provided in **Fig.** 22 in the Appendix, with key agent logic described below:
Upon invocation, the agent verifies whether received input instances are suitable to assess the `AveragePricePerSqm`, *i.e.*, both a `PostalCode` and `PropertyPriceIndex` are available (*i.e.*, have been marked up properly). Subsequently, the number of `TransactionRecords` associated with the derivation instance (*i.e.*, available for current postcode) is derived. The agent requires a minimum threshold of transaction instances to compute a meaningful average (*e.g.*, minimum of 5 transactions). If less (or even no) transactions are available, data from nearby postcodes are included: the agent retrieves all postcodes within the same Super Output Area from ONS public SPARQL endpoint [71], queries the instantiated number of transactions for each of them from TWA and orders the postcodes by increasing euclidean distance from current one. Lastly, the nearest postcodes required to fulfill the minimum number of transactions are extracted and all associated `TransactionRecords` are used in the assessment.
Having retrieved a sufficient set of `TransactionRecords`, each sales price is normalised with the total `FloorArea` of the property and scaled using the `PropertyPriceIndex` to derive today's equivalent value for each historical transaction. Finally, the arithmetic mean is computed and instantiated as current `AveragePricePerSqm` for the respective `PostalCode`.

Property Value Estimation Agent The *Property Value Estimation Agent* is a derivation agent to calculate the current market value estimate of properties instantiated in TWA. The estimated market value of any property can be determined by 1) re-calibrate the latest available historical transaction price to today's market conditions or 2) multiplying the total floor area of the property with a representative average square metre price. A detailed UML diagram is provided in **Fig.** 24 in the Appendix, with key agent logic described below:

Initially the agent verifies that sufficient inputs are provided (*i.e.*, have been marked up properly), meaning that either a historical `TransactionRecord` and `PropertyPriceIndex` or total `FloorArea` and `AveragePricePerSqm` instance are available. The first combination is prioritised as actual previous sales transactions for a certain property are considered better proxies for the current market value compared to the more general approach using an average square metre price. Hence, the latter set of inputs shall only be used as fall-back for properties without any previous sales information. After computing the `PropertyValueEstimation` based on available inputs, it is instantiated in TWA as derivation output according to OntoBuiltEnv.
Although significantly more elaborate methods exist for assessing property prices, considering numerous influential factors, such as micro-location, building characteristics, usage classification, *etc.*, this simplified evaluation approach suffices for an initial proof-of-concept.

Flood Warnings Agent The *Flood Warnings Agent* is an input agent which queries flood alert and warning data from the EA Real Time flood-monitoring API [30] and instantiates it according to the OntoFlood ontology. For readability, both alerts and warnings are referred to as flood warnings in the following. A detailed UML diagram is provided in **Fig.** 11, with the key agent logic described below:
Initially, all instantiated `FloodAlertOrWarnings` are queried from TWA and matched against available, and hence currently active, ones from the API: 1) Whenever new flood warnings are issued by the API, the agent initially verifies whether the affected `FloodArea` has already been instantiated (*i.e.*, as flood areas are frequently reused by EA). If the `FloodArea` is already instantiated, the corresponding instance is retrieved. Otherwise, a new instance is created, including all meta information about geospatial extent, attached water body, *etc.* Subsequently, a new `FloodAlertOrWarning` is instantiated, containing all relevant details about severity, warning message as well as the relationship to the applicable `FloodArea`. 2) Existing but outdated `FloodAlertOrWarnings` get updated with latest API information on severity and message details; however, some information, such as the relations to associated `FloodArea` and `Flood` event IRI, are kept unaltered. 3) Any previously instantiated warnings that are no longer available from the API are archived. Archiving shall create a log of the flood warnings history for a particular `FloodArea` to allow for elaborate flood risk analyses; however, this feature is not yet fully implemented and obsolete `FloodAlertOrWarning` instances simply get deleted from TWA, together with all relationships, derivation markup, associated timestamps, and derivation outputs. However, derivation inputs and associated flood areas are not affected, as they may be associated with or re-used by other flood warnings.
After assimilating latest flood warnings, the agent also instantiates the relevant derivation markup and adds or updates associated input timestamps to allow for automatic impact assessment by the *Flood Assessment Agent* (details in paragraph below).

Flood Assessment Derivation Markup This method is part of the *Flood Warnings Agent* and handles the required derivation markup for the automatic assessment of potential impacts associated with individual flood warnings. It creates and maintains an OntoDerivation instance for each `FloodAlertOrWarning`, adds all necessary information about the responsible derivation agent, and connects it with all actual input instances via an

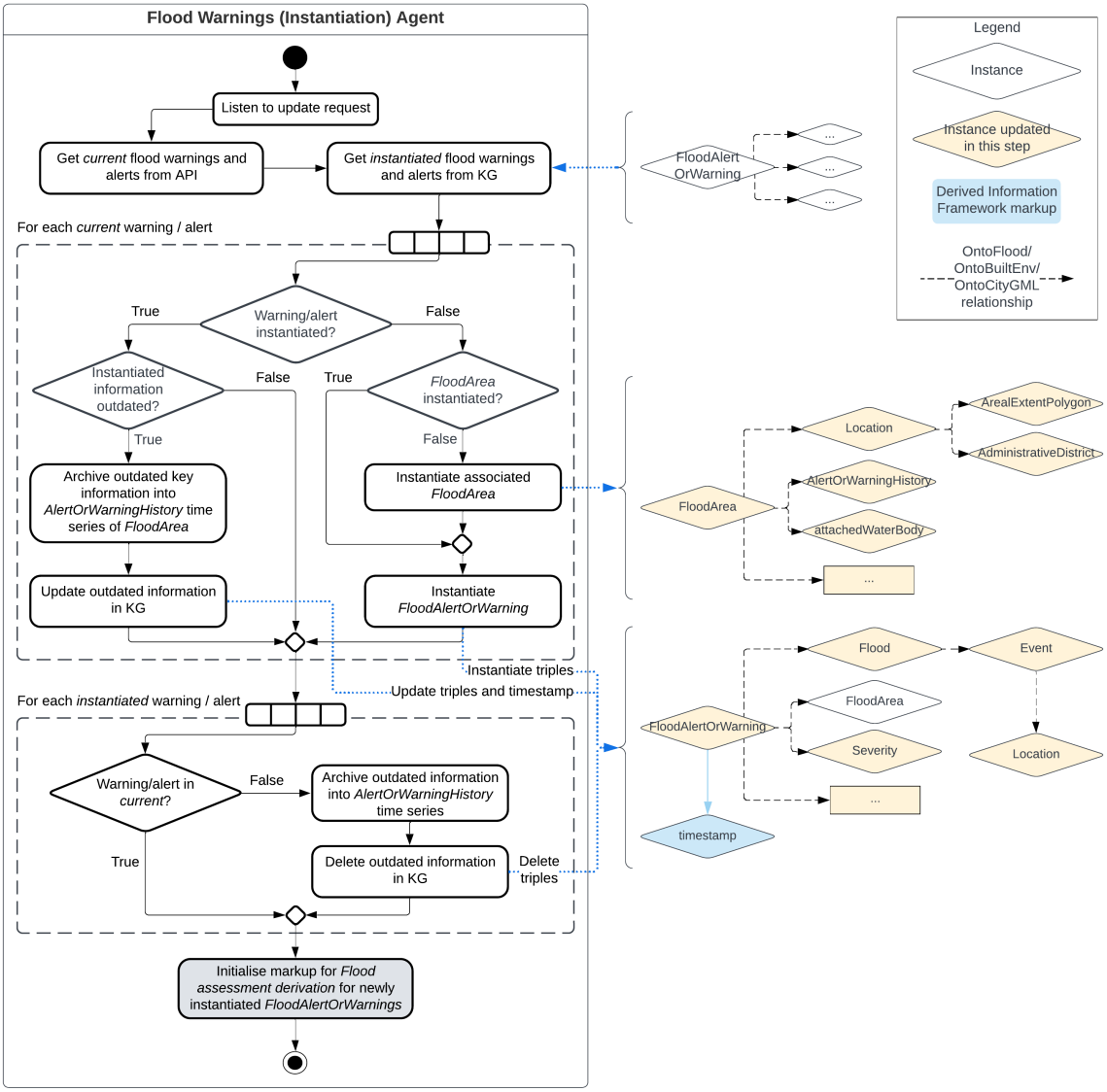


Figure 11: *Flood Warnings Agent. The Flood Warnings Agent recurringly (i.e., every hour) queries the EA Real Time flood-monitoring API [30] and instantiates current flood alerts and warnings. Newly raised alerts/warnings are instantiated (incl. the instantiation of associated flood area(s)), while already existing ones are updated only. Ceased alerts/warnings are deleted from the KG, while associated areas are kept for future reference. Each instantiated alert or warning receives a derived information markup to connect its derivation instance with all relevant inputs for a flood impact assessment. This markup is either newly instantiated or updated (details in Fig. 12)*

`isDerivedFrom` relationship. A detailed UML diagram of the markup method is provided in **Fig.** 12, with the key logic described below:
For newly instantiated `FloodAlertOrWarnings`, the agent retrieves a list of all `Buildings` within the flood area by using geospatial querying to assess which building footprints fall within the `ArealExtentPolygon` of the flood. Subsequently, for each of these buildings,

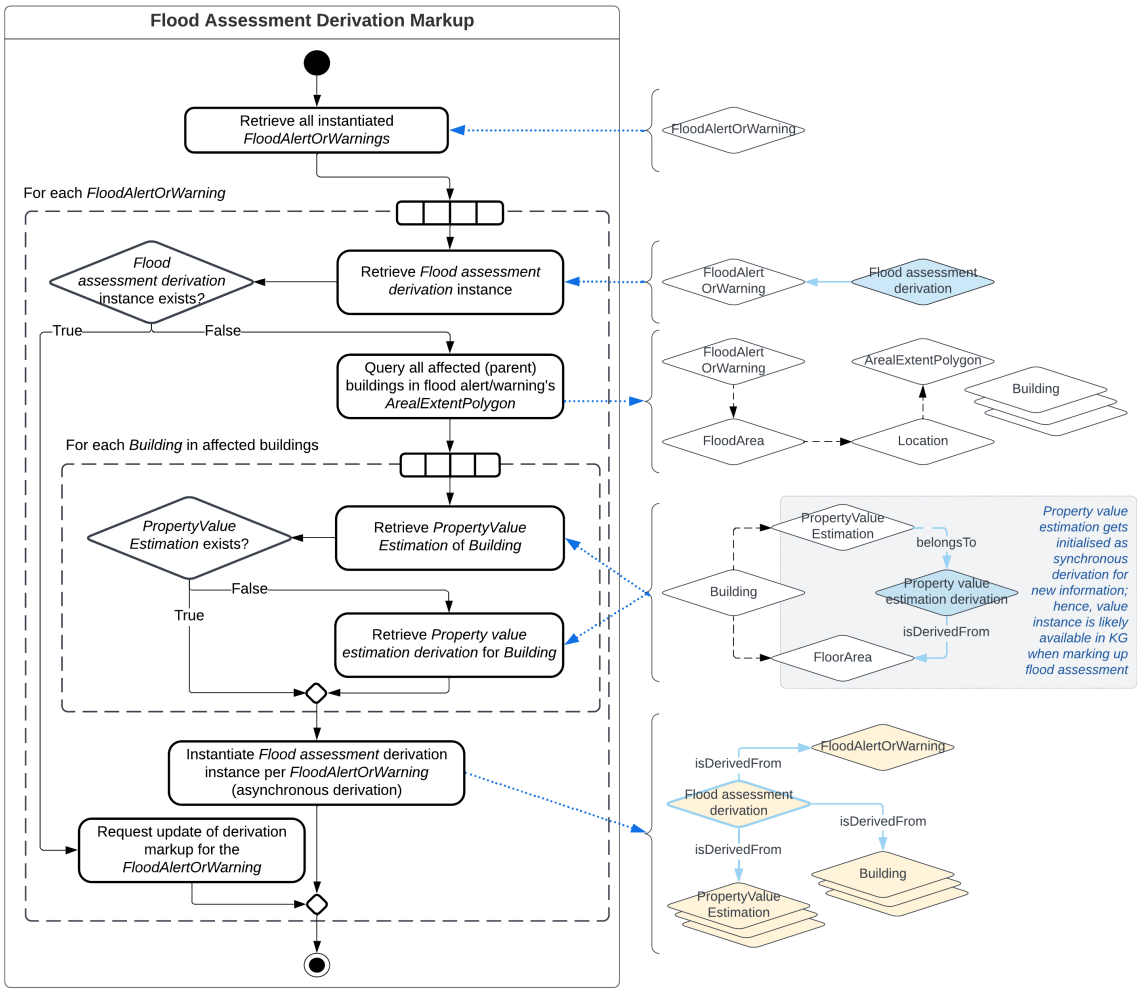


Figure 12: *Flood Assessment Derivation Markup. The Flood Assessment Derivation Markup is a method of the Flood Warnings Agent to connect instantiated flood alert/warning information to corresponding flood assessment derivations. Furthermore, all potentially affected buildings (i.e., buildings located within the flood area polygon) are determined and attached to the derivation instances. Those relationships are required to specify the input instances for each flood assessment and allow the Flood Assessment Agent (see Fig. 13) to automatically detect any outdated information and trigger a re-evalution of potential impacts when they are accessed.*

the agent retrieves the instantiated `PropertyValueEstimation` instance if available. If not, it retrieves the corresponding derivation instance. Finally, an asynchronous derivation is instantiated to connect the flood assessment derivation instance with all potentially affected `Building` IRIs, their `PropertyValueEstimation` IRIs, and the respective `FloodAlertOrWarning` instance. Any potentially not yet instantiated timestamps for pure inputs are added, and an initial assessment is requested, which will be executed the next time the *Flood Assessment Agent* monitors the triple store.
For already instantiated but updated `FloodAlertOrWarnings`, the timestamp of the

flood warning instance is updated and a derivation update for the existing derivation IRI is requested.

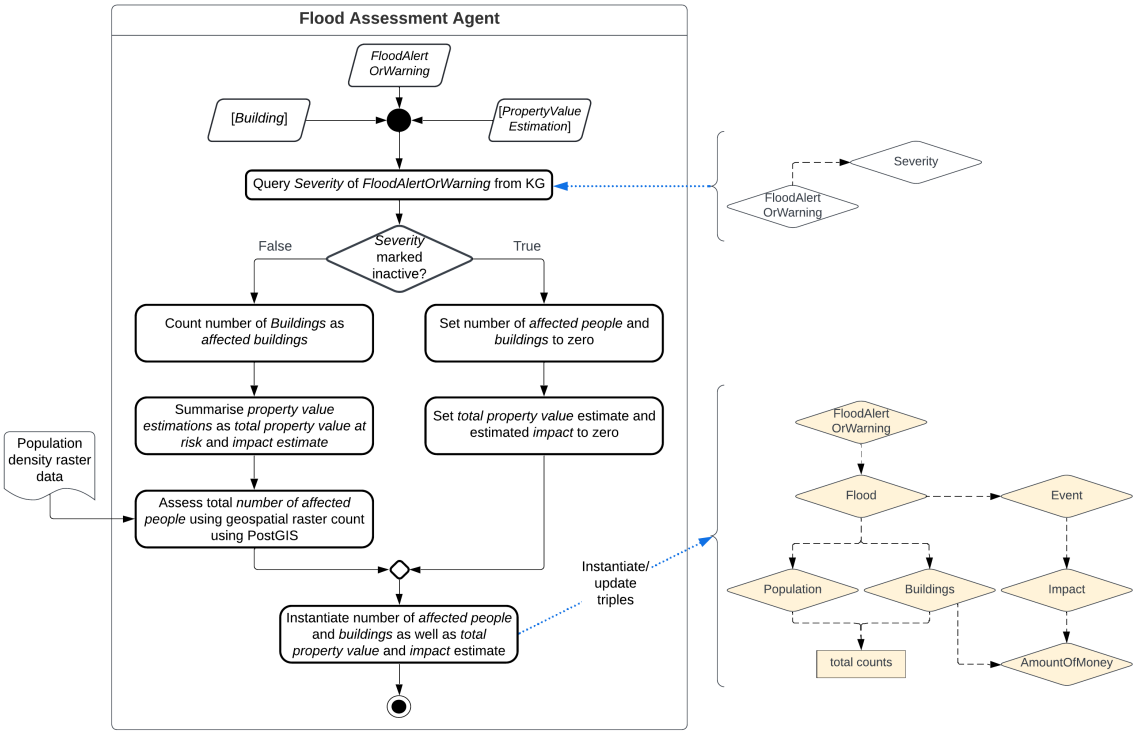


Figure 13: *Flood Assessment Agent. The Flood Assessment Agent uses the derived information framework to assess potential impacts of (anticipated) floods with regards to 1) number of people at risk, 2) number of buildings at risk, and 3) estimated building stock value at risk. The agent is implemented as asynchronous derivation agent which monitors the instantiated information within a specified triple store namespace at pre-defined frequency. Pure input instances required to evaluate the impacts of a flood warning are marked up as part of the instantiation of new flood alerts and warnings (see Fig. 11).*

Flood Assessment Agent The *Flood Assessment Agent* is a derivation agent to estimate the number of people and buildings as well as the total monetary value of the building stock at risk of flooding. The required input data for each flood assessment comprises a collection of `Building` and `PropertyValueEstimation` instances and the respective `FloodAlertOrWarning` instance itself. A detailed UML diagram is provided in **Fig.** 13, with key agent logic described below:
Initially the agent verifies whether the marked up input instances are suitable to perform a flood impact assessment, *i.e.*, that exactly one `FloodAlertOrWarning` instance is provided and the set of `PropertyValueEstimation` instances does not exceed the number of related `Building` instances. Missing property value estimations for some buildings as well as completely absent building and value estimation inputs are allowed (*i.e.*, however, would result in an impact assessment of zero).
Subsequently, the `Severity` of the `FloodAlertOrWarning` is retrieved. For inactive

flood warnings, all potential impacts are set to zero. Otherwise, the impact is assessed: 1) The number of people affected is determined by conducting a geospatial count over the population density raster data within the boundary of the `ArealExtentPolygon` associated with the `FloodAlertOrWarning` of interest. 2) The number of buildings at risk is assessed by summing up all Building IRIs that are marked up as potentially affected derivation inputs. And 3) the total monetary value at risk is estimated by retrieving and summing up the property market values from marked up `PropertyValueEstimation` IRIs. Finally, all potential flood impacts are instantiated according to OntoFlood as derivation outputs.

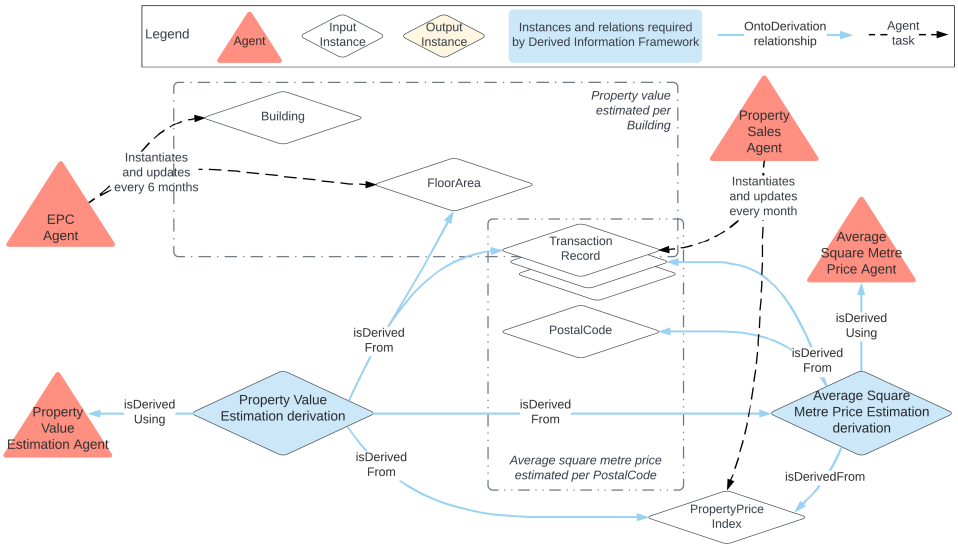
3.4 Derived Information Cascade

The automated flood impact assessment is based on the interplay of three autonomous derivation agents, namely the *Flood Assessment Agent*, the *Property Value Estimation Agent*, and the *Average Square Metre Price Agent*. In reverse order, these agents compute outputs which in turn act as input to the previously listed agent(s), resulting in a sequence of agent interactions to assess the potential impacts of a flood alert or warning.

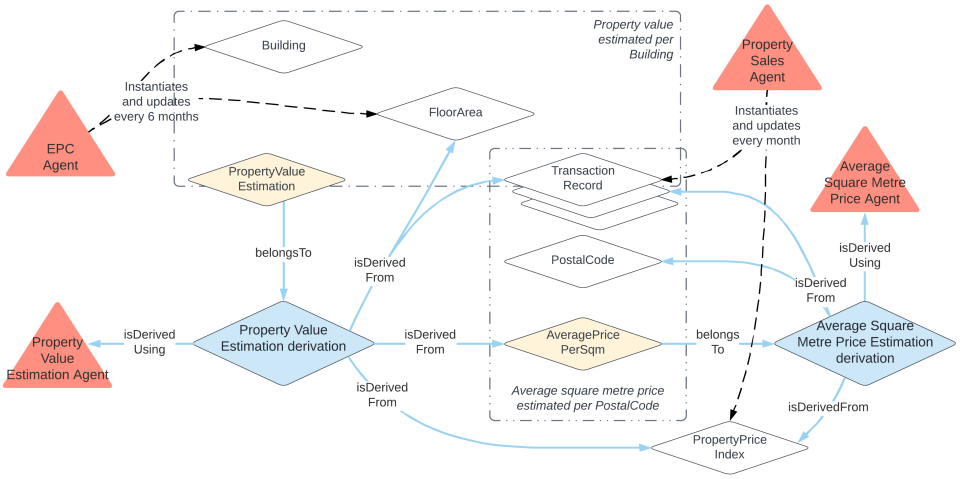
The `AveragePricePerSqm` is assessed on the postal code level and depends on the following inputs: 1) the `PostalCode` for which to assess the average price, 2) a list of previous property sales `TransactionRecords` in the applicable area, and 3) the representative `PropertyPriceIndex` for that postal code. The `PropertyValueEstimation` is assessed for each individual building and can either be derived based on 1) the latest `TransactionRecord` for that property (if available), and 2) again the representative `PropertyPriceIndex` or 3) the `FloorArea` of the given property, and 4) the representative `AveragePricePerSqm` for its location. As depicted in **Fig.** 14, this chains together both the *Property Value Estimation Agent* and the *Average Square Metre Price Agent* to estimate the current market value of a certain property.

The relevant real-world input data is assimilated on a monthly basis by both the *EPC Agent* and *Property Sales Agent*. While the initial derivation markup for both derivations is conducted as part of the latter one as depicted in **Fig.** 21, both agents ensure to update attached timestamps whenever they update an instantiated piece of information. There are two options to initially mark up the chained derivations: 1) solely instantiate the derivation markup and 2) instantiate the derivation markup and request an immediate instantiation of newly derived information. In the first case, the initial derivation assessment will only be conducted once the required outputs are requested by a user or agent (*i.e.*, at the next scheduled execution of the asynchronous derivation agent or upon HTTP request). Until then, the markup looks like depicted in **Fig.** 14(a)). The latter case (depicted in **Fig.** 14(b)) immediately triggers an initial derivation assessment and instantiates the respective outputs in the KG (incl. re-connecting the derivation markup). This scenario also reflects situations where derived information is found outdated and re-evaluated based on updated pure inputs.

As both the average price per square metre and the property market value assessments are relatively fast and computationally inexpensive, both derivations are instantiated subsequently as synchronous derivations for new information to immediately compute and in-



(a) Derivation markup at creation. In case the `AveragePricePerSqm` is not yet instantiated, the `PropertyValueEstimation` derivation can be instantiated with the upstream derivation as input. The upstream derivation is automatically evaluated and instantiated once the `PropertyValueEstimation` is requested.



(b) Derivation markup after instantiation. After initial evaluation and instantiation of the `AveragePricePerSqm`, the `PropertyValueEstimation` derivation is automatically reconnected to it instead of its derivation instance. Assessing the property value will now only trigger an update of the upstream derivation if the instantiated `AveragePricePerSqm` is outdated, otherwise, it remains unaltered.

Figure 14: *Derivation markup for* `PropertyValueEstimation`*. The estimated market value of any property depends on (i.e.,* `isDerivedFrom`*) the* `FloorArea` *and latest available* `TransactionRecord` *of the property as well as the* `AveragePricePerSqm` *and the* `PropertyPriceIndex` *of the associated postal code and administrative district, respectively (details in Fig. 24). As the* `AveragePricePerSqm` *is a derived quantity itself, there are two potential scenarios as detailed in Fig. 14(a) and Fig. 14(b).*

stantiate the newly derived information (*i.e.*, case 2). Hence, the `AveragePricePerSqm` shall already be instantiated when creating the `PropertyValueEstimation` markup, making the depicted case in Fig. 14(b) the default scenario. Since all required inputs can be assumed to be instantiated, this also eases the further markup for the overall flood impact assessment (depicted in **Figure** 15).

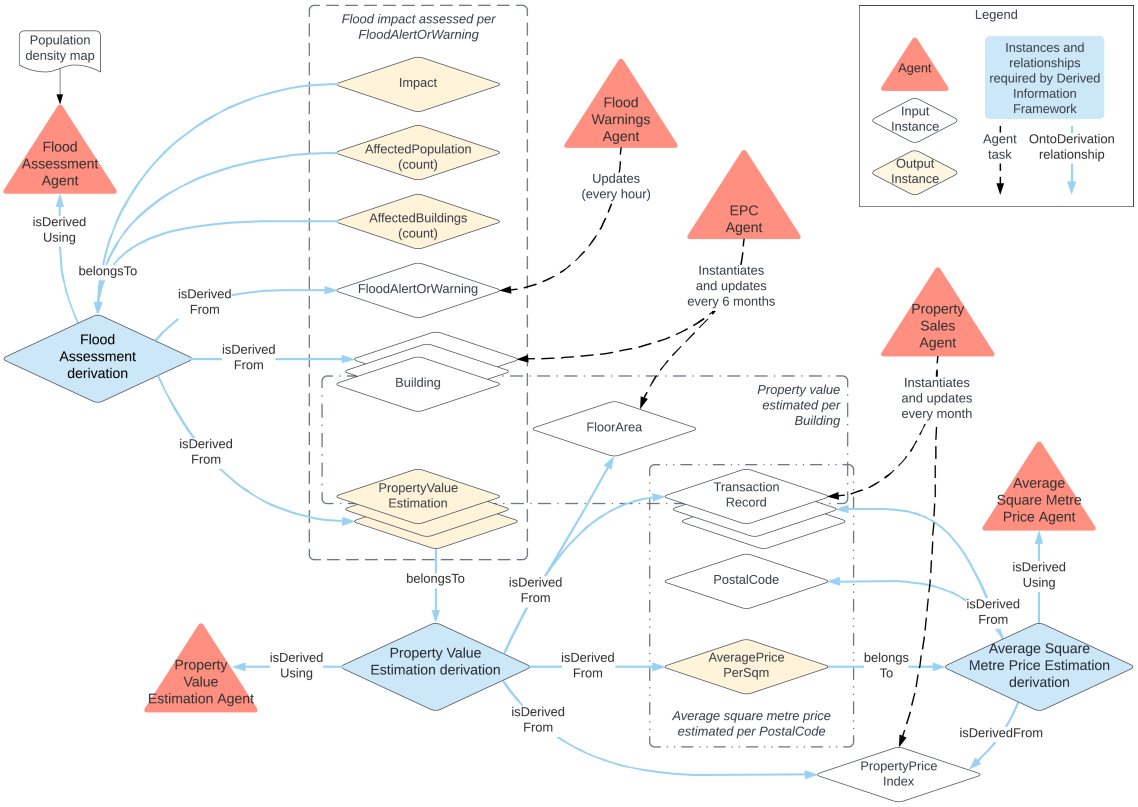


Figure 15: *Flood assessment markup (instantiated). The potential impact of a flood alert or warning (i.e., number of people, buildings and total property value at risk)* `isDerivedFrom` *the* `FloodAlertOrWarning` *instance itself as well as all* `Buildings` *and respective* `PropertyValueEstimations` *within the associated flood area. Depending on the status of the property value estimation derivation, a new flood assessment can trigger a cascade of up to three derivation agents in sequence: Flood Assessment Agent requiring input instances to be updated by the Property Value Estimation Agent, which in turn relies on outputs of the Average Square Metre Price Agent.*

The flood impacts assessment is evaluated for each instantiated `FloodAlertOrWarning` and depends on 1) the `FloodAlertOrWarning` itself as well as 2) a collection of `Building` instances and 3) associated `PropertyValueEstimation`. A flood warning/alert can stretch across multiple postal codes with thousands of buildings. Hence, the potential impact assessment is expected to take some time and is thus marked up as asynchronous derivation. The *Flood Assessment Agent* monitors the status of instantiated derivations with a pre-defined frequency to detect potentially outdated inputs which could affect the impact of a flood over the lifetime of the warning. Those inputs include both the

`Severity` and `ArealExtent` of the alert/warning as well as newly instantiated or updated property information, such as `FloorArea` or sales `TransactionRecords`, for potentially affected buildings or the `PropertyPriceIndex` for impacted administrative districts. When an update is requested, the *Flood Assessment Agent* uses a sequence of calls to the *Average Square Metre Price Agent* and *Property Value Estimation Agent* to update relevant derived information and generate a current flood impact estimate.

4 Implementation and Deployment

The overall implementation follows a containerised approach offering a flexible, scalable, and platform-independent way to deploy and manage individual agents. Each agent is implemented as individual Docker container and deployed to an overarching Docker stack enabling inter-container communication. The design supports both (remote) server and local deployment to facilitate a truly distributed knowledge graph with decentral hosting of data and computational capabilities. To ensure reliable service availability, the entire stack for this use case is deployed remotely in the cloud.

Within each stack, all agents have access to the same triple store (*i.e.*, Blazegraph) as well as an instance of PostGIS, GeoServer, and Ontop to overcome limitations in triple-store native handling of geospatial information (see section 2.6). The proposed approach stores geospatial information in PostGIS (*i.e.*, using standard libraries such as GDAL to perform data ingestion), while providing a corresponding OBDA mapping to allow access to the data according to the corresponding ontology using regular SPARQL syntax. Furthermore, PostGIS can be directly linked to GeoServer to support streaming as well as (dynamic) styling of relevant geospatial information to the Digital Twin Visualisation Framework (DTVF). Moreover, PostGIS is used as relational database by the *TimeSeriesClient* to store actual time series data.

The DTVF is used as uniform visualisation interface for all instantiated data. It is implemented in TypeScript and compiled into a single minified JS file (accompanied by a single minified CSS file). New visualisations can be created as new Docker containers by providing a few configuration files to import the remotely hosted DTVF library (along with other dependencies) as well as including further data and HTML elements as required. The used mapping provider Mapbox supports the visualisation of 2D data (with the option to extrude 2D polygons into basic 3D polyhedrons) from local files or from WMS endpoints (*e.g.*, served via GeoServer). Both meta and time series data for features of interest are queried directly from TWA and displayed on the side panel on the fly upon request.

4.1 Agent Deployment

The overall agent deployment consists of two phases, one rather manual preprocessing of geospatial building data together with the initial building instantiation and the autonomous recurring assimilation of further building data as well as additional dynamic data feeds.

During the initial building instantiation, several OS datasets need to be downloaded,

namely the BHA containing high-resolution building footprints and building height information, the OpenMap Local with coarser building footprints data, the Code-Point with Polygon containing postcode polygons, and the Digital Terrain Model DTM5 with detailed terrain raster data in 5m resolution. These datasets are processed using QGIS [81] to create a single consolidated shapefile containing all relevant building data: Both the BHA and OpenMap Local building footprints are merged to maximise building coverage as some buildings are only available in either of the datasets. In case of overlaps, the BHA data is used. Additionally, corresponding postcodes and building elevations are extracted from Code-Point polygons and DTM5, respectively, and added to each building node. Subsequently, FME [85] is used to convert the consolidated shapefile into CityGML format as required by the *Building Importer Agent*. This includes assigning the postcodes as Generic Attributes and extruding each building based on its height information. Finally, the created CityGML file is provided to the *Building Importer Agent* to be instantiated according to OntoCityGML. The *TSD Agent* is then called to identify and instantiate both thematic surfaces and LoD0 building footprints before the *UPRN Agent* attaches UPRN

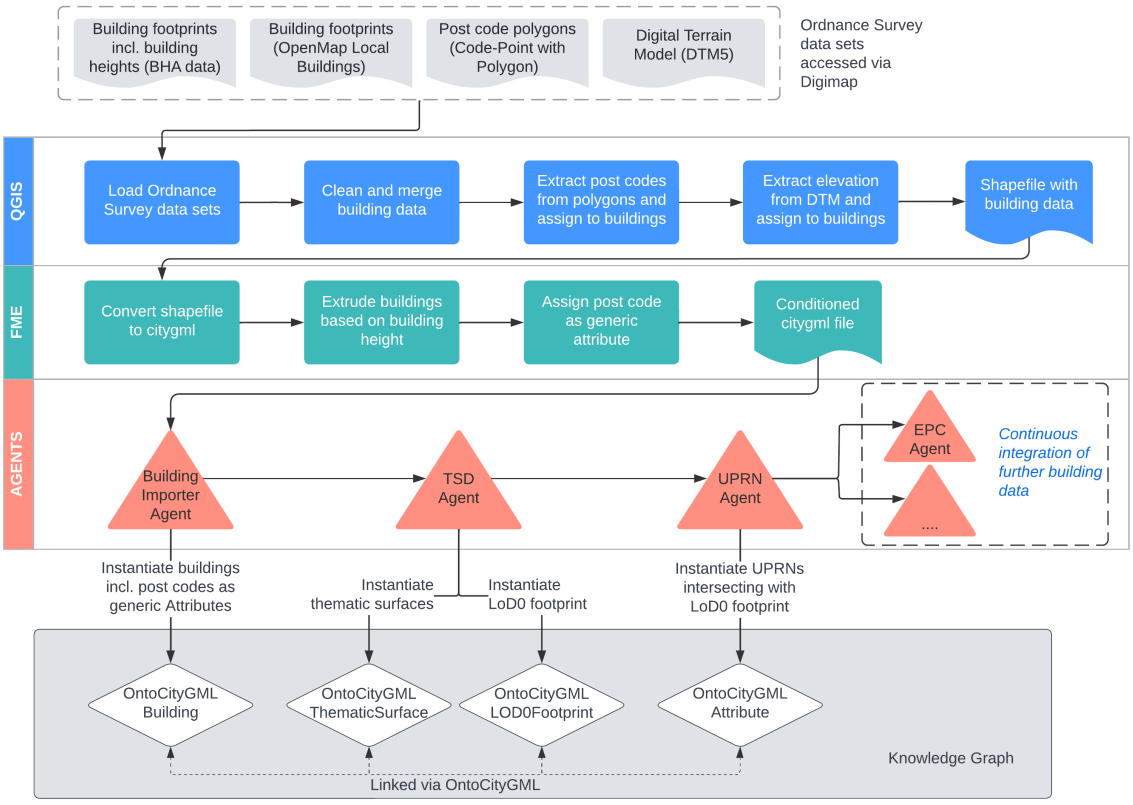


Figure 16: *Building instantiation workflow. The instantiation of the Base World currently still requires some manual steps, i.e., geospatial processing using QGIS and FME. After instantiating the buildings using the Building Importer Agent, also the Thematic Surface Discovery and UPRN Agent need to be invoked manually to ensure geospatially represented buildings have UPRNs attached (if available). Further building data instantiation (i.e., EPC Agent, Property Sales Agent) happens automatically once the respective agents are deployed.*

information from the OS Feature API to all buildings based on their LoD0 footprints. This marks the end of the manual building data instantiation and the entire workflow is schematically outlined in **Fig.** 16.

After the initial geospatial building instantiation, all remaining agents can be deployed to the Docker stack. The *EPC Agent*, which initialises non-OntoCityGML building representations, needs to be deployed first, before any additional building data can be instantiated. While all OntoEMS instantiation agents can populate individual triple store namespaces, all EPC, property sales, and flood warnings data need to get instantiated into the same namespace to enable the derivation agents to monitor and detect relevant updates. Start-up of the *Met Office*, *Air Quality* and *River Level Agents* automatically registers a recurring background task to assimilate latest data once per day. Similarly, the *EPC* and *Property Sales Agent* ingest latest building related data on a monthly basis, while current flood warnings are assimilated hourly by the *Flood Warnings Agent*.

4.2 Consolidated Visualisation of Base World

Based on ontologies to represent both data and inherent knowledge, TWA provides a semantic ecosystem to interact with a variety of data from previously isolated sources in an aligned format (*i.e.*, using SPARQL). TWA creates a rich *Base World* by combining both rather static (*i.e.*, buildings) and dynamic (*i.e.*, environmental measurements and flood warnings) data in one single system, enriching individual pieces of information with meaning and providing additional context.

Figure 17: *Consolidated visualisation. The Digital Twin Visualisation Framework provides a uniform interface to retrieve and visualise data from TWA. It creates an aligned visualisation of previously isolated data sources side by side to foster fact-based decision-making and enable insights across domains.*

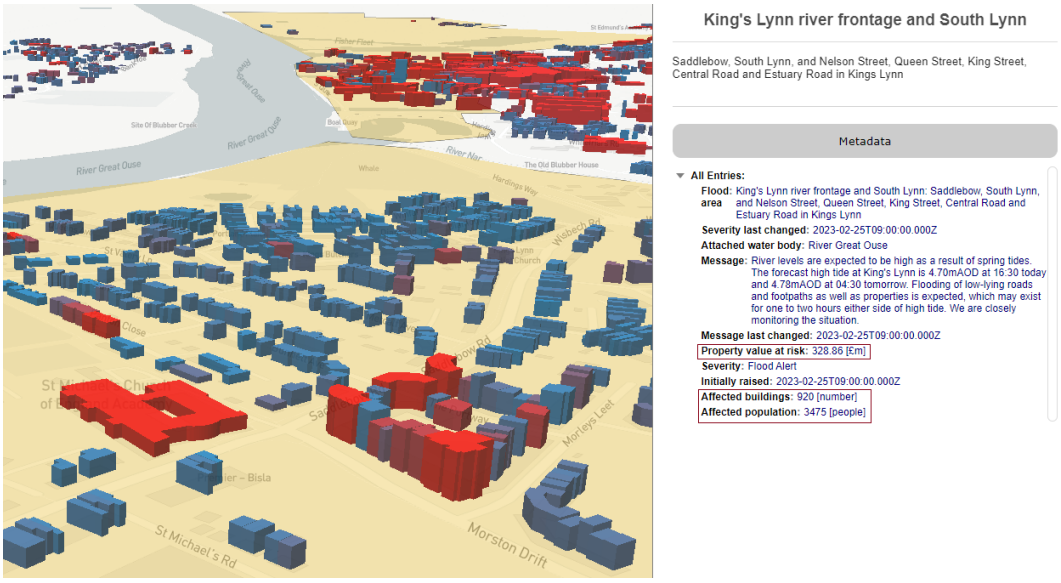
The Digital Twin Visualisation Framework provides a web-based interface to visualise data from TWA. **Fig.** 17 shows an example visualisation of the current use case in the smart city context, combining consolidated building information with environmental measurements and forecasts from various data providers. It offers a mutual and aligned visualisation of previously isolated weather, air quality, and river level data, including information about associated measuring stations as well as corresponding time series data. This empowers citizens to see related information (*e.g.*, river level readings as well as current and projected rainfall data or air pollutant concentration measurements and projected wind conditions) in one single system, as compared to various independent locations or websites. The available building data provides detailed information about key construction characteristics, energy performance, usage types, previous and current market value estimates, *etc.* and is maintained by a set of continuously running input agents to keep it current in time. The integrated map-based visualisation enables both geospatial and time series analyses, *e.g.*, to understand building usages in a certain area.

4.3 Cross-domain Flood Impact Assessment

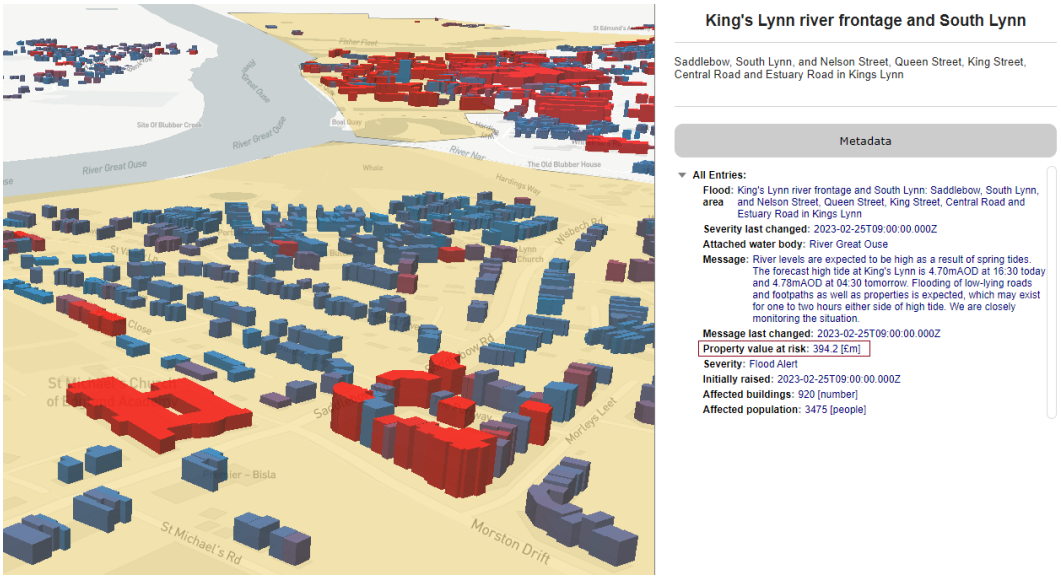
The *Flood Assessment Agent* continuously monitors a pre-defined triple store and automatically re-evaluates any outdated flood impact derivation instances. **Fig.** 18 illustrates this capability by showing an initial flood impact assessment as well as an updated one after an increase in property prices side by side: Initially, the *Flood Warnings Instantiation Agent* ingests a new flood alert into TWA and marks it up as an asynchronous derivation. The next time the derivation agent processes all requested derivations, it estimates the potential flood impacts of the raised alert. This initial flood impact assessment indicates that the issued alert may affect 3,475 people living in the associated flood area, with 920 buildings at risk, valued at £328.9m (Fig. 18(a)). The colors in the figure represent property market value estimates, with red indicating high and blue indicating low property prices.

After the initial assessment, a 20% increase in property price index is instantiated to simulate a rise in the value of residential properties. This update results in all property value estimations becoming outdated and, hence, similarly affects the dependent flood assessment derivation instance. The *Flood Assessment Agent* automatically detects and updates the affected flood assessment derivation instance the next time it monitors the triple store. The new evaluation still shows that 3,475 people and 920 buildings are at risk; however, the total property value has increased to £394.2m (Fig. 18(b)). It needs to be noted that this increase is rather arbitrary and mainly for visualisation purposes; however, monthly UKHPI adjustments will automatically be reflected in subsequent flood assessments.

Although this simulated change is rather arbitrary, it showcases the overall capabilities. Similarly, updated flood alert/warning data (*i.e.*, change in severity, warning message details, *etc.*), newly instantiated buildings, or outdated property value instances would result in an automatic re-assessment. Whenever a flood alert is marked as inactive at the flood-monitoring API, potential impacts are assessed as zero and all assessment triples are ultimately deleted from TWA once the flood alert vanishes from the API.



(a) Initial assessment. The property value at risk is estimated with £328.9m



(b) Updated assessment. The change in property price index triggers an automatic re-assessment of total property value at risk, which is now estimated with £394.2m

Figure 18: *Automated flood assessment. The automated re-evaluation of potential flood impacts is depicted for subsequent assessments of the same flood alert with a simulated property price index hike of 20 % in between. Fig. 18(a) depicts the initial assessment of properties at risk. Subsequently, the updated* `PropertyPriceIndex` *is identified by the Flood Assessment Agent triggering an update of all* `PropertyValueEstimations`*. Corresponding changes to the overall property value at risk are shown in Fig. 18(b). Property market value changes can be seen from updated colors of individual buildings (very mildly though) or easier from the side panel in aggregated form.*

Limitations Despite providing this proof-of-concept for a semantic software agent-based automated flood assessment, several limitations of the work shall be emphasised: Firstly, the assessment does not include all buildings, but only considers the actually instantiated share. As the building instantiation workflow is based on EPC data, only buildings with available EPC information are part of the analysis; however, it has been observed, that EPC data is only available for slightly above 50% of the buildings in the vicinity of King's Lynn [50], which results in an expected underprediction of monetary impact from flood warnings. Secondly, the current approach to estimating property market values is rather rudimentary. While this is sufficient for a proof-of-concept, it overlooks several crucial factors which can significantly affect property prices, such as usage classification (*i.e.*, domestic vs. non-domestic), retrofitting, and micro-location, just to name a few. Hence, the results shall only be interpreted as rough estimates and the evaluation approach shall be refined in the future to allow for a more elaborate assessment. Thirdly, the current impact assessment is limited to the vulnerable share of population as well as the number and value of potentially affected buildings, while other aspect of the built environment (*e.g.*, roads or further network assets) are currently neglected. Lastly, it should be noted that the process of enriching instantiated buildings with corresponding sales transactions relies on fuzzy address matching between EPC and HM Land Registry data, as UPRNs are not provided in Land Registry's transaction record data. Unfortunately, free text address information are not always provided in an aligned fashion. For instance, mismatches in address lines, the use of building names instead of numbers, and differences in granularity with regards to building (sub-)numbers have been observed. Such discrepancies can result in erroneous instantiations of previous sales transactions, either by attaching a transaction record to the wrong building instance (*i.e.*, another building with higher confidence score of fuzzy match) or not instantiating a transaction record for a building instance at all (*i.e.*, required confidence score of fuzzy match not reached). However, it needs to be noted that various scoring methods and confidence scores have been screened to minimise this problem.

5 Conclusions

This work proposes a collection of agents that is able to synthesise multiple previously isolated data feeds into an automated impact assessment for imminent flood hazards. Three connected ontologies are developed to dynamically instantiate a semantically rich representation of buildings, various environmental observations and flooding related information. An ecosystem of autonomous input agents is proposed and deployed to assimilate multiple data feeds from disconnected sources, including both rather static and near real-time data, to ensure the World Avatar's *Base World* remains current in time. The capabilities to provide cross-domain insights and decision support for smart city and disaster management are demonstrated by bringing together live data about potential flood events with information about buildings as well as further environmental observations in their vicinity. A unified visualisation interface is provided to interact with and analyse the instantiated data. Beyond the visualisation, this work provides access to all incorporated publicly available data sources via one single SPARQL endpoint, using aligned knowledge models based on the developed ontologies.

Furthermore, the agent-based autonomous cascading of information enabled by the derived information framework is demonstrated: a re-assessment of potential flood impacts is triggered and automatically executed whenever a new flood alert or warning is instantiated or instantiated information for any relevant input is updated. The impact of potential flooding events is assessed with regards to the number of affected people as well as the number and estimated market value of buildings at risk. Together with the ecosystem of continuously running input agents, this ensures that the World Avatar's *Base World* provides up-to-date insights into potential flooding consequences at all times. While currently only assessing people and assets at risk, the demonstrated information cascading infrastructure provides a promising approach to fully automate smart city workflows and foster interoperability. In disaster management, this is expected to improve timely decision-making and initiate proactive measures to mitigate adverse effects of imminent flood hazards, ultimately contributing to the safety and well-being of vulnerable population and infrastructure. The proposed system is easily deployable to any other city in the UK, and this work is currently ongoing.

Research data

All the codes developed are available on The World Avatar GitHub repository https://github.com/cambridge-cares/TheWorldAvatar:

The latest versions of all developed ontologies are available in the JPS_Ontology sub-directory of the Github repository.

Detailed deployment instructions to reproduce the work are available in the Kings Lynn Project sub-directory of the Github repository. The source code of all referenced agents is available in the Agent sub-directory.

Acknowledgements

This research was supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. A part of this study has been undertaken in the context of DOME 4.0 project, which has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 953163. M. Hofmeister acknowledges financial support provided by the Cambridge Trust and CMCL. J. Bai acknowledges financial support provided by CSC Cambridge International Scholarship from Cambridge Trust and China Scholarship Council. M. Kraft gratefully acknowledges the support of the Alexander von Humboldt Foundation.

Attribution statements:

This work contains OS data © Crown copyright and database rights 2022 (100025252). Furthermore, it contains HM Land Registry data © Crown copyright and database right 2020 as well as public sector information licensed under the Open Government Licence

v3.0. Lastly, this showcase uses Environment Agency flood and river level data from the real-time data API (Beta)

For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

Nomenclature

ABox Assertional Component (of an ontology)
API Application Programming Interface
BHA Building Height Attribute (OS Premium dataset)
CEA City Energy Analyst
DL Description Logic
DOLCE Descriptive Ontology for Linguistic and Cognitive Engineering
DTVF Digital Twin Visualisation Framework
EA Environment Agency
ENVO Environmental Ontology
EPC Energy performance certificate
GEMET General Multilingual Environmental Thesaurus
GeoSPARQL Geographic Query Language for RDF Data
IRI Internationalised Resource Identifier
KG Knowledge Graph
LoD Level of Detail
MEMOn Modular Environmental Monitoring (ontology)
OBDA Ontology-Based Data Access
OGC Open Geospatial Consortium
ONS Office for National Statistics
OS Ordnance Survey
OWL Web Ontology Language
PPD (HM Land Registry's) Price Paid Data
RDF Resource Description Framework
SAREF Smart Appliances REFerence (ontology)
SOSA Sensor, Observation, Sample, and Actuator (ontology)
SPARQL SPARQL Protocol and RDF Query Language
SSN Semantic Sensor Network (ontology)
SWEET Semantic Web for Earth and Environmental Terminology
TBox Terminological Component (of an ontology)
TWA The World Avatar
UK-AIR UK Air Information Resource
UKHPI UK House Price Index
UML Unified Modeling Language

UPRN Unique Property Reference Number
URI Uniform Resource Identifier
W3C World Wide Web Consortium

A Appendix

A.1 Namespaces

bot: <https://w3id.org/bot#>
dabgeo: <http://www.purl.org/oema/infrastructure/>
bimerr: <http://bimerr.iot.linkeddata.es/def/building#>
db: <http://www.google.com/digitalbuildings/0.0.1/facilities#>
deriv: <https://www.theworldavatar.com/kg/ontoderivation/>
envo: <http://purl.obolibrary.org/obo/>
geo: <http://www.opengis.net/ont/geosparql#>
geolit: <http://www.bigdata.com/rdf/geospatial/literals/v1#>
icity: <http://ontology.eil.utoronto.ca/icity/Building/>
ifc: <https://standards.buildingsmart.org/IFC/DEV/IFC4/ADD2_TC1/OWL#>
icontact: <http://ontology.eil.utoronto.ca/icontact.owl#>
lrppi: <http://landregistry.data.gov.uk/def/ppi/>
m3l: <http://purl.org/iot/vocab/m3-lite#>
om: <http://www.ontology-of-units-of-measure.org/resource/om-2/>
ocgml: <http://www.theworldavatar.com/ontology/ontocitygml/citieskg/OntoCityGML.owl#>
ontobuiltenv: <https://www.theworldavatar.com/kg/ontobuiltenv/>
ontoems: <https://www.theworldavatar.com/kg/ontoems/>
ontoflood: <https://www.theworldavatar.com/kg/ontoflood/>
ontouom: <https://www.theworldavatar.com/kg/ontouom/>
owl: <http://www.w3.org/2002/07/owl#>
rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
rdfs: <http://www.w3.org/2000/01/rdf-schema#>
rt: <http://environment.data.gov.uk/flood-monitoring/def/core/>
sio: <http://semanticscience.org/resource/>
skos: <http://www.w3.org/2004/02/skos/core#>
soph: <http://sweetontology.net/phen/>
sophhy: <http://sweetontology.net/phenHydro/>
time: <http://www.w3.org/2006/time#>
ts: <https://www.theworldavatar.com/kg/ontotimeseries/>
weather: <https://www.auto.tuwien.ac.at/downloads/thinkhome/ontology/WeatherOntology.owl#>
xsd: <http://www.w3.org/2001/XMLSchema#>

A.2 Agent UML Activity Diagrams

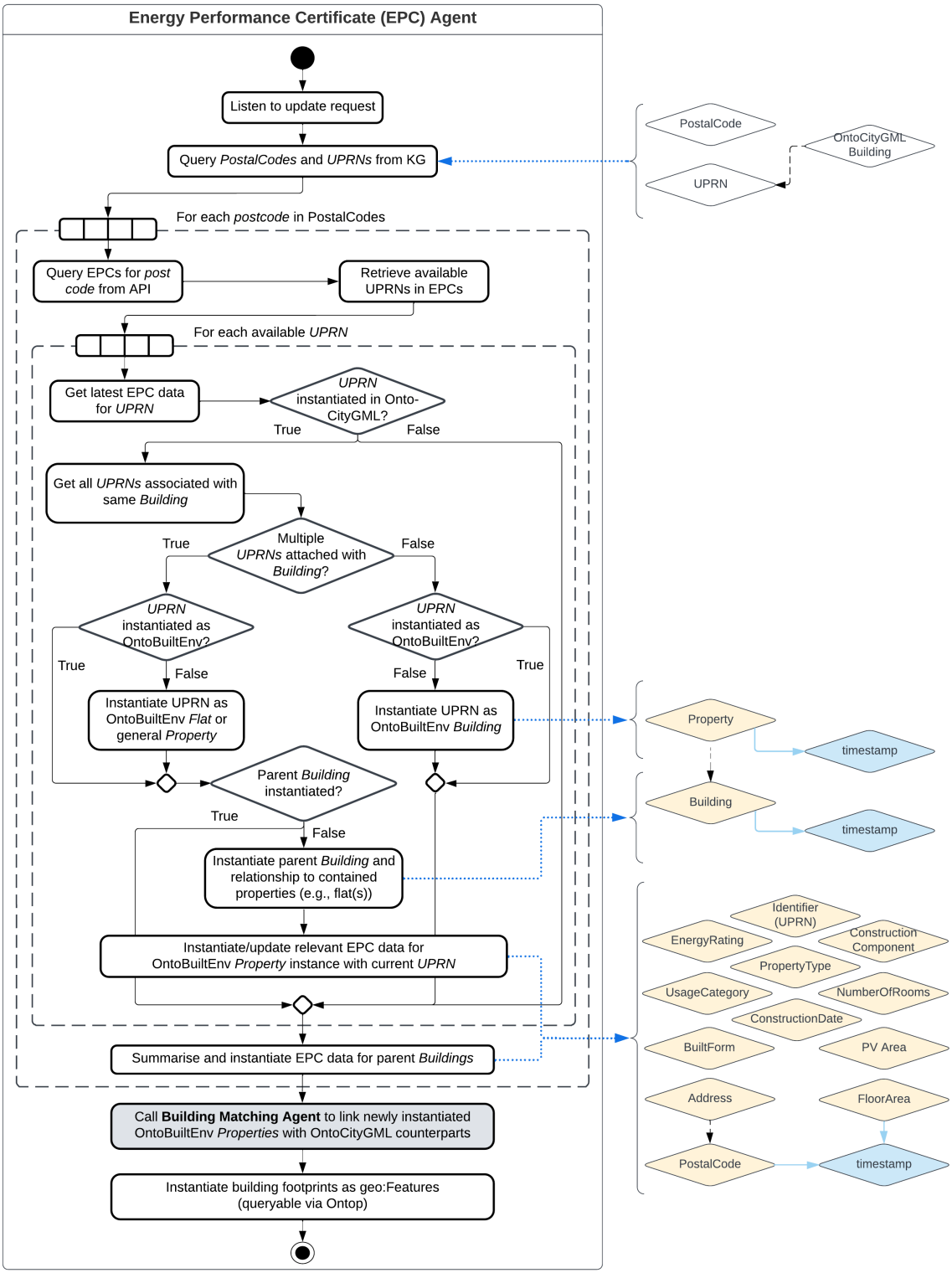


Figure 19: *EPC Agent. The EPC Agent continuously assimilates latest EPC data from all three EPC APIs [32] (i.e., for domestic, non-domestic, and public properties) for all OntoCityGML buildings with instantiated UPRN information (e.g., on monthly basis). For each instantiated UPRN with available EPC data, an OntoBuiltEnv* `Property` *is instantiated/updated with the respective information. ...*

Figure 19: *[Continued caption:] In case multiple UPRNs are associated with one OntoCityGML building, a parent* `Building` *is instantiated to accommodate all associated* `Property` *instances. The Building Matching Agent is required to link corresponding building instances in OntoBuiltEnv and OntoCityGML (details in Fig. 20).*

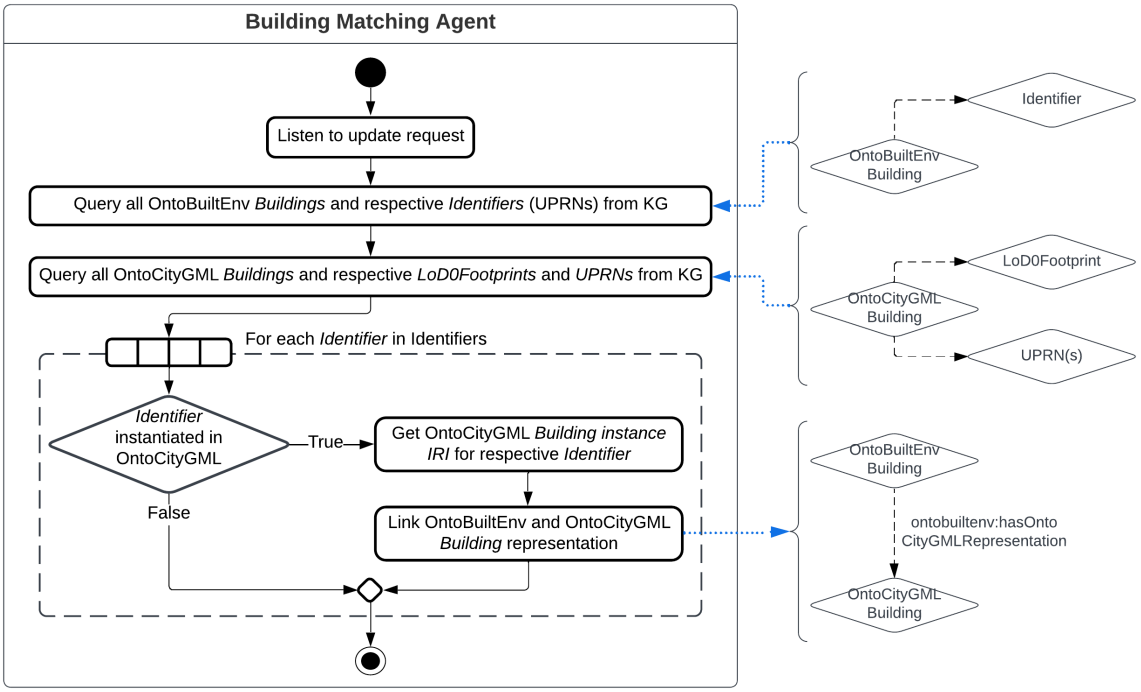


Figure 20: *Building Matching Agent. The Building Matching Agent is used to link a building instantiated in OntoBuiltEnv (i.e., building instances based on available EPC data) to its corresponding instance in OntoCityGML (i.e., building instances based on OS data). The link is created for buildings only (i.e., excluding flats which do not have a geospatial representation in OntoCityGML) by using UPRNs as the identifiers.*

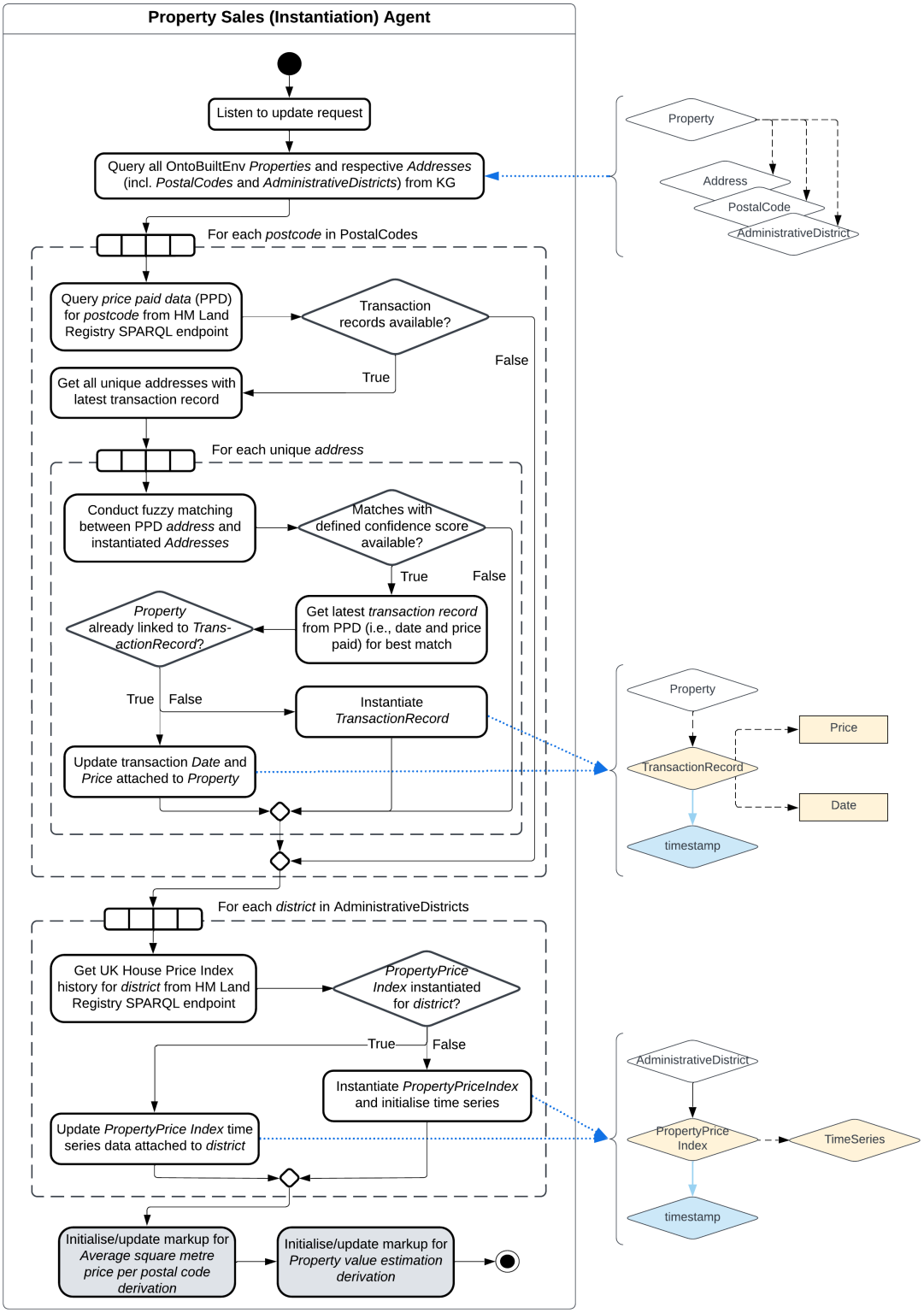


Figure 21: *Property Sales Agent. The Property Sales Agent queries His Majesty's Land Registry Open Data [46] via its public SPARQL endpoint [49] and instantiates latest available* `TransactionRecords` *for instantiated* `Properties` *as well as the* `PropertyPriceIndex` *(i.e., UKHPI) . After instantiating ...*

Figure 21: *[Continued caption:] new property sales data, the agent also instantiates the relevant derivation markups to allow for automatic assessment of the* `AveragePricePerSqm` *per postal code as well as the* `PropertyValue-Estimation` *per dwelling (for details see Figs. 23 and 25, respectively.)*

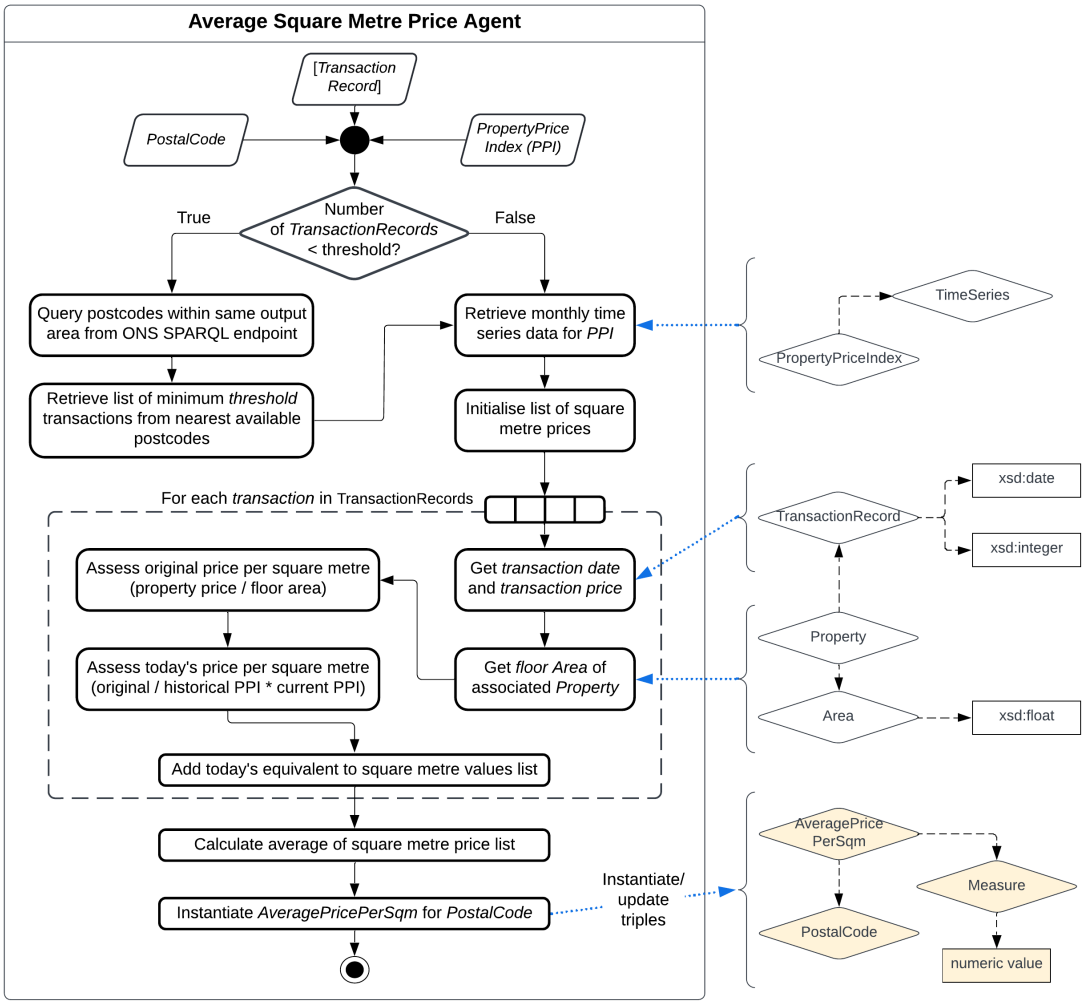


Figure 22: *Average Square Metre Price Agent. The Average Square Metre Price Agent calculates the current average square metre price of properties (i.e., buildings and flats) within each instantiated postal code. The required building data comprises* `FloorAreas` *(from EPC Agent) as well as previous sales* `TransactionRecords` *and the* `PropertyPriceIndex` *(from Property Sales Agent). In case of an insufficient number of instantiated* `TransactionRecords` *per* `PostalCode`*, data from surrounding postal codes are included. The agent is implemented using the derived information framework to trigger automated re-assessment once inputs become outdated. The required derivation markup is part of the Property Sales Agent and detailed in Fig. 23.*

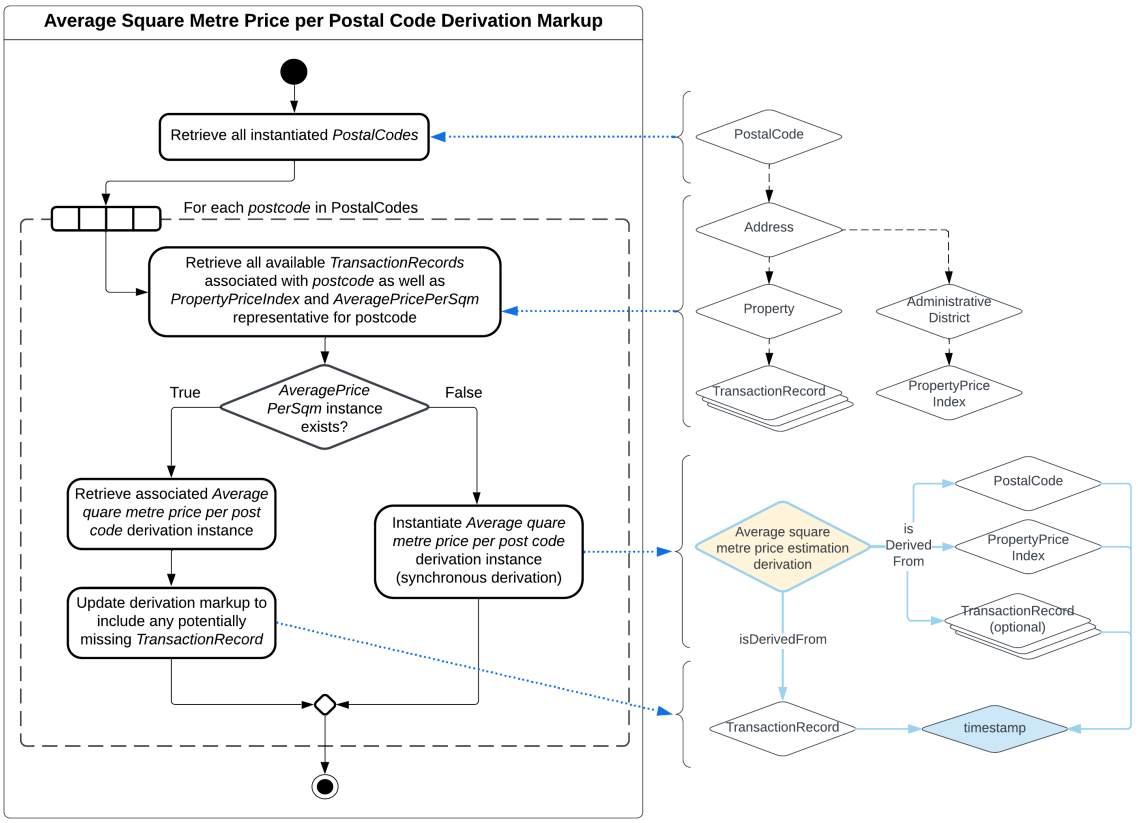


Figure 23: *Average Square Metre Price Derivation Markup. The derivation markup method connects all available* `TransactionRecords` *within each* `PostalCode` *with the respective* `AveragePricePerSqm` *derivation instance for that postal code: In case no derivation instance is instantiated yet, a synchronous derivation for new info is instantiated to request an initial assessment of the average price immediately. Otherwise, potentially missing* `TransactionRecords` *(i.e., newly instantiated data) are added to the existing derivation instance and a subsequent update is requested.*

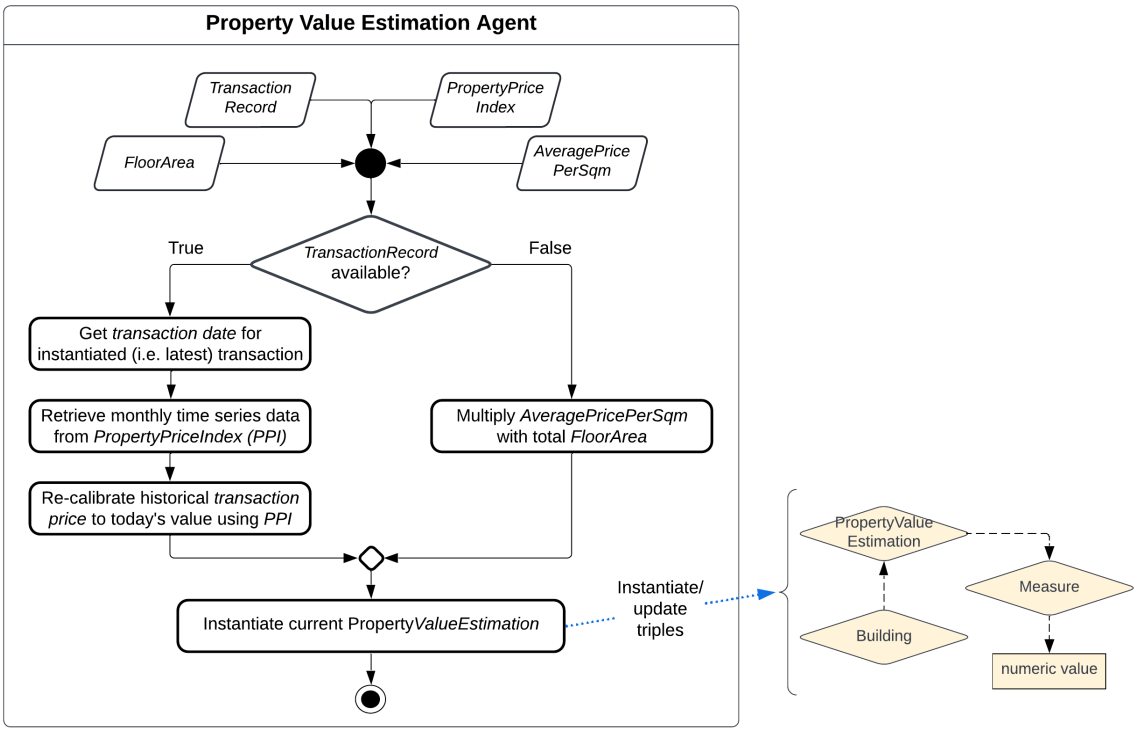


Figure 24: *Property Value Estimation Agent. The Property Value Estimation Agent assesses the current market value of instantiated properties. The assessment requires either 1) a previous sales* `TransactionRecord` *for the property of interest as well as the* `PropertyPriceIndex` *(both from Property Sales Agent) or, in cases where no previous sales transactions are available, 2) the* `FloorArea` *(from EPC Agent) together with the current* `AveragePricePerSqm` *for the respective* `PostalCode` *(from Average Square Metre Price Agent). The agent is implemented using the derived information framework to trigger automated re-assessment once inputs become outdated. The required derivation markup is part of the Property Sales Agent and detailed in Fig. 25.*

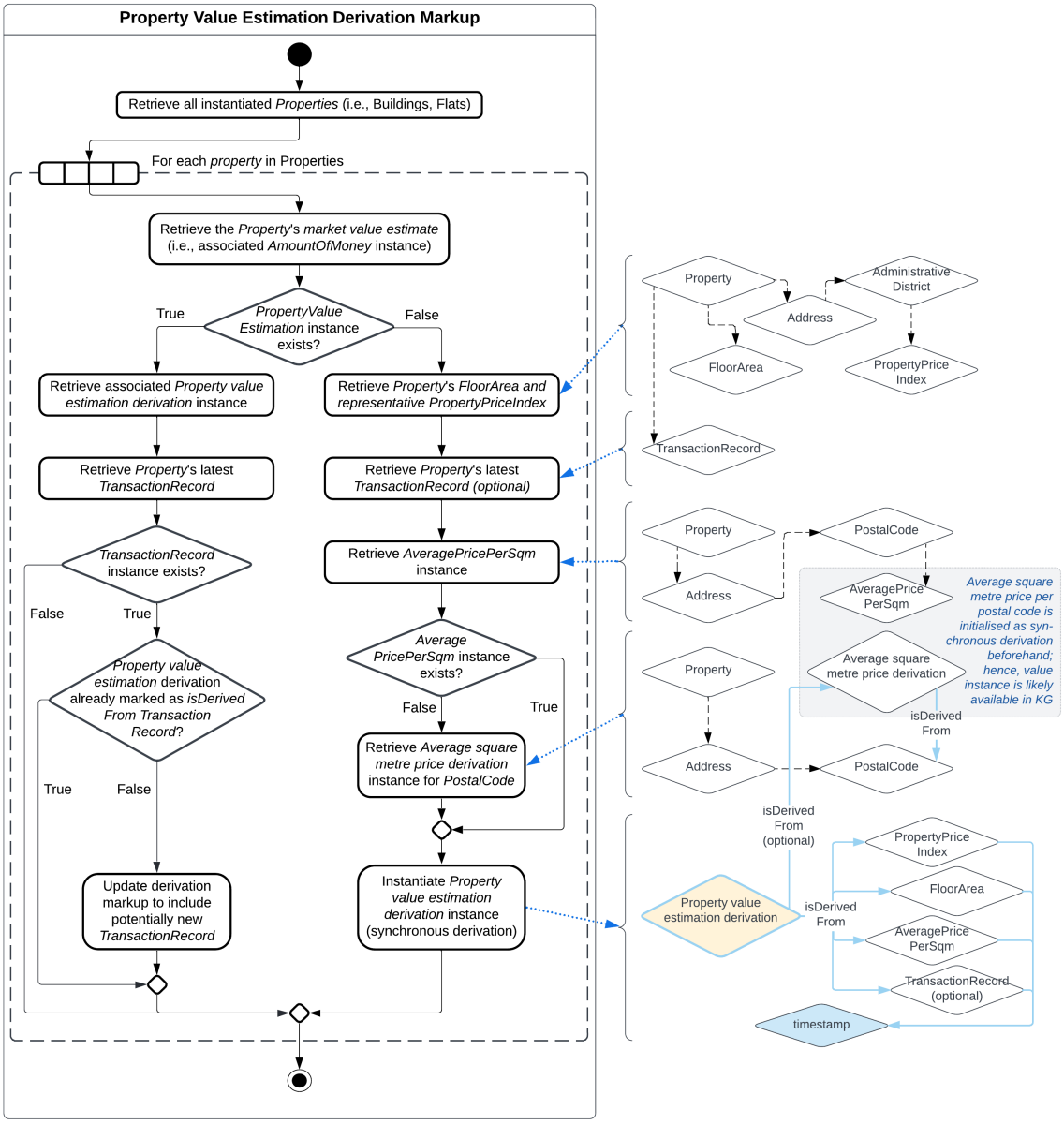


Figure 25: *Property Value Estimation Markup. The derivation markup method connects the* `FloorArea` *and previous* `TransactionRecord` *(if available) of a* `Property` *as well as the applicable* `AveragePricePerSqm` *(per postal code) and* `PropertyPriceIndex` *(per district) with the respective* `PropertyValueEstimation` *derivation instance for this dwelling: In case no derivation instance is instantiated yet, a synchronous derivation for new info gets instantiated to request an initial estimation of the property value. If the* `AveragePricePerSqm` *has not been computed yet, the property value derivation gets connected to the upstream average price derivation (although this is very unlikely due to instantiation as synchronous derivation for new info). For existing derivation instances, a potentially new* `TransactionRecord` *(i.e., newly instantiated data) gets added and a subsequent update is requested.*

A.3 Public Data Source Details

A.3.1 Environmental Observations and Flood Data

The Met Office DataPoint API provides open access to both current weather observations and forecasts of future weather conditions [65]. The service offers predictions for around 6,000 and actual observations for approximately 140 locations throughout the UK. Forecasts cover a five-day period while observations encompass the previous 24 hours, including typical parameters such as temperature, wind speed and direction, humidity, and air pressure.

The Environment Agency provides several API endpoints with (near-real time) information related to flooding and flood risk: The Real Time flood-monitoring API [30] provides a listing of all current flood alerts and warnings, including relevant meta information (*e.g.*, severity, associated water bodies, and warning message), and is updated every 15 minutes. The same API also provides an endpoint for live readings of water levels and flows recorded at various measuring stations along rivers and other water bodies. Precipitation data is also integrated and provided for nearly 1000 telemetry-connected tipping bucket rain gauges. The API furnishes metadata on these stations and the diverse measurements available at each one, as well as the actual readings themselves. Beyond those (near) real-time information, the Hydrology API also provides past, and (partially) quality controlled, hydrological data about river levels, river flows, groundwater levels, precipitation, and water quality [31]. The Flood Forecasting Centre generates a daily flood risk forecast, with more frequent updates issued when severe flooding is anticipated [43]. The forecast assesses the likelihood of flooding for a period of five days and covers flooding from rivers, the sea, surface water and groundwater for England and Wales. Compared to immediate flood alerts and warnings, this information is associated with higher uncertainty, both with regards to areal extend and anticipated severity.

The UK Air Information Resource (UK-AIR) provides real-time air quality data for various pollutants, including nitrogen dioxide, particulate matter, and ozone. The data is collected from a network of monitoring stations located throughout the UK and publicly provisioned via a machine readable Sensor Observation Service [91]. Parts of the data are enriched with meta information about reported pollutants according to the European GEMET (General Multilingual Environmental Thesaurus) vocabulary [41].

A.3.2 Building Data

The Ordnance Survey (OS) is the national mapping agency of Great Britain responsible for producing and disseminating geospatial data for government and public use [77]. The OS MasterMap™ is recognised as the most detailed available large-scale mapping of the UK, covering administrative boundaries, postcodes, OS identifiers, detailed digital terrain models (DTM), and building data, including the Building Height Attribute (BHA) dataset. The OS BHA data provides detailed information about the physical characteristics of the built environment, more precisely vector specifications for the base polygon, base elevation and height of individual buildings as well as other related attributes such as the number of floors and an open, officially maintained identifier, called Unique Property

Reference Number (UPRN) to cross-link information across datasets. The UPRN is a unique identifier assigned to every addressable location in the UK, including residential and commercial dwellings, as well as other types of buildings and structures. It provides a consistent and reliable way to identify and reference specific locations, regardless of changes in the property name or address. The BHA data is derived from a combination of remote sensing techniques such as LiDAR (Light Detection and Ranging) and aerial photography, and is processed using advanced algorithms to produce highly accurate and detailed height measurements. Although restricted redistribution and complex licensing arrangements make it not ideal for TWA, multiple relevant subsets are made accessible via Digimap [37] for educational and research purposes. OS data is either downloaded via Digimap (*i.e.*, terrain and BHA data) or accessed via the OS Features API (*i.e.*, *UPRN Agen*t).

The Department for Levelling Up, Housing & Communities offers open Energy Performance Certificate (EPC) data [32] to increase transparency on the energy efficiency and carbon emissions of individual buildings in the UK and to provide improvement recommendations. There are three dedicated APIs for domestic, non-domestic and display (*i.e.*, mostly public buildings) certificates providing property-level information about current and potential energy efficiency, key construction characteristics (*i.e.*, number of rooms, total floor area, building type, *etc.*), high-level usage classification as well as address and location details, including UPRN. New or updated information is released every four to six months and can be directly linked to OS building information via UPRN matching.

His Majesty's Land Registry publishes several public datasets related to residential property sales on a monthly basis, and even makes them available as Linked Data via its public SPARQL endpoint [49]: The UK House Price Index [48] captures the monthly change in the value of residential properties in England and Wales, on different levels of granularity, covering national, county, and local authority scale. The Price Paid data [47] contains information about the actual transaction price and date of individual residential properties sold in England and Wales, including address, type of property, and tenure; however, without explicit UPRN information. Due to licensing constraints (*i.e.*, around OS AddressBase), open transaction data can only be linked to OS building information via (fuzzy) address matching.

A.3.3 Population Data

The OpenPopGrid [68] provides an open gridded population dataset for England and Wales based on the ONS 2011 Census (*i.e.*, Output Area boundaries, Postcode headcounts, ONS Postcode Directory) as well as OS OpenData (*i.e.*, VectorMap District). It aims to enhance the spatial accuracy of the ONS population dataset by utilising an asymmetric mapping technique that limits the redistribution of population to particular regions, such as residential buildings, through the use of supplementary data. Specifically, the census postcode headcounts are redistributed across a grid, which is based on the OS VectorMap District buildings dataset, which prior has been carefully screened to remove non-residential areas using unpopulated postcode centroids. It is ensured that population values remain consistent with the Census data when aggregated at the Output Area level.

A.4 Ontology Review

A.4.1 Sensor and Measurement Ontologies

Sensor ontologies provide a formal way to represent sensor related concepts and relationships to enable interoperability and integration among different sensor systems with non-homogeneous data acquisition and monitoring techniques:

The Semantic Sensor Network (SSN) ontology [96] is designed to describe sensors and their observations, samples and procedures used, observed properties, and actuators. It utilises a modular architecture, based on the Sensor, Observation, Sample, and Actuator (SOSA) ontology [51] as its core, which allows for flexible use and covers a wide range of applications. The SOSA ontology provides a general-purpose foundation for sensor related applications with defining elementary classes and properties, while the SSN ontology incorporates more specialised and domain-specific concepts to provides a more comprehensive ontology for describing sensors and their observations in greater detail. The SSN ontology is developed and maintained by the W3C, making it widely recognised and adopted, and its well-designed modular architecture enables flexibility and ease of re-use (even individual components). While its comprehensive coverage of standardisation is one of the key advantages, some aspects of the ontology may be overly complex for certain use cases. The SOSA ontology, on the other hand, provides a lightweight, modular, and self-contained core ontology for describing basic concepts related to sensors, observations, and actuators to foster re-use by simplicity compared to other ontologies. However, one of the challenges of SOSA is its limited coverage, which may not be suitable for all sensor related applications.

The Modular Environmental Monitoring (MEMOn) ontology [63] has been developed to represents various environmental monitoring data and to support semantic interoperability between heterogeneous data collected through a variety of observation techniques and systems. It defines a set of concepts and relationships for describing environmental monitoring equipment, including sensors, mainly borrowing concepts from SSN and SOSA. MEMOn is designed based on a modular architecture, making it easy to extend and reuse its components. The ontology covers various aspects of sensor data, such as sensor metadata, observations, and the context in which they occur and provides clear guidelines for mapping sensor data to the ontology. One of the main advantages of MEMOn is its domain-specificity, which ensures accurate representation of environmental monitoring data. However, MEMOn only supports limited coverage of non-environmental sensors and may be overly specific for some applications.

The Smart Appliances REFerence (SAREF) ontology [39] is a top-level ontology designed to describe smart appliances with their features and functions, including sensors and actuators. SAREF aims to provide a standardised framework for representing smart appliances and associated data and is designed in a modular fashion covering various aspects of smart appliances and systems. Compared to SSN and SOSA, SAREF has a narrower scope, focusing specifically on smart appliances rather than sensors and observations more broadly. However, SAREF can be used in conjunction with other ontologies, including SSN and SOSA, to provide a more comprehensive representation of smart systems. One strength of SAREF is its focus on consumer-facing applications, making it

particularly useful for the development of smart home systems and the Internet of Things. However, its limited scope may also be seen as a weakness, as it may not provide enough coverage for more specialised or complex sensor related applications outside of the smart appliance domain.

Several approaches have been proposed to ontologically describe environmental water resources and associated sensor readings; however, the focus has mainly been to either align the heterogeneous data from various sensor web resources (*e.g.*, [93, 94]) or support water quality monitoring (*e.g.*, [99]) with focus on data management and query capabilities. A hydrological sensor web ontology based on the SSN ontology has been developed to align semantics and support collaboration between individual water related sensor networks and data feeds [93], primarily in response to natural disasters such as floods. It introduces hydrological domain classes and establishes relevant reasoning rules. Further SSN-based domain ontologies describing information from sensors, observation values, and reading time series for distributed water sensor networks have been proposed [35, 62], partially even including data reliability assessment capabilities [35].

A semantic-enhanced modelling approach for river water quality monitoring and observation data processing has been proposed using the Observational Process Ontology (OPO) [94]. The ontology describes entities related to water resource management and associated observation data. While the SSN ontology contains more comprehensive concepts of sensor metadata, OPO provides more details about observational processes and models. Furthermore, OPO also defines entities related to water quality monitoring and pollution alerting to automatically trigger potential alerts.

A.4.2 Flood Ontologies

SWEET is a highly modular mid-level ontology suite for Earth system science and contains approximately 6000 concepts in 200 separate ontologies [18]. SWEET ontologies define a hierarchy of many flood risk related terms and are often referenced for a variety of environmental concepts. SWEET provides nine top-level concepts (*e.g.*, phenomena, and processes) which can be used as a foundation, and extended further to build domain-specific ontologies. Additionally, SWEET already provides several domain-specific ontologies (*e.g.*, for hydrosphere phenomena).

The Environmental Ontology (ENVO) is a FAIR-compliant domain ontology concerned with environments as encountered in ecological applications (*e.g.*, describing ecosystems, astronomical bodies, or environmental processes) [19]. It aims to promote interoperability of diverse datasets through the concise, controlled description of environment types across several levels of granularity. The *environmental hazard* subset contains suitable concepts to represent a flood and flooding in general, including more detailed descriptions of various kinds of flooding. Furthermore, it includes links to GEMET [41] as one of the general thesauri to describe core terminology for the environment as well as additional metadata from Wikidata [1]. Links to a variety of further machine-readable thesauri (*e.g.*, AGROVOC by the Food and Agriculture Organization of the United Nations) can be retrieved via GEMET. ENVO has been published in OWL format, which fosters its re-use to represent the phenomenological nature of flood events; however, no concepts to represent social or economic damage are considered.

The Modular Environmental Monitoring Ontology also aims to support semantic interoperability in the environmental monitoring domain [64]; however, most of the concepts related to floods are directly borrowed from ENVO, and only amended with further (non-semantic) annotations. A modular ontology-based framework for flood forecasting using a continuous stream of data about watersheds and sewer flow conditions has been developed by Agresta et al. [2]. A group of information-centric ontologies encompassing the flood domain are described in [88], including their potential to access, analyse, and visualise flood related data with natural language queries. The proposed artificial intelligence system facilitates the generation of knowledge that supports the communication of flood data and information. An ontology for riverine flood management has been developed by Wrachien et al. [97] to foster decision support strategies by generating a knowledge base for decision making.

The Ontology for River Flow and Flood Mitigation (ORFFM) [67] has been developed to resolve ambiguity during flood disaster management. The key objective of this ontology is to improve collaboration between multiple stakeholders and the integrated domains of irrigation, flood management, and administration through effective coordination and explicit semantic formalisation of common concepts. ORFFM estimates the impact and damage of a flood based on the affected instances (*e.g.*, areas, assets, roads, livestock, crops, *etc.*) and the assessment of likelihood and magnitude of a flood is based on the physiographic attributes and meteorological information. Affected areas are linked to administrative spatial regions.

Khantong et al. [56] have developed a domain ontology to enable seamless collaboration and information exchange between various stakeholders during a flood disaster response. The focus of this ontology is rather conceptualising the structure and sequence of relevant information, involved organisations and roles as well as the interplay and processes between them rather than conceptualising the data itself. To address both static and dynamic aspects of real world concepts involved in disaster response, one of the most prominent and influential upper ontologies, the Linguistic and Cognitive Engineering ontology (DOLCE), is used. DOLCE distinguishes entities into two main types, namely, `perdurants` to describe entities that happens in time and `endurants` referring to entities that exist in a timeless way.

Kollarits et al. [59] have developed the MONITOR risk management ontology as formalised knowledge base to describe the relations between natural, social and built environments, potentially hazardous events and several risk assessment and management terms. MONITOR builds on well established top-level ontologies, such as DOLCE, for formalised definitions of general terms, *i.e.*, reusing the concepts of `endurants` and `perdurants` to represent the (non-)transient nature of concepts. `Damage` is a subclass of `impact` and the central concept for all risk related propositions. `Risk` is defined as the probability of a damage of defined extent. An `event` is a `perdurant` (*i.e.*, occurrence, happening) which can cause an impact (*i.e.*, change) and has a magnitude (also referred to as intensity) and a spatio-temporal location, which can have a certain probability. `Hazard` is an event, which causes damage, and includes both an actual event and a potential event. `Damage potential` depends on the value of objects affected by a particular event and their vulnerability. `Hazard potential` describes the (potential) impact of a hazard and depends on the probability and the magnitude of the event.

Scheuer et al. [87] have proposed an ontology for a multi-criteria flood risk assessment domain, which builds and extends the conceptual model of risk as described by MONITOR. Furthermore, introduced concepts have been matched against their semantic counterparts in SWEET and other relevant ontologies to foster knowledge-based integration. A `flood` is a particular type of `event`, which in turn represents any potentially hazardous event. Any `event` and is characterised by its `intensity` and `recurrence interval`. Intensity is described as the combination of three different intensity parameters, namely inundation depth, duration and areal extent. To represent elements at risk of being affected by an event, *i.e.*, elements of natural, social or built environments, the concepts of `element at risk` is proposed. This concept encompasses infrastructure elements (*i.e.*, material and institutional infrastructure), population, and environmental components (*i.e.*, nature). The work states that material infrastructure and population are regarded the most relevant elements at risk by relevant flood risk assessment stakeholders. The `vulnerability` of an element at risk is assessed using susceptibility functions which relate given intensity parameters to an expected damage. To describe the hazard potential imposed by a particular flood event, `flood hazard maps` are introduced. Its subclasses `flood extend` and `flood depth` map link to visual representations of the areal extend and geospatial depth distribution, respectively.

A.4.3 Building Ontologies

The Building Topology Ontology (BOT) [82] is an upper ontology that aims to represent the core topological concepts of a building, including relationships between subcomponents contained within a building. BOT has been established by the W3C Linked Building Data Community Group to provide a minimal ontology for describing relationships between a building's sub-components and is designed as extensible baseline ontology to be used together with domain-specific schemas for more complex, specific use cases. BOT's scope is limited to buildings and their specific topologies. While BOT covers many high-level concepts of a building such as sites, floors, zones, and rooms, it does not include representations of properties or units of measurements related to systems, equipment, and devices, such as sensors or actuators.

The RealEstateCore (REC) ontology [83] builds upon a broader conceptualisation of buildings, including land and real estate classes. REC is specifically designed to facilitate building control as well as the development of integrated services within smart cities. It is funded and produced by a consortium of major real estate companies in Northern Europe. REC is a modular ontology based on data schemas describing concepts and relationships relevant to a variety of building systems and also captures the control aspects of technical systems, facility maintenance, certification, and financial aspects.

The Brick Schema [17] is an application ontology that standardises how physical, logical, and virtual assets in buildings are represented, as well as their relationships and associated telemetry, including sensors and actuators. It has been created through empirical analysis of concepts and relationships needed in real building applications and provides a formalised and extensible vocabulary for common building assets. The Brick Schema consists of three components: an RDF class hierarchy that describes the various building sub-systems and entities/equipment, a minimal and principled set of relationships for

connecting these entities into a directed graph, and a method of encapsulation that allows complex components to be composed from a set of lower-level ones. Its design is focused on abstracting and simplifying some aspects of common building equipment to make queries easier for users and integrate and interoperate with other linked data models, such as BOT and REC.

The Google Digital Building ontology [44] is an open-source project that aims to create a uniform schema and toolset for representing structured information about buildings and building-installed equipment. The ontology is inspired by and compatible with both Project Haystack [80] and Brick Schema [17], and is currently used by Google to manage its very large, heterogeneous building portfolio in a scalable way. The Haystack ontology is a lightweight, extensible ontology that is focused on the semantic modeling of various building systems with a simpler data model than the Brick Schema. The ontology enables applications and analyses to be easily transferable between buildings through a combination of semantically-expressive abstract modeling and an easy-to-use configuration language.

The iCity Building ontology [55] is a module of the larger Urban System Ontology. It provides a foundational vocabulary to represent building related data within the urban context, including physical building elements, systems, and their interactions. The Urban System Ontology extends this by including additional modules for representing other urban related domains, such as transportation, environment, and governance to enable interoperability between different urban systems and applications.

The Domain Analysis-Based Global Energy Ontology (DABGEO) has been proposed as global ontology for the energy domain that provides a common representation of relevant (sub-)domains captured by existing ontologies to foster interoperability across energy management applications and scenarios [28]. The ontology [27] follows a modular approach to provide a balance of abstraction (*i.e.*, to foster re-usability) and specification (*i.e.*, to foster direct usability) of energy domain data, with a focus on smart home and smart city energy management. The ontology contains a tailored sub-module for knowledge about infrastructure and buildings, containing classes, properties and axioms to represent static building features (*e.g.*, surface, material), geometrical details (rooms, floors) as well as internal and external environmental conditions (*e.g.*, room temperature). DABGEO also contains concepts to describe dynamic building behaviour, especially relevant to derive a holistic view of the energy consumption and generation performance of green buildings, self-sufficient in solar energy.

The BIMERR Building ontology [75] forms the core module for building topology and components data within the BIMERR Ontology Network [76], which connects several other domains related to the building industry and smart cities. The ontology extends the BOT ontology [82], which provides the vocabulary to describe the topology of a building as well as the relationships between its main components such as zones, spaces, and construction elements. Additionally, it reuses concepts from the SAREF4Building ontology [40] to represent components and systems that directly impact the energy consumption buildings and are susceptible to change in a renovation project.

A.5 Competency Questions

OntoEMS

- What properties are typical for a reporting station?
- What quantities can be reported by a station?
- Are the reported values actual observations or forecast values?
- How many reporting stations (for a particular quantity) are in a given area?
- What is the nearest measuring station (for a particular quantity) from a geolocation?
- What is the current reading for a particular quantity at a specific station?
- Are information about reporting stations consistent across different data providers?
- What water level stations are located along the same river?
- Which water level stations are currently indicating high readings?

OntoFlood

- What types of floods exist?
- How is the risk level of a flood defined?
- What can be affected by a flood?
- What is the areal extent of a given flood or flood warning?
- How many people/buildings are affected by a particular flood/flood warning?
- What is the total damage estimate of a particular flood?
- What water body causes a particular flood alert/warning?
- Which areas are forecast to be flooded with a particular likelihood?
- Is flood risk (*i.e.*, flood warning frequency and/or intensity) increasing in a particular area?
- What is the most common flood type (in a particular area)?

OntoBuiltEnv

- What types of properties exist?
- What are typical property usages?
- What are typical construction characteristics of a property?
- What is the geospatial footprint of a given building?
- Which buildings are in the vicinity of a given location?
- How many and which buildings are located in a given postcode?
- What is the average property price in a certain area?
- What is the most frequent property usage within a certain postcode?

A.6 Description Logic Representations of Ontologies

A.6.1 OntoEMS

The latest version of the ontology is publicly available as OWL file on github under: https://github.com/cambridge-cares/TheWorldAvatar/tree/main/JPS_Ontology/ontology/ontoems. A representation of the ontology in Description Logic is provided below:

Classes:

ReportingStation $\sqsubseteq$ geo:Feature
ReportingStation $\sqsubseteq \forall$ reports:om:Quantity
WaterLevelReportingStation $\sqsubseteq$ ReportingStation
WaterLevelReportingStation $\sqsubseteq \leq 1$ hasDownstreamStation.WaterLevelReportingStation
$\sqcap \geq 1$ hasDownstreamStation.WaterLevelReportingStation
om:Measure $\sqsubseteq \leq 1$ hasCurrentTrend.Trend $\sqcap \geq 1$ hasCurrentTrend.Trend
om:Measure $\sqsubseteq \leq 1$ hasCurrentRange.Range $\sqcap \geq 1$ hasCurrentRange.Range
om:Measure $\sqsubseteq \leq 1$ ts:hasTimeSeries.ts:TimeSeries $\sqcap$
≥ 1 ts:hasTimeSeries.ts:TimeSeries
om:Measure $\sqsubseteq \leq 1$ om:hasUnit.om:Unit $\sqcap \geq 1$ om:hasUnit.om:Unit
Forecast $\sqsubseteq \leq 1$ om:hasUnit.om:Unit $\sqcap \geq 1$ om:hasUnit.om:Unit
Forecast $\sqsubseteq \leq 1$ ts:hasTimeSeries.ts:TimeSeries $\sqcap$
≥ 1 ts:hasTimeSeries.ts:TimeSeries
AirTemperature $\sqsubseteq$ om:Temperature
AirTemperature $\equiv$ m3l:AirTemperature
AirTemperature $\equiv$ weather:AirTemperature
FeelsLikeTemperature $\sqsubseteq$ om:Temperature
DewPoint $\sqsubseteq$ om:Temperature
DewPoint $\equiv$ m3l:DewPoint
DewPoint $\equiv$ weather:DewPointTemperature
AtmosphericPressure $\sqsubseteq$ om:Pressure
AtmosphericPressure $\equiv$ m3l:AtmosphericPressure
AtmosphericPressure $\equiv$ weather:AtmosphericPressure
RelativeHumidity $\equiv$ om:RelativeHumidity
RelativeHumidity $\equiv$ m3l:RelativeHumidity
RelativeHumidity $\equiv$ weather:Humidity
Rainfall $\sqsubseteq$ om:Height
Rainfall $\sqsubseteq$ m3l:Rainfall
Rainfall $\sqsubseteq$ weather:Precipitation
SnowFall $\sqsubseteq$ om:Height
SnowFall $\sqsubseteq$ weather:Precipitation
PrecipitationProbability $\sqsubseteq$ om:Number
PrecipitationProbability $\sqsubseteq$ sio:SIO_000638
CloudCover $\sqsubseteq$ om:Number
CloudCover $\equiv$ m3l:CloudCover

UVIndex ⊑ om:Number
Visibility ⊑ om:Distance
WindSpeed ⊑ om:Speed
WindSpeed ≡ m3l:WindSpeed
WindGust ⊑ om:Speed
WindGust ⊑ m3l:WindSpeed
WindDirection ⊑ om:Angle
WindDirection ≡ m3l:WindDirection
AirPollutantConcentration ⊑ ontouom:Concentration
om:VolumeFraction ⊑ ontouom:Concentration
ontouom:MassOfSubstanceConcentration ⊑ ontouom:Concentration
CarbonMonoxideConcentration ⊑ AirPollutantConcentration
CarbonDioxideConcentration ⊑ AirPollutantConcentration
OzoneConcentration ⊑ AirPollutantConcentration
SulfurDioxideConcentration ⊑ AirPollutantConcentration
NitrogenOxidesConcentration ⊑ AirPollutantConcentration
NitrogenDioxideConcentration ⊑ NitrogenOxidesConcentration
NitrogenMonoxideConcentration ⊑ NitrogenOxidesConcentration
ParticulateMatterConcentration ⊑ AirPollutantConcentration
PM10Concentration ⊑ ParticulateMatterConcentration
PM2.5Concentration ⊑ ParticulateMatterConcentration
GlobalHorizontalIrradiance ⊑ om:Irradiance
GlobalHorizontalIrradiance ⊑ weather:SolarIrradiance
GlobalHorizontalIrradiance ⊑ m3l:SolarRadiation
DiffuseHorizontalIrradiance ⊑ om:Irradiance
DiffuseHorizontalIrradiance ⊑ weather:SolarIrradiance
DiffuseHorizontalIrradiance ⊑ m3l:SolarRadiation
DirectNormalIrradiance ⊑ om:Irradiance
DirectNormalIrradiance ⊑ weather:SolarIrradiance
DirectNormalIrradiance ⊑ m3l:SolarRadiation
WaterLevel ⊑ om:Height
WaterLevel ⊑ m3l:WaterLevel
WaterFlow ⊑ om:VolumetricFlowRate
Falling ⊑ Trend
Steady ⊑ Trend
Rising ⊑ Trend
UnavailableTrend ⊑ Trend
LowRange ⊑ Range
NormalRange ⊑ Range
HighRange ⊑ Range
UnavailableRange ⊑ Range

Object properties:

$\exists$ rt:StageScale.owl:Thing $\sqsubseteq$ WaterLevelReportingStation
$\top \sqsubseteq \forall$ rt:StageScale.owl:Thing
$\exists$ hasCurrentRange.owl:Thing $\sqsubseteq$ om:Measure
$\top \sqsubseteq \forall$ hasCurrentRange.Range
$\exists$ hasCurrentTrend.owl:Thing $\sqsubseteq$ om:Measure
$\top \sqsubseteq \forall$ hasCurrentTrend.Trend
$\exists$ hasDownstreamStation.owl:Thing $\sqsubseteq$ WaterLevelReportingStation
$\top \sqsubseteq \forall$ hasDownstreamStation.WaterLevelReportingStation
$\exists$ hasForecastedValue.owl:Thing $\sqsubseteq$ om:Quantity
$\top \sqsubseteq \forall$ hasForecastedValue.Forecast
$\exists$ ts:hasTimeSeries.owl:Thing $\sqsubseteq$ om:Measure $\sqcup$ Forecast
$\top \sqsubseteq \forall$ ts:hasTimeSeries.ts:TimeSeries
$\exists$ om:hasUnit.owl:Thing $\sqsubseteq$ Forecast
$\exists$ om:hasUnit.owl:Thing $\sqsubseteq$ om:Measure
$\top \sqsubseteq \forall$ om:hasUnit.om:Unit
$\exists$ om:hasValue.owl:Thing $\sqsubseteq$ om:Quantity
$\top \sqsubseteq \forall$ om:hasValue.om:Measure
$\exists$ includedIn.owl:Thing $\sqsubseteq$ DiffuseHorizontalIrradiance $\sqcup$ DirectNormalIrradiance
$\top \sqsubseteq \forall$ includedIn.GlobalHorizontalIrradiance
$\exists$ measureOf.owl:Thing $\sqsubseteq$ AirPollutantConcentration
$\top \sqsubseteq \forall$ measureOf (m3l:AirPollution $\sqcup$ weather:AirPollution)
$\exists$ reports:owl:Thing $\sqsubseteq$ ReportingStation
$\top \sqsubseteq \forall$ reports:om:Quantity

Data properties:

$\exists$ rt:catchmentName.rdfs:Literal $\sqsubseteq$ WaterLevelReportingStation
$\top \sqsubseteq \forall$ rt:catchmentName.xsd:string
$\exists$ createdOn.rdfs:Literal $\sqsubseteq$ Forecast
$\top \sqsubseteq \forall$ createdOn.xsd:dateTime
$\exists$ dataSource.rdfs:Literal $\sqsubseteq$ ReportingStation
$\top \sqsubseteq \forall$ dataSource.xsd:string
$\exists$ hasIdentifier.rdfs:Literal $\sqsubseteq$ ReportingStation
$\top \sqsubseteq \forall$ hasIdentifier.xsd:string
$\exists$ hasObservationElevation.rdfs:Literal $\sqsubseteq$ ReportingStation
$\top \sqsubseteq \forall$ hasObservationElevation.xsd:float
$\exists$ ts:hasRDB.rdfs:Literal $\sqsubseteq$ ts:TimeSeries
$\top \sqsubseteq \forall$ ts:hasRDB.xsd:string
$\exists$ ts:hasTimeUnit.rdfs:Literal $\sqsubseteq$ ts:TimeSeries
$\top \sqsubseteq \forall$ ts:hasTimeUnit.xsd:string
$\exists$ rdfs.label.rdfs:Literal $\sqsubseteq$ AirPollutantConcentration $\sqcup$ ReportingStation
$\top \sqsubseteq \forall$ rdfs.label.xsd:string
$\exists$ rt:period.rdfs:Literal $\sqsubseteq$ om:Measure

$\top \sqsubseteq \forall$ rt:period.xsd:string
$\exists$ rt:qualifier.rdfs:Literal $\sqsubseteq$ om:Measure
$\top \sqsubseteq \forall$ rt:qualifier.xsd:string
$\exists$ rt:riverName.rdfs:Literal $\sqsubseteq$ WaterLevelReportingStation
$\top \sqsubseteq \forall$ rt:riverName.xsd:string
$\exists$ om:symbol.rdfs:Literal $\sqsubseteq$ om:Unit
$\top \sqsubseteq \forall$ om:symbol.xsd:string
$\exists$ rt:typicalRangeHigh.rdfs:Literal $\sqsubseteq$ owl:Thing
$\top \sqsubseteq \forall$ rt:typicalRangeHigh.xsd:float
$\exists$ rt:typicalRangeLow.rdfs:Literal $\sqsubseteq$ owl:Thing
$\top \sqsubseteq \forall$ rt:typicalRangeLow.xsd:float
$\exists$ rt:valueType.rdfs:Literal $\sqsubseteq$ om:Measure
$\top \sqsubseteq \forall$ rt:valueType.xsd:string

A.6.2 OntoFlood

The latest version of the ontology is publicly available as OWL file on github under: https://github.com/cambridge-cares/TheWorldAvatar/tree/main/JPS_Ontology/ontology/ontoflood. A representation of the ontology in Description Logic is provided below:

Classes:

envo:ENVO_01000710 $\sqsubseteq$ soph:Event
envo:ENVO_01000710 $\equiv$ sophhy:Flood
envo:ENVO_01000711 $\sqsubseteq$ envo:ENVO_01000710
envo:ENVO_01000712 $\sqsubseteq$ envo:ENVO_01000710
envo:ENVO_01000713 $\sqsubseteq$ envo:ENVO_01000710
soph:Event $\sqsubseteq \leq 1$ hasLocation.Location $\sqcap \geq 1$ hasLocation.Location
soph:Event $\sqsubseteq \leq 1$ hasTimeInterval.time:Interval $\sqcap$
≥ 1 hasTimeInterval.time:Interval
Impact $\sqsubseteq \leq 1$ hasMonetaryValue.om:AmountOfMoney $\sqcap$
≥ 1 hasMonetaryValue.om:AmountOfMoney
ArealExtentPolygon $\sqsubseteq$ geo:Feature
FloodDepth $\sqsubseteq$ om:Depth
WaterVelocity $\sqsubseteq$ om:Velocity
om:Measure $\sqsubseteq \leq 1$ hasGeospatialDistribution.ArealDistribution $\sqcap$
≥ 1 hasGeospatialDistribution.ArealDistribution
Building $\sqsubseteq$ InfrastructureComponent
ArealInfrastructure $\sqsubseteq$ InfrastructureComponent
Vehicle $\sqsubseteq$ InfrastructureComponent
NetworkInfrastructure $\sqsubseteq$ InfrastructureComponent
IndustrialArea $\sqsubseteq$ ArealInfrastructure
ResidentialArea $\sqsubseteq$ ArealInfrastructure

CommercialArea $\sqsubseteq$ ArealInfrastructure
MobilityNetwork $\sqsubseteq$ NetworkInfrastructure
ProvisioningNetwork $\sqsubseteq$ NetworkInfrastructure
envo:ENVO_01000707 $\sqsubseteq$ envo:ENVO_02500002
envo:ENVO_01000708 $\sqsubseteq$ envo:ENVO_02500002
envo:ENVO_01000709 $\sqsubseteq$ envo:ENVO_02500002
envo:ENVO_01000717 $\sqsubseteq$ envo:ENVO_02500002
envo:ENVO_01000718 $\sqsubseteq$ envo:ENVO_02500002
rt:FloodAlertOrWarning $\sqsubseteq \leq 1$ warnsAbout.envo:ENVO_01000710 $\sqcap$
≥ 1 warnsAbout.envo:ENVO_01000710
rt:FloodAlertOrWarning $\sqsubseteq \leq 1$ hasSeverity.Severity $\sqcap \geq 1$ hasSeverity.Severity
SevereFloodWarning $\sqsubseteq$ Severity
FloodWarning $\sqsubseteq$ Severity
FloodAlert $\sqsubseteq$ Severity
WarningNoLongerInForce $\sqsubseteq$ Severity
rt:FloodArea $\equiv$ rt:FloodAlertArea
rt:FloodArea $\equiv$ rt:FloodWarningArea
rt:FloodArea $\sqsubseteq \leq 1$ hasAlertOrWarningHistory.FloodAlertOrWarningHistory $\sqcap$
≥ 1 hasAlertOrWarningHistory.FloodAlertOrWarningHistory
FloodAlertOrWarningHistory $\sqsubseteq \leq 1$ ts:hasTimeSeries.ts:TimeSeries $\sqcap$
≥ 1 ts:hasTimeSeries.ts:TimeSeries
envo:ENVO_00000014 $\sqsubseteq$ envo:ENVO_00000063
envo:ENVO_00000016 $\sqsubseteq$ envo:ENVO_00000063
envo:ENVO_00000020 $\sqsubseteq$ envo:ENVO_00000063
envo:ENVO_00000022 $\sqsubseteq$ envo:ENVO_00000063
rt:FloodArea $\sqsubseteq \leq 1$ hasLocation.Location $\sqcap \geq 1$ hasLocation.Location
FloodForecast $\sqsubseteq \leq 1$ predicts.envo:ENVO_01000710 $\sqcap$
≥ 1 predicts.envo:ENVO_01000710
FloodForecast $\sqsubseteq \leq 1$ hasLocation.Location $\sqcap \geq 1$ hasLocation.Location
FloodForecast $\sqsubseteq \leq 1$ hasRiskLevel.RiskLevel $\sqcap \geq 1$ hasRiskLevel.RiskLevel
RiskLevel $\sqsubseteq \leq 1$ hasPotentialImpact.PotentialImpact $\sqcap$
≥ 1 hasPotentialImpact.PotentialImpact
RiskLevel $\sqsubseteq \leq 1$ hasLikelihood.Likelihood $\sqcap \geq 1$ hasLikelihood.Likelihood
SevereImpact $\sqsubseteq$ PotentialImpact
SignificantImpact $\sqsubseteq$ PotentialImpact
MinorImpact $\sqsubseteq$ PotentialImpact
MinimalImpact $\sqsubseteq$ PotentialImpact
HighLikelihood $\sqsubseteq$ Likelihood
MediumLikelihood $\sqsubseteq$ Likelihood
LowLikelihood $\sqsubseteq$ Likelihood
VeryLowLikelihood $\sqsubseteq$ Likelihood
GroundWater $\sqsubseteq$ FloodSource
RiverWater $\sqsubseteq$ FloodSource
SurfaceWater $\sqsubseteq$ FloodSource
CoastalWater $\sqsubseteq$ FloodSource

Object properties:

$\exists$ envo:RO_0002354.owl:Thing $\sqsubseteq$ envo:ENVO_01000710
$\top \sqsubseteq \forall$ envo:RO_0002354.envo:ENVO_02500002
$\exists$ affects.owl:Thing $\sqsubseteq$ envo:ENVO_01000710
$\top \sqsubseteq \forall$ affects (EnvironmentalComponent $\sqcup$ InfrastructureComponent $\sqcup$ Population)
$\exists$ attachedWaterBody.owl:Thing $\sqsubseteq$ rt:FloodArea
$\top \sqsubseteq \forall$ attachedWaterBody.envo:ENVO_00000063
$\exists$ rt:currentWarning.owl:Thing $\sqsubseteq$ rt:FloodArea
$\top \sqsubseteq \forall$ rt:currentWarning.FloodAlertOrWarning
$\exists$ hasAdministrativeDistrict.owl:Thing $\sqsubseteq$ Location
$\top \sqsubseteq \forall$ hasAdministrativeDistrict.AdministrativeDistrict
$\exists$ hasAlertOrWarningHistory.owl:Thing $\sqsubseteq$ rt:FloodArea
$\top \sqsubseteq \forall$ hasAlertOrWarningHistory.FloodAlertOrWarningHistory
$\exists$ hasArealExtent.owl:Thing $\sqsubseteq$ Location
$\top \sqsubseteq \forall$ hasArealExtent.ArealExtentPolygon
$\exists$ time:hasBeginning.owl:Thing $\sqsubseteq$ Interval
$\top \sqsubseteq \forall$ time:hasBeginning.Instant
$\exists$ time:hasEnd.owl:Thing $\sqsubseteq$ Interval
$\top \sqsubseteq \forall$ time:hasEnd.Instant
$\exists$ hasFloodSource.owl:Thing $\sqsubseteq$ FloodForecast
$\top \sqsubseteq \forall$ hasFloodSource.FloodSource
$\exists$ hasGeospatialDistribution.owl:Thing $\sqsubseteq$ Measure
$\top \sqsubseteq \forall$ hasGeospatialDistribution.ArealDistribution
$\exists$ hasIntensity.owl:Thing $\sqsubseteq$ envo:ENVO_01000710
$\top \sqsubseteq \forall$ hasIntensity.Quantity
$\exists$ hasLikelihood.owl:Thing $\sqsubseteq$ RiskLevel
$\top \sqsubseteq \forall$ hasLikelihood.Likelihood
$\exists$ hasLocation.owl:Thing $\sqsubseteq$ rt:FloodArea $\sqcup$ soph:Event $\sqcup$ FloodForecast
$\top \sqsubseteq \forall$ hasLocation.Location
$\exists$ hasMonetaryValue.owl:Thing $\sqsubseteq$ Impact
$\top \sqsubseteq \forall$ hasMonetaryValue.AmountOfMoney
$\exists$ hasPotentialImpact.owl:Thing $\sqsubseteq$ RiskLevel
$\top \sqsubseteq \forall$ hasPotentialImpact.PotentialImpact
$\exists$ hasRiskLevel.owl:Thing $\sqsubseteq$ FloodForecast
$\top \sqsubseteq \forall$ hasRiskLevel.RiskLevel
$\exists$ hasSeverity.owl:Thing $\sqsubseteq$ FloodAlertOrWarning
$\top \sqsubseteq \forall$ hasSeverity.Severity
$\exists$ hasTimeInterval.owl:Thing $\sqsubseteq$ Event
$\top \sqsubseteq \forall$ hasTimeInterval.Interval
$\exists$ ts:hasTimeSeries.owl:Thing $\sqsubseteq$ FloodAlertOrWarningHistory
$\top \sqsubseteq \forall$ ts:hasTimeSeries.TimeSeries
$\exists$ hasTotalAffectedArea.owl:Thing $\sqsubseteq$ InfrastructureComponent
$\top \sqsubseteq \forall$ hasTotalAffectedArea.Area
$\exists$ hasTotalMonetaryValue.owl:Thing $\sqsubseteq$ InfrastructureComponent
$\top \sqsubseteq \forall$ hasTotalMonetaryValue.AmountOfMoney

$\exists$ om:hasUnit.owl:Thing $\sqsubseteq$ Measure
$\top \sqsubseteq \forall$ om:hasUnit.Unit
$\exists$ om:hasValue.owl:Thing $\sqsubseteq$ om:AmountOfMoney $\sqcup$ Area
$\exists$ om:hasValue.owl:Thing $\sqsubseteq$ Quantity
$\top \sqsubseteq \forall$ om:hasValue.Measure
$\exists$ predicts.owl:Thing $\sqsubseteq$ FloodForecast
$\top \sqsubseteq \forall$ predicts.envo:ENVO_01000710
$\exists$ resultsIn.owl:Thing $\sqsubseteq$ Event
$\top \sqsubseteq \forall$ resultsIn.Impact
$\exists$ warnsAbout.owl:Thing $\sqsubseteq$ FloodAlertOrWarning
$\top \sqsubseteq \forall$ warnsAbout.envo:ENVO_01000710

Data properties:

$\exists$ hasAreaIdentifier.rdfs:Literal $\sqsubseteq$ rt:FloodArea
$\top \sqsubseteq \forall$ hasAreaIdentifier.xsd:string
$\exists$ hasClassification.rdfs:Literal $\sqsubseteq$ Impact
$\top \sqsubseteq \forall$ hasClassification.xsd:string
$\exists$ hasEffectiveDate.rdfs:Literal $\sqsubseteq$ FloodForecast
$\top \sqsubseteq \forall$ hasEffectiveDate.xsd:date
$\exists$ hasImpactLevel.rdfs:Literal $\sqsubseteq$ PotentialImpact
$\top \sqsubseteq \forall$ hasImpactLevel.xsd:integer
$\exists$ hasLikelihoodScore.rdfs:Literal $\sqsubseteq$ Likelihood
$\top \sqsubseteq \forall$ hasLikelihoodScore.xsd:integer
$\exists$ om:hasNumericalValue.rdfs:Literal $\sqsubseteq$ Measure
$\top \sqsubseteq \forall$ om:hasNumericalValue.xsd:float
$\exists$ ts:hasRDB.rdfs:Literal $\sqsubseteq$ TimeSeries
$\top \sqsubseteq \forall$ ts:hasRDB.xsd:string
$\exists$ hasSeverityLevel.rdfs:Literal $\sqsubseteq$ Severity
$\top \sqsubseteq \forall$ hasSeverityLevel.xsd:integer
$\exists$ ts:hasTimeUnit.rdfs:Literal $\sqsubseteq$ TimeSeries
$\top \sqsubseteq \forall$ ts:hasTimeUnit.xsd:string
$\exists$ hasTotalCount.rdfs:Literal $\sqsubseteq$ InfrastructureComponent $\sqcup$ Population
$\top \sqsubseteq \forall$ hasTotalCount.xsd:integer
$\exists$ hasWGS84LatitudeLongitude.rdfs:Literal $\sqsubseteq$ Location
$\top \sqsubseteq \forall$ hasWGS84LatitudeLongitude.geolit:lat-lon
$\exists$ time:hasXSDDuration.rdfs:Literal $\sqsubseteq$ Interval
$\top \sqsubseteq \forall$ time:hasXSDDuration.xsd:string
$\exists$ time:inXSDDateTimeStamp.rdfs:Literal $\sqsubseteq$ Instant
$\top \sqsubseteq \forall$ time:inXSDDateTimeStamp.xsd:string
$\exists$ rdfs:label.rdfs:Literal $\sqsubseteq$ Severity
$\exists$ rdfs:label.rdfs:Literal $\sqsubseteq$ Likelihood
$\exists$ rdfs:label.rdfs:Literal $\sqsubseteq$ PotentialImpact
$\exists$ rdfs:label.rdfs:Literal $\sqsubseteq$ rt:FloodArea
$\exists$ rdfs:label.rdfs:Literal $\sqsubseteq$ envo:ENVO_00000063

$\top \sqsubseteq \forall$ rdfs:label.xsd:string
$\exists$ rt:message.rdfs:Literal $\sqsubseteq$ FloodAlertOrWarning
$\top \sqsubseteq \forall$ rt:message.xsd:string
$\exists$ om:symbol.rdfs:Literal $\sqsubseteq$ Unit
$\top \sqsubseteq \forall$ om:symbol.xsd:string
$\exists$ rt:timeMessageChanged.rdfs:Literal $\sqsubseteq$ FloodAlertOrWarning
$\top \sqsubseteq \forall$ rt:timeMessageChanged.xsd:dateTime
$\exists$ rt:timeRaised.rdfs:Literal $\sqsubseteq$ FloodAlertOrWarning
$\top \sqsubseteq \forall$ rt:timeRaised.xsd:dateTime
$\exists$ rt:timeSeverityChanged.rdfs:Literal $\sqsubseteq$ FloodAlertOrWarning
$\top \sqsubseteq \forall$ rt:timeSeverityChanged.xsd:dateTime

A.6.3 OntoBuiltEnv

The latest version of the ontology is publicly available as OWL file on github under: https://github.com/cambridge-cares/TheWorldAvatar/blob/main/JPS_Ontology/ontology/ontobuiltenv/OntoBuiltEnv.owl. A representation of the ontology in Description Logic is provided below:

Classes:

Property $\sqsubseteq \top$
Flat $\sqsubseteq$ Property
dabgeo:Building $\sqsubseteq$ Property
dabgeo:Building $\sqsubseteq$ geo:Feature
dabgeo:Building $\equiv$ bot:Building
dabgeo:Building $\equiv$ bimerr:Building
dabgeo:Building $\equiv$ db:Building
dabgeo:Building $\equiv$ icity:Building
dabgeo:Building $\equiv$ ifc:IfcBuilding
dabgeo:Building $\sqsubseteq \leq 1$ hasInstalledPVArea.om:Area $\sqcap$
≥ 1 hasInstalledPVArea.om:Area
dabgeo:Building $\sqsubseteq \leq 1$ hasGroundElevation.om:Height $\sqcap$
≥ 1 hasGroundElevation.om:Height
dabgeo:Building $\sqsubseteq \leq 1$ hasTotalRoofArea.om:Area $\sqcap$
≥ 1 hasTotalRoofArea.om:Area
dabgeo:Building $\sqsubseteq \leq 1$ hasOntoCityGMLRepresentation.owl:Thing $\sqcap$
≥ 1 hasOntoCityGMLRepresentation.owl:Thing
dabgeo:Building $\sqsubseteq \leq 1$ hasPVsuitableRoofArea.om:Area $\sqcap$
≥ 1 hasPVsuitableRoofArea.om:Area
Property $\sqsubseteq \leq 1$ hasAddress.icontact:Address $\sqcap \geq 1$ hasAddress.icontact:Address
Property $\sqsubseteq \leq 1$ hasMarketValue.om:AmountOfMoney $\sqcap$
≥ 1 hasMarketValue.om:AmountOfMoney
Property $\sqsubseteq \leq 1$ hasPropertyType.PropertyType $\sqcap$

≥ 1 hasPropertyType.PropertyType
Property $\sqsubseteq \leq 1$ hasTotalFloorArea.om:Area $\sqcap \geq 1$ hasTotalFloorArea.om:Area
Property $\sqsubseteq \leq 1$ hasBuiltForm.BuiltForm $\sqcap \geq 1$ hasBuiltForm.BuiltForm
Property $\sqsubseteq \leq 1$ hasLatestTransactionRecord.lrppi:TransactionRecord $\sqcap$
≥ 1 hasLatestTransactionRecord.lrppi:TransactionRecord
icontact:Address $\sqsubseteq \leq 1$ hasPostalCode.PostalCode $\sqcap$
≥ 1 hasPostalCode.PostalCode
Domestic $\sqsubseteq$ PropertyUsage
Non-Domestic $\sqsubseteq$ PropertyUsage
SingleResidential $\sqsubseteq$ Domestic
MultiResidential $\sqsubseteq$ Domestic
EmergencyService $\sqsubseteq$ Non-Domestic
FireStation $\sqsubseteq$ EmergencyService
PoliceStation $\sqsubseteq$ EmergencyService
Education $\sqsubseteq$ Non-Domestic
School $\sqsubseteq$ Education
University $\sqsubseteq$ Education
MedicalCare $\sqsubseteq$ Non-Domestic
Clinic $\sqsubseteq$ MedicalCare
Hospital $\sqsubseteq$ MedicalCare
CulturalFacility $\sqsubseteq$ Non-Domestic
DrinkingEstablishment $\sqsubseteq$ Non-Domestic
EatingEstablishment $\sqsubseteq$ Non-Domestic
Hotel $\sqsubseteq$ Non-Domestic
IndustrialFacility $\sqsubseteq$ Non-Domestic
Office $\sqsubseteq$ Non-Domestic
ReligiousFacility $\sqsubseteq$ Non-Domestic
RetailEstablishment $\sqsubseteq$ Non-Domestic
SportsFacility $\sqsubseteq$ Non-Domestic
Bungalow $\sqsubseteq$ PropertyType
House $\sqsubseteq$ PropertyType
Maisonette $\sqsubseteq$ PropertyType
ParkHome $\sqsubseteq$ PropertyType
TransportFacility $\sqsubseteq$ Non-Domestic
Detached $\sqsubseteq$ BuiltForm
Semi-Detached $\sqsubseteq$ BuiltForm
Terraced $\sqsubseteq$ BuiltForm
Floor $\sqsubseteq$ ConstructionComponent
Roof $\sqsubseteq$ ConstructionComponent
Wall $\sqsubseteq$ ConstructionComponent
Window $\sqsubseteq$ ConstructionComponent
AveragePricePerSqm $\sqsubseteq$ ontouom:AmountOfMoneyPerArea
AveragePricePerSqm $\sqsubseteq \leq 1$ representativeFor.PostalCode $\sqcap$
≥ 1 representativeFor.PostalCode
PropertyPriceIndex $\sqsubseteq \leq 1$ ts:hasTimeSeries.ts:TimeSeries $\sqcap$
≥ 1 ts:hasTimeSeries.ts:TimeSeries

ontouom:pound_sterling_per_sqm ⊑ om:Unit

Object properties:

∃ hasAddress.owl:Thing ⊑ Property
⊤ ⊑ ∀ hasAddress.icontact:Address
∃ hasAdministrativeDistrict.owl:Thing ⊑ icontact:Address
⊤ ⊑ ∀ hasAdministrativeDistrict.AdministrativeDistrict
∃ time:hasBeginning.owl:Thing ⊑ time:Interval
⊤ ⊑ ∀ time:hasBeginning.time:Instant
∃ hasBuiltForm.owl:Thing ⊑ Property
⊤ ⊑ ∀ hasBuiltForm.BuiltForm
∃ hasConstructionComponent.owl:Thing ⊑ Property
⊤ ⊑ ∀ hasConstructionComponent.ConstructionComponent
∃ hasConstructionDate.owl:Thing ⊑ Property
⊤ ⊑ ∀ hasConstructionDate.time:Interval
∃ time:hasEnd.owl:Thing ⊑ time:Interval
⊤ ⊑ ∀ time:hasEnd.time:Instant
∃ hasGroundElevation.owl:Thing ⊑ dabgeo:Building
⊤ ⊑ ∀ hasGroundElevation.om:Height
∃ hasInstalledPVArea.owl:Thing ⊑ dabgeo:Building
⊤ ⊑ ∀ hasInstalledPVArea.om:Area
∃ hasLatestTransactionRecord.owl:Thing ⊑ Property
⊤ ⊑ ∀ hasLatestTransactionRecord.lrppi:TransactionRecord
∃ hasMarketValue.owl:Thing ⊑ Property
⊤ ⊑ ∀ hasMarketValue.om:AmountOfMoney
∃ hasOntoCityGMLRepresentation.owl:Thing ⊑ dabgeo:Building
⊤ ⊑ ∀ hasOntoCityGMLRepresentation.owl:Thing
∃ hasPVsuitableRoofArea.owl:Thing ⊑ dabgeo:Building
⊤ ⊑ ∀ hasPVsuitableRoofArea.om:Area
∃ hasPostalCode.owl:Thing ⊑ icontact:Address
⊤ ⊑ ∀ hasPostalCode.PostalCode
∃ hasPropertyType.owl:Thing ⊑ Property
⊤ ⊑ ∀ hasPropertyType.PropertyType
∃ hasPropertyUsage.owl:Thing ⊑ Property
⊤ ⊑ ∀ hasPropertyUsage.PropertyUsage
∃ ts:hasTimeSeries.owl:Thing ⊑ PropertyPriceIndex
⊤ ⊑ ∀ ts:hasTimeSeries.ts:TimeSeries
∃ hasTotalFloorArea.owl:Thing ⊑ Property
⊤ ⊑ ∀ hasTotalFloorArea.om:Area
∃ hasTotalRoofArea.owl:Thing ⊑ dabgeo:Building
⊤ ⊑ ∀ hasTotalRoofArea.om:Area
∃ om:hasUnit.owl:Thing ⊑ om:Measure
⊤ ⊑ ∀ om:hasUnit.om:Unit
∃ om:hasValue.owl:Thing ⊑ om:AmountOfMoney ⊔ om:Area ⊔ om:Height ⊔

AveragePricePerSqm
$\top \sqsubseteq \forall$ om:hasValue.om:Measure
$\exists$ isIn.owl:Thing $\sqsubseteq$ Flat
$\top \sqsubseteq \forall$ isIn.dabgeo:Building
$\exists$ isPresumedMatchOf.owl:Thing $\sqsubseteq$ icontact:Address
$\top \sqsubseteq \forall$ isPresumedMatchOf.owl:Thing
$\exists$ locatedIn.owl:Thing $\sqsubseteq$ Property
$\top \sqsubseteq \forall$ locatedIn.AdministrativeDistrict
$\exists$ representativeFor.owl:Thing $\sqsubseteq$ PropertyPriceIndex
$\top \sqsubseteq \forall$ representativeFor.PostalCode
$\exists$ representativeFor.owl:Thing $\sqsubseteq$ AveragePricePerSqm
$\top \sqsubseteq \forall$ representativeFor.AdministrativeDistrict

Data properties:

$\exists$ icontact:hasBuilding.rdfs:Literal $\sqsubseteq$ icontact:Address
$\top \sqsubseteq \forall$ icontact:hasBuilding.xsd:string
$\exists$ hasEnergyRating.rdfs:Literal $\sqsubseteq$ Property
$\top \sqsubseteq \forall$ hasEnergyRating.xsd:string
$\exists$ hasIdentifier.rdfs:Literal $\sqsubseteq$ Property
$\top \sqsubseteq \forall$ hasIdentifier.xsd:string
$\exists$ hasLatestEPC.rdfs:Literal $\sqsubseteq$ Property
$\top \sqsubseteq \forall$ hasLatestEPC.xsd:string
$\exists$ hasNumberOfHabitableRooms.rdfs:Literal $\sqsubseteq$ Property
$\top \sqsubseteq \forall$ hasNumberOfHabitableRooms.xsd:integer
$\exists$ om:hasNumericalValue.rdfs:Literal $\sqsubseteq$ om:Measure
$\top \sqsubseteq \forall$ om:hasNumericalValue.xsd:float
$\exists$ ts:hasRDB.rdfs:Literal $\sqsubseteq$ ts:TimeSeries
$\top \sqsubseteq \forall$ ts:hasRDB.xsd:string
$\exists$ icontact:hasStreet.rdfs:Literal $\sqsubseteq$ icontact:Address
$\top \sqsubseteq \forall$ icontact:hasStreet.xsd:string
$\exists$ icontact:hasStreetNumber.rdfs:Literal $\sqsubseteq$ icontact:Address
$\top \sqsubseteq \forall$ icontact:hasStreetNumber.xsd:string
$\exists$ ts:hasTimeUnit.rdfs:Literal $\sqsubseteq$ ts:TimeSeries
$\top \sqsubseteq \forall$ ts:hasTimeUnit.xsd:string
$\exists$ om:hasUnitName.rdfs:Literal $\sqsubseteq$ icontact:Address
$\top \sqsubseteq \forall$ om:hasUnitName.xsd:string
$\exists$ hasUsageShare.rdfs:Literal $\sqsubseteq$ PropertyUsage
$\top \sqsubseteq \forall$ hasUsageShare.xsd:float
$\exists$ hasWGS84LatitudeLongitude.rdfs:Literal $\sqsubseteq$ Property
$\top \sqsubseteq \forall$ hasWGS84LatitudeLongitude.geolit:lat-lon
$\exists$ time:inXSDDateTimeStamp.rdfs:Literal $\sqsubseteq$ time:Instant
$\top \sqsubseteq \forall$ time:inXSDDateTimeStamp.xsd:string
$\exists$ rdfs:label.rdfs:Literal $\sqsubseteq$ AdministrativeDistrict
$\exists$ rdfs:label.rdfs:Literal $\sqsubseteq$ PropertyUsage

$\exists$ rdfs:label.rdfs:Literal $\sqsubseteq$ PostalCode
$\top \sqsubseteq \forall$ rdfs:label.xsd:string
$\exists$ lrppi:pricePaid.rdfs:Literal $\sqsubseteq$ lrppi:TransactionRecord
$\top \sqsubseteq \forall$ lrppi:pricePaid.xsd:integer
$\exists$ om:symbol.rdfs:Literal $\sqsubseteq$ om:Unit
$\top \sqsubseteq \forall$ om:symbol.xsd:string
$\exists$ lrppi:transactionDate.rdfs:Literal $\sqsubseteq$ lrppi:TransactionRecord
$\top \sqsubseteq \forall$ lrppi:transactionDate.xsd:date

References

[1] Wikidata, 2019. Available at https://www.wikidata.org/wiki/Wikidata:Main_Page (accessed Feb 2023).

[2] A. Agresta, G. Fattoruso, F. Pasanisi, C. Tebano, S. De Vito, and G. Di Francia. An Ontology Framework for Flooding Forecasting. In B. Murgante, S. Misra, A. M. A. C. Rocha, C. Torre, J. G. Rocha, M. I. Falcão, D. Taniar, B. O. Apduhan, and O. Gervasi, editors, *Computational Science and Its Applications – ICCSA 2014*, pages 417–428. Springer International Publishing, Cham, 2014. doi:10.1007/978-3-319-09147-1_30.

[3] J. Akroyd, S. Mosbach, A. Bhave, and M. Kraft. Universal Digital Twin - A Dynamic Knowledge Graph. *Data-Centric Engineering*, 2, 2021. doi:10.1017/dce.2021.10.

[4] J. Akroyd, Z. Harper, D. Soutar, F. Farazi, A. Bhave, S. Mosbach, and M. Kraft. Universal Digital Twin: Land use. *Data-Centric Engineering*, 3:e3, 2022. doi:10.1017/dce.2021.21.

[5] D. Allemang and J. Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2011. ISBN 9780123859655.

[6] C. B. Aranda, O. Corby, S. Das, L. Feigenbaum, P. Gearon, B. Glimm, S. Harris, S. Hawke, I. Herman, N. Humfrey, N. Michaelis, C. Ogbuji, M. Perry, A. Passant, A. Polleres, E. Prud'hommeaux, A. Seaborne, and G. T. Williams. SPARQL 1.1 Overview, W3C Recommendation 21 March 2013. World Wide Web Consortium (W3C), 2013. Available at https://www.w3.org/TR/sparql11-overview/ (accessed Apr 2023).

[7] J. Atherton, W. Xie, L. K. Aditya, X. Zhou, G. Karmakar, J. Akroyd, S. Mosbach, M. Q. Lim, and M. Kraft. How does a carbon tax affect Britain's power generation composition? *Applied Energy*, 298:117117, 2021. doi:10.1016/j.apenergy.2021.117117.

[8] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, Cambridge, UK, 2nd edition, 2007. ISBN 978-0-521-87625-4.

[9] J. Bai, L. Cao, S. Mosbach, J. Akroyd, A. A. Lapkin, and M. Kraft. From Platform to Knowledge Graph: Evolution of Laboratory Automation. *JACS Au*, 2(2):292–309, 2022. doi:10.1021/jacsau.1c00438.

[10] J. Bai, K. F. Lee, M. Hofmeister, S. Mosbach, J. Akroyd, and M. Kraft. A derived information framework for a dynamic knowledge graph and its application to smart cities, 2023. Submitted for publication. Available from https://como.ceb.cam.ac.uk/preprints/302/.

[11] T. Berners-Lee. Linked data - design issues, 2006. Available at http://www.w3.org/DesignIssues/LinkedData.html (accessed Apr 2023).

[12] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):28–37, 2001.

[13] C. Bizer, T. Heath, and T. Berners-Lee. Linked data: The story so far. In A. Sheth, editor, *Semantic Services, Interoperability and Web Applications: Emerging Concepts*. IGI Global, 2011. doi:10.4018/978-1-60960-593-3.ch008.

[14] Blazegraph, 2020. Available at https://blazegraph.com (accessed Feb 2023).

[15] Blazegraph, 2020. Available at https://github.com/blazegraph/database/wiki/GeoSpatial (accessed Mar 2023).

[16] E. Botoeva, D. Calvanese, B. Cogrel, M. Rezk, and G. Xiao. OBDA beyond relational DBs: A study for MongoDB. In M. Lenzerini and R. Peñaloza, editors, *Proc. of the 29th Int. Workshop on Description Logics (DL 2016)*, volume 1577 of *CEUR Workshop Proceedings, http://ceur-ws.org*, 2016.

[17] Brick Schema. Brick Ontology, 2022. Available at https://brickschema.org/ontology (accessed Apr 2023).

[18] P. Buttigieg, C. Mungall, C. Rueda, L. McGibbney, S. Cox, R. Duerr, D. Fils, J. Graybeal, B. Huffer, T. Narock, B. Ramachandran, S. Soiland-Reyes, and B. Whitehead. SWEET Ontology Suite v3.0.0: Development, Alignments and Use Cases. In *American Association of Geographers 2018 (AAG 2018)*, New Orleans, United States, 2018. doi:102.100.100/87125. Available at https://github.com/ESIPFed/sweet (accessed Feb 2023).

[19] P. L. Buttigieg, E. Pafilis, S. E. Lewis, M. P. Schildhauer, R. L. Walls, and C. J. Mungall. The environment ontology in 2016: bridging domains with increased scope, semantic density, and interoperation. *Journal of Biomedical Semantics*, 7 (57), 2016. doi:10.1186/s13326-016-0097-6. Available at https://raw.githubusercontent.com/EnvironmentOntology/envo/master/envo.owl (accessed Feb 2023).

[20] Centre Universitaire d'Informatique at University of Geneva, 2012. Available at http://cui.unige.ch/isi/onto/citygml2.0.owl (accessed Dec 2018).

[21] A. Chadzynski, N. Krdzavac, F. Farazi, M. Q. Lim, S. Li, A. Grisiute, P. Herthogs, A. von Richthofen, S. Cairns, and M. Kraft. Semantic 3D City Database — An enabler for a dynamic geospatial knowledge graph. *Energy and AI*, 6:100106, 2021. doi:10.1016/j.egyai.2021.100106.

[22] A. Chadzynski, S. Li, A. Grisiute, J. Chua, J. Yan, H. Y. Tai, E. Lloyd, M. Agarwal, M. Hofmeister, J. Akroyd, P. Herthogs, and M. Kraft. Semantic 3D City Interfaces - intelligent interactions on Dynamic Geospatial Knowledge Graphs, 2022. Submitted for publication. Available from https://como.ceb.cam.ac.uk/preprints/297/.

[23] A. Chadzynski, S. Li, A. Grisiute, F. Farazi, C. Lindberg, S. Mosbach, P. Herthogs, and M. Kraft. Semantic 3D City Agents—An intelligent automation for dynamic geospatial knowledge graphs. *Energy and AI*, 8:100137, 2022. doi:10.1016/j.egyai.2022.100137.

[24] A. Chadzynski, H. Silvennoinen, A. Grisiute, F. Farazi, S. Mosbach, M. Raubal, P. Herthogs, and M. Kraft. Semantic 3D City Inferences - multi-domain reasoning on Dynamic Geospatial Knowledge Graphs, 2023. Submitted for publication. Available from https://como.ceb.cam.ac.uk/preprints/303/.

[25] A. Cohen and SeatGeek Inc. fuzzywuzzy: Fuzzy string matching in python, 2020. Available at https://pypi.org/project/fuzzywuzzy/ (accessed Mar 2023).

[26] S. Cox, C. Little, J. R. Hobbs, and F. Pan. Time Ontology in OWL, 2020. Available at https://www.w3.org/TR/owl-time/ (accessed Mar 2023).

[27] J. Cuenca and F. Larrinaga. DABGEO: Domain Analysis-Based Global Energy Ontology, 2019. Available at https://innoweb.mondragon.edu/ontologies/dabgeo/ (accessed Apr 2023).

[28] J. Cuenca, F. Larrinaga, and E. Curry. DABGEO: A reusable and usable global energy ontology for the energy domain. *Journal of Web Semantics*, 61-62:100550, 2020. doi:10.1016/j.websem.2020.100550.

[29] Data & Knowledge Group. HermiT OWL Reasoner, 2019. Available at http://www.hermit-reasoner.com/ (accessed Apr 2023).

[30] Department for Environment Food & Rural Affairs. Real Time flood-monitoring API, 2021. Available at https://environment.data.gov.uk/flood-monitoring/doc/reference (accessed Feb 2023).

[31] Department for Environment Food & Rural Affairs. Hydrology Data API, 2021. Available at https://environment.data.gov.uk/hydrology/doc/reference (accessed Feb 2023).

[32] Department for Levelling Up, Housing & Communities. Energy Performance of Buildings Data, 2022. Available at https://epc.opendatacommunities.org/docs/api (accessed Apr 2023).

[33] A. Devanand, G. Karmakar, N. Krdzavac, R. Rigo-Mariani, E. Y. S. Foo, I. A. Karimi, and M. Kraft. OntoPowSys: A power system ontology for cross domain interactions in an eco industrial park. *Energy and AI*, 1:100008, 2020. doi:10.1016/j.egyai.2020.100008.

[34] A. Dsouza, N. Tempelmeier, R. Yu, S. Gottschalk, and E. Demidova. WorldKG: A World-Scale Geographic Knowledge Graph. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management*. ACM, 2021. doi:10.1145/3459637.3482023. Available at https://www.worldkg.org/ (accessed May 2023).

[35] R. Dutta and A. Morshed. Performance evaluation of south esk hydrological sensor web: Unsupervised machine learning and semantic linked data approach. *IEEE Sensors Journal*, 13(10):3806–3815, 2013. doi:10.1109/JSEN.2013.2264666.

[36] Eclipse Foundation. RDF4J Documentation – GeoSPARQL, 2021. Available at https://rdf4j.org/documentation/programming/geosparql/ (accessed Apr2023).

[37] EDINA Digimap Ordnance Survey Service, 2022. Available at https://digimap.edina.ac.uk/ (accessed Apr 2023).

[38] A. Eibeck, A. Chadzynski, M. Q. Lim, K. Aditya, L. Ong, A. Devanand, G. Karmakar, S. Mosbach, R. Lau, I. A. Karimi, E. Y. S. Foo, and M. Kraft. A parallel world framework for scenario analysis in knowledge graphs. *Data-Centric Engineering*, 1, 2020. doi:10.1017/dce.2020.6.

[39] ETSI. SAREF: the Smart Applications REFerence ontology, 2020. Available at https://saref.etsi.org/core/ (accessed Feb 2023).

[40] ETSI. SAREF extension for building, 2020. Available at https://saref.etsi.org/saref4bldg/v1.1.2/ (accessed Mar 2023).

[41] European Environment Information and Observation Network. General Multilingual Environmental Thesaurus, 2021. Available at https://www.eionet.europa.eu/gemet/en/themes/ (accessed Feb 2023).

[42] F. Farazi, J. Akroyd, S. Mosbach, P. Buerger, D. Nurkowski, M. Salamanca, and M. Kraft. OntoKin: An ontology for chemical kinetic reaction mechanisms. *Journal of Chemical Information and Modeling*, 60(1):108–120, 2020. doi:10.1021/acs.jcim.9b00960.

[43] Flood Forecasting Centre. FGS Public API, 2017. Available at https://api.foursources.metoffice.gov.uk/docs/flood-guidance-statement-api-public# (accessed Feb 2023).

[44] Google. Digital buildings ontology, 2022. Available at https://github.com/google/digitalbuildings/tree/master/ontology (accessed Apr 2023).

[45] G. Gröger, T. H. Kolbe, C. Nagel, and K.-H. Häfele. OGC City Geography Markup Language (CityGML) Encoding Standard, Version 2.0.0, 2012. Available at http://www.opengeospatial.org/standards/citygml (accessed Mar 2023).

[46] HM Land Registry. HM Land Registry Open Data, 2022. Available at https://landregistry.data.gov.uk/ (accessed Apr 2023).

[47] HM Land Registry. Price Paid Linked Data, 2022. Available at https://landregistry.data.gov.uk/app/root/doc/ppd (accessed Apr 2023).

[48] HM Land Registry. UK House Price Index Linked Data, 2022. Available at https://landregistry.data.gov.uk/app/ukhpi/doc (accessed Apr 2023).

[49] HM Land Registry. Price Paid Linked Data, 2022. Available at http://landregistry.data.gov.uk/landregistry/query (accessed Apr 2023).

[50] M. Hofmeister, G. Brownbridge, M. Hillman, S. Mosbach, J. Akroyd, K. F. Lee, and M. Kraft. Cross-Domain Flood Risk Assessment for Smart Cities using Dynamic Knowledge Graphs, 2023. Submitted for publication. Available from https://como.ceb.cam.ac.uk/preprints/308/.

[51] K. Janowicz, A. Haller, S. J. Cox, D. Le Phuoc, and M. Lefrançois. SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, 56:1–10, 2019. doi:10.1016/j.websem.2018.06.003.

[52] J. M. Johnson, T. Narock, J. Singh-Mohudpur, D. Fils, K. C. Clarke, S. Saksena, A. Shepherd, S. Arumugam, and L. Yeghiazarian. Knowledge graphs to support real-time flood impact evaluation. *AI Magazine*, 43(1):40–45, 2022. doi:10.1002/aaai.12035.

[53] M. Jovanovik, T. Homburg, and M. Spasić. A geosparql compliance benchmark. *ISPRS International Journal of Geo-Information*, 10(7), 2021. ISSN 2220-9964. doi:10.3390/ijgi10070487.

[54] N. Karalis, G. Mandilaras, and M. Koubarakis. Extending the YAGO2 knowledge graph with precise geospatial knowledge. In *Lecture Notes in Computer Science*, pages 181–197. Springer International Publishing, 2019. doi:10.1007/978-3-030-30796-7_12.

[55] M. Katsumi. Building ontology, 2021. Available at http://ontology.eil.utoronto.ca/icity/Building/1.2/ (accessed Apr 2023).

[56] S. Khantong, M. N. A. Sharif, and A. K. Mahmood. An ontology for sharing and managing information in disaster response: An illustrative case study of flood evacuation. *International Review of Applied Sciences and Engineering IRASE*, 11: 22–33, 2020. doi:10.1556/1848.2020.00004.

[57] E. Klien, M. Lutz, and W. Kuhn. Ontology-based discovery of geographic information services—An application in disaster management. *Computers, Environment and Urban Systems*, 30(1):102–123, 2006. doi:10.1016/j.compenvurbsys.2005.04.002.

[58] G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation 10 February 2004. World Wide Web Consortium (W3C), 2004. Available at http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ (accessed Apr 2023).

[59] S. Kollarits, N. Wergles, H. Siegel, C. Liehr, S. Kreuzer, D. Torsoni, U. Sulzenbacher, J. Papez, R. Mayer, C. Plank, L. Maurer, S. Cimarosto, and E. Bukta. MONITOR – an ontological basis for risk management. Technical report, Heriot Watt University, 2009. Available at http://www.macs.hw.ac.uk/~yjc32/project/ref-ontology/ref-ontology%20examples/MONITOR_Base_Ontology_Report_1_0.pdf (accessed Feb 2023).

[60] A. Kondinski, J. Bai, S. Mosbach, J. Akroyd, and M. Kraft. Knowledge Engineering in Chemistry: From Expert Systems to Agents of Creation. *Accounts of Chemical Research*, 56(2):128–139, 2023. doi:10.1021/acs.accounts.2c00617.

[61] K. F. Lee, M. Hofmeister, S. Mosbach, and Y. K. Tsai. Time Series Ontology, 2021. Available at https://github.com/cambridge-cares/TheWorldAvatar/tree/main/JPS_Ontology/ontology/ontotimeseries (accessed Feb 2023).

[62] Q. Liu, Q. Bai, C. Kloppers, P. Fitch, Q. Bai, K. Taylor, P. Fox, S. Zednik, L. Ding, A. Terhorst, and D. McGuinness. An ontology-based knowledge management framework for a distributed water information system. *Journal of Hydroinformatics*, 15(4):1169–1188, 07 2012. doi:10.2166/hydro.2012.152.

[63] M. Masmoudi, M. H. Karray, S. Ben Abdallah Ben Lamine, H. B. Zghal, and B. Archimede. Memon: Modular environmental monitoring ontology to link heterogeneous earth observed data. *Environmental Modelling & Software*, 124: 104581, 2020. doi:10.1016/j.envsoft.2019.104581. Ontology available at https://github.com/MEMOntology/memon (accessed Feb 2023).

[64] M. Masmoudi, M. H. Karray, S. Ben Abdallah Ben Lamine, H. B. Zghal, and B. Archimede. MEMOn: Modular Environmental Monitoring Ontology to link heterogeneous Earth observed data. *Environmental Modelling & Software*, 124: 104581, 2020. doi:10.1016/j.envsoft.2019.104581. Available at https://github.com/MEMOntology/memon (accessed Feb 2023).

[65] Met Office. MetOffice DataPoint API, 2022. Available at https://www.metoffice.gov.uk/services/data/datapoint/api-reference (accessed Apr 2023).

[66] Met Office and Environment Agency. Check for flooding, 2022. Available at https://check-for-flooding.service.gov.uk/river-and-sea-levels (accessed Apr 2023).

[67] M. H. Mughal, Z. A. Shaikh, A. I. Wagan, Z. H. Khand, and S. Hassan. ORFFM: An ontology-based semantic model of river flow and flood mitigation. *IEEE Access*, 9:44003–44031, 2021. doi:10.1109/ACCESS.2021.3066255.

[68] A. Murdock, A. Harfoot, D. Martin, S. Cockings, and C. Hill. OpenPopGrid: an open gridded population dataset for England and Wales. GeoData, University of Southampton, 2015. Available at http://openpopgrid.geodata.soton.ac.uk/ (accessed Apr 2023).

[69] M. A. Musen. The protégé project: a look back and a look forward. *AI Matters*, 1 (4):4–12, 2015. doi:10.1145/2757001.2757003.

[70] N. Noy and D. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory, Technical Report KSL-01-05*, 2001.

[71] Office for National Statistics. ONS Geography Linked Data: SPARQL 1.1 Query: Endpoint, 2022. Available at https://statistics.data.gov.uk/sparql (accessed Apr 2023).

[72] OGC. OGC GeoSPARQL - A Geographic Query Language for RDF Data, 2012. Available at https://www.ogc.org/standards/geosparql/ (accessed Mar 2023).

[73] OGC. CityGML, 2021. Available at https://www.ogc.org/standard/citygml/ (accessed Mar 2023).

[74] R. S. Oktari, K. Munadi, R. Idroes, and H. Sofyan. Knowledge management practices in disaster management: Systematic review. *International journal of disaster risk reduction*, 51:101881, 2020.

[75] Ontology Engineering Group (OEG). Building ontology, 2020. Available at https://bimerr.iot.linkeddata.es/def/building/ (accessed Apr 2023).

[76] Ontology Engineering Group (OEG). Bimerr ontology network, 2020. Available at https://bimerr.iot.linkeddata.es/ (accessed Apr 2023).

[77] Ordnance Survey. Ordnance Survey Data Hub, 2022. Available at https://osdatahub.os.uk/ (accessed Apr 2023).

[78] S. H. Pour, A. K. A. Wahab, S. Shahid, M. Asaduzzaman, and A. Dewan. Low impact development techniques to mitigate the impacts of climate-change-induced urban floods: Current trends, issues and challenges. *Sustainable Cities and Society*, 62:102373, 2020. doi:10.1016/j.scs.2020.102373.

[79] M. Pritoni, D. Paine, G. Fierro, C. Mosiman, M. Poplawski, A. Saha, J. Bender, and J. Granderson. Metadata Schemas and Ontologies for Building Energy Applications: A Critical Review and Use Case Analysis. *Energies*, 14(7), 2021. doi:10.3390/en14072024.

[80] Project Haystack. Haystack 4 defs, 2022. Available at https://github.com/Project-Haystack/haystack-defs (accessed Apr 2023).

[81] QGIS Development Team. QGIS Geographic Information System, 2021. URL http://qgis.osgeo.org.

[82] M. Rasmussen, P. Pauwels, M. Lefrançois, G. Schneider, C. Hviid, and J. Karlshøj. Building Topology Ontology, 2021. Available at https://w3c-lbd-cg.github.io/bot/ (accessed Apr 2023).

[83] RealEstateCore Consortium. Real Estate Core Ontology, 2020. Available at https://doc.realestatecore.io/3.2/core.html# (accessed Apr 2023).

[84] H. Rijgersberg, D. Willems, and J. Top. Ontology of units of Measure, 2020. Available at http://www.ontology-of-units-of-measure.org/page/om-2 (accessed Apr 2023).

[85] Safe Software Inc. FME Software ©, 2022. URL www.safe.com.

[86] T. Savage, J. Akroyd, S. Mosbach, N. Krdzavac, M. Hillman, and M. Kraft. Universal digital twin: Integration of national-scale energy systems and climate data. *Data-Centric Engineering*, 3, 2022. doi:10.1017/dce.2022.22.

[87] S. Scheuer, D. Haase, and V. Meyer. Towards a flood risk assessment ontology – Knowledge integration into a multi-criteria risk assessment approach. *Computers, Environment and Urban Systems*, 37:82–94, 2013. doi:10.1016/j.compenvurbsys.2012.07.007.

[88] Y. Sermet and I. Demir. Towards an information centric flood ontology for information management and communication. *Earth Science Informatics*, 12:541–551, 2019. doi:10.1007/s12145-019-00398-9.

[89] P. K. Sinha and B. Dutta. A Systematic Analysis of Flood Ontologies: A Parametric Approach. *Knowledge Organization*, 47:138–159, 2020. doi:10.5771/0943-7444-2020-2-138.

[90] C. Stadler, J. Lehmann, K. Höffner, and S. Auer. LinkedGeoData: A core for a web of spatial open data. *Semantic Web*, 3(4):333–354, 2012. doi:10.3233/sw-2011-0052.

[91] UK-AIR Sensor Observation Service (Beta release). Department for Environment Food & Rural Affairs, 2023. Available at https://uk-air.defra.gov.uk/data/about_sos (accessed Apr 2023).

[92] W3C. OWL Web Ontology Language Overview, 2012. Available at https://www.w3.org/TR/owl-features/ (accessed Apr 2023).

[93] C. Wang, N. Chen, W. Wang, and Z. Chen. A hydrological sensor web ontology based on the ssn ontology: A case study for a flood. *ISPRS International Journal of Geo-Information*, 7(1), 2018. doi:10.3390/ijgi7010002.

[94] X. Wang, H. Wei, N. Chen, X. He, and Z. Tian. An Observational Process Ontology-Based Modeling Approach for Water Quality Monitoring. *Water*, 12(3), 2020. doi:10.3390/w12030715.

[95] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, and et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(1): 160018, Mar 2016. ISSN 2052-4463. doi:10.1038/sdata.2016.18.

[96] World Wide Web Consortium. Semantic Sensor Network Ontology, 2017. Available at https://www.w3.org/TR/vocab-ssn/ (accessed Feb 2023).

[97] D. D. Wrachien, J. Garrido, S. Mambretti, and I. Requena. Ontology For Flood Management: A Proposal. In D. Proverbs, S. Mambretti, C. Brebbia, and D. D. Wrachien, editors, *Flood Recovery, Innovation and Response III*. WIT Press, 2012. doi:10.2495/FRIAR120011.

[98] G. Xiao, D. Lanti, R. Kontchakov, S. Komla-Ebri, E. Güzel-Kalaycı, L. Ding, J. Corman, B. Cogrel, D. Calvanese, and E. Botoeva. The virtual knowledge graph system ontop. In J. Z. Pan, V. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, and L. Kagal, editors, *The Semantic Web – ISWC 2020*, pages 259–277. Springer International Publishing, 2020.

[99] Z. Xiaomin, Y. Jianjun, H. Xiaoci, and C. Shaoli. An ontology-based knowledge modelling approach for river water quality monitoring and assessment. *Procedia Computer Science*, 96:335–344, 2016. doi:10.1016/j.procs.2016.08.146.

[100] X. Zhou, A. Eibeck, M. Q. Lim, N. Krdzavac, and M. Kraft. An agent composition framework for the J-Park Simulator – a knowledge graph for the process industry. *Computers & Chemical Engineering*, 130:106577, 2019. doi:10.1016/j.compchemeng.2019.106577.