Rainwater Ontology Documentation

Ontology

1. Overview

The Rainwater Ontology models the various components and processes involved in rainwater collection, storage, treatment, distribution, monitoring, and reuse. It provides a structured framework for representing knowledge about rainwater management systems.

In the future, the model will be extended to accommodate existing ontologies in order to avoid duplicate work. The starting point would be integrating “SSN SOSA” with the “Water Quality Sensor”.

This document describes the developed rainwater ontology along with its data, shapes and constraints. The ontology file is written in Turtle (TTL) format, which is a syntax for expressing data in the Resource Description Framework (RDF).

2. Use-Case Scenario: Smart Urban Rainwater Management in a Residential Community

In the context of sustainable urban development, effective rainwater management is crucial to reducing freshwater consumption and mitigating urban flooding. A residential community is implementing a comprehensive rainwater harvesting and reuse system, supported by semantic technologies. The Rainwater Ontology is employed as a foundational knowledge model to structure and manage data across all system components, from rainwater collection to water quality monitoring and reuse.

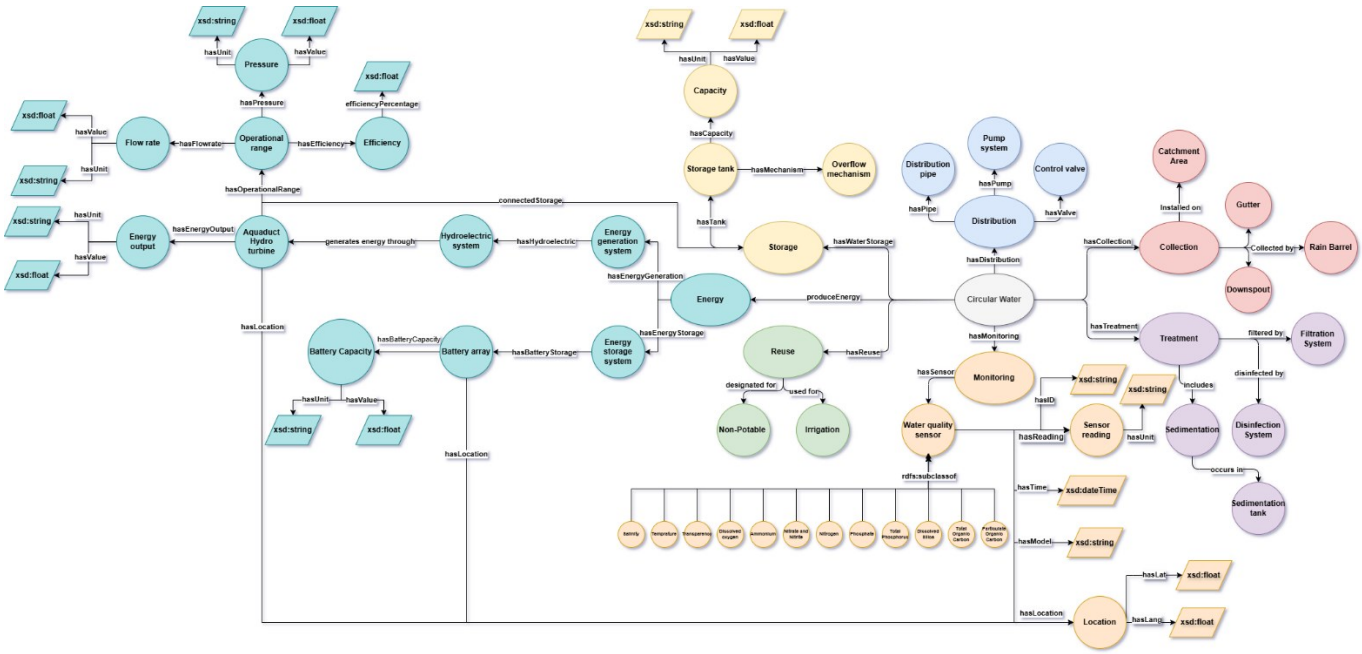
The system is designed to collect rainwater through rooftop gutters and downspouts feeding into rain barrels and underground storage tanks. The ontology’s Collection, CatchmentArea, RainBarrel, and StorageTank classes model these physical components. Water stored in tanks is subject to multiple treatment stages — including sedimentation, filtration, and UV disinfection — represented in the ontology through Treatment, SedimentationTank, FiltrationSystem, and DisinfectionSystem classes.

Real-time water quality monitoring is facilitated using a network of sensors measuring parameters such as pH, salinity, temperature, and nitrogen compounds. These sensors and their data are modeled via the WaterQualitySensor and SensorReading classes, along with associated properties like hasTime, hasLocation, and hasUnit. Integration with the SSN/SOSA ontology enables semantic interoperability with existing IoT frameworks.

The treated water is distributed through a network of pipes and pumps (DistributionPipe, PumpSystem), with the flow regulated by smart control valves (ControlValve). Reuse applications, such as garden irrigation and toilet flushing, are modeled using the Reuse, Irrigation, and NonPottable classes.

Data generated from the operational system is validated against SHACL constraints to ensure semantic compliance and consistency. This enables robust data integration, automated error detection, and streamlined analytics for decision support. The ontology-based model also allows stakeholders to query and visualize system performance over time, supporting proactive maintenance and water safety assurance.

By leveraging the Rainwater Ontology, this scenario demonstrates how semantic technologies can support intelligent, modular, and extensible rainwater management systems in real-world urban environments.



3. Ontology Metadata

- **URI**: http://example.org#RainWaterOntology
- **Label**: "Rain Water Ontology"
- **Description**: "This ontology models Rain water"
- **Version**: "1.0"

4. Core Classes

Main Class

- **RainWater** (ex:RainWater)
    - The main class representing rainwater in the system
    - Subclass of owl:Thing

Collection System

- **Collection** (ex:Collection)
    - Structures to collect rain
- **CatchmentArea** (ex:CatchmentArea)
    - Surface area where rainwater is collected

- **RainBarrel** (ex:RainBarrel)
- **Gutter** (ex:Gutter)
- **Downspout** (ex:Downspout)

Storage System

- **Storage** (ex:Storage)
- **StorageTank** (ex:StorageTank)
    - Containers where collected rainwater is stored
- **OverflowMechanism** (ex:OverflowMechanism)
    - Systems to handle excess water when storage tanks are full

Treatment System

- **Treatment** (ex:Treatment)
- **Sedimentation** (ex:Sedimentation)
    - Process where larger particles settle at the bottom of a tank
- **SedimentationTank** (ex:SedimentationTank)
- **FiltrationSystem** (ex:FiltrationSystem)
    - Devices or processes to remove particulates
- **DisinfectionSystem** (ex:DisinfectionSystem)
    - Methods to treat rainwater for potable uses

Distribution System

- **Distribution** (ex:Distribution)
- **DistributionPipe** (ex:DistributionPipe)
    - Networks that transport rainwater
- **PumpSystem** (ex:PumpSystem)
    - Devices that facilitate water movement
- **ControlValve** (ex:ControlValve)
    - Devices that regulate flow

Reuse System

- **Reuse** (ex:Reuse)
- **Irrigation** (ex:Irrigation)

- **NonPottable** (ex:NonPottable)

Monitoring System

- **Monitoring** (ex:Monitoring)
- **WaterQualitySensor** (ex:WaterQualitySensor)
    - Base class for all water quality sensors
- **SensorReading** (ex:SensorReading)
- **TimeStamp** (ex:TimeStamp)
- **Location** (ex:Location)

Specific Sensor Types (all subclass of WaterQualitySensor)

- **Salinity** (ex:Salinity)
- **Temperature** (ex:Temperature)
- **Transparency** (ex:Transparency)
- **DissolvedOxygen** (ex:DissolvedOxygen)
- **Ammonium** (ex:Ammonium)
- **NitrateNitrite** (ex:NitrateNitrite)
- **TotalNitrogen** (ex:TotalNitrogen)
- **Phosphate** (ex:Phosphate)
- **TotalPhosphorus** (ex:TotalPhosphorus)
- **DissolvedSilica** (ex:DissolvedSilica)
- **TotalOrganicCarbon** (ex:TotalOrganicCarbon)
- **ParticulateOrganicCarbon** (ex:ParticulateOrganicCarbon)

5. Object Properties

RainWater Relationships

- **hasStorage** (ex:hasStorage)
    - Domain: RainWater
    - Range: Storage
- **hasReuse** (ex:hasReuse)
    - Domain: RainWater
    - Range: Reuse
- **hasDistribution** (ex:hasDistribution)

    - Domain: RainWater
    - Range: Distribution
- **hasMonitoring** (ex:hasMonitoring)
    - Domain: RainWater
    - Range: Monitoring
- **hasCollection** (ex:hasCollection)
    - Domain: RainWater
    - Range: Collection
- **hasTreatment** (ex:hasTreatment)
    - Domain: RainWater
    - Range: Treatment

Distribution System Relationships

- **hasPipe** (ex:hasPipe)
    - Domain: Distribution
    - Range: DistributionPipe
- **hasPump** (ex:hasPump)
    - Domain: Distribution
    - Range: PumpSystem
- **hasValve** (ex:hasValve)
    - Domain: Distribution
    - Range: ControlValve

Storage System Relationships

- **hasTank** (ex:hasTank)
    - Domain: Storage
    - Range: StorageTank
- **hasmechanism** (ex:hasmechanism)
    - Domain: StorageTank
    - Range: OverflowMechanism

Monitoring System Relationships

- **hasSensor** (ex:hasSensor)

    - Domain: Monitoring
    - Range: WaterQualitySensor
- **hasReading** (ex:hasReading)
    - Domain: WaterQualitySensor
    - Range: SensorReading
- **hasLocation** (ex:hasLocation)
    - Domain: WaterQualitySensor
    - Range: Location

Reuse System Relationships

- **DesignatedFor** (ex:DesignatedFor)
    - Domain: Reuse
    - Range: NonPottable
- **UsedFor** (ex:UsedFor)
    - Domain: Reuse
    - Range: Irrigation

Collection System Relationships

- **InstalledOn** (ex:InstalledOn)
    - Domain: Collection
    - Range: CatchmentArea
- **CollectedBy** (ex:CollectedBy)
    - Domain: Collection
    - Range: Union of (Gutter, Downspout, RainBarrel)

Treatment System Relationships

- **includes** (ex:includes)
    - Domain: Treatment
    - Range: Sedimentation
- **OccursIn** (ex:OccursIn)
    - Domain: Sedimentation
    - Range: SedimentationTank
- **FilteredBy** (ex:FilteredBy)

    - Domain: Treatment
    - Range: FiltrationSystem
- **DisinfectedBy** (ex:DisinfectedBy)
    - Domain: Treatment
    - Range: DisinfectionSystem

Datatype Properties

Sensor Properties

- **hasID** (ex:hasID)
    - Domain: WaterQualitySensor
    - Range: xsd:string
- **hasTime** (ex:hasTime)
    - Domain: WaterQualitySensor
    - Range: xsd:dateTime
- **hasModel** (ex:hasModel)
    - Domain: WaterQualitySensor
    - Range: xsd:string

Reading Properties

- **hasUnit** (ex:hasUnit)
    - Domain: SensorReading
    - Range: xsd:string

Location Properties

- **hasLat** (ex:hasLat)
    - Domain: Location
    - Range: xsd:float
- **hasLang** (ex:hasLang)
    - Domain: Location
    - Range: xsd:float

6. Usage Notes

This ontology provides a comprehensive model for rainwater management systems, with particular emphasis on water quality monitoring through various sensor types. Each sensor type includes documentation about the measurement method used.

The ontology can be used to:

- Model rainwater collection and distribution systems
- Track water quality measurements over time
- Represent relationships between system components
- Support decision-making for rainwater reuse applications

Data generator

Due to the lack of real-world data, a data generator was designed to produce sample rainwater harvesting system with all components defined in the ontology. The data produced is compared against constraints written in SHACL to ensure data compliance. The script saves the generated data in Turtle format.

```
ex:system_1a2b3c4d a ex:RainWater ;
    ex:hasCollection ex:catchment_5e6f7g8h ;
    ex:hasDistribution ex:distribution_9i0j1k2l ;
    ex:hasMonitoring ex:monitoring_3m4n5o6p ;
    ex:hasReuse ex:reuse_7q8r9s0t ;
    ex:hasStorage ex:storage_u1v2w3x4 ;
    ex:hasTreatment ex:treatment_y5z6a7b8 .

```

Sample of generated data

Shapes & Constraints

The rules and constraints that the data must adhere to are written in SHACL (Shapes Constraint Language). It defines validation rules for the rainwater ontology. It's written in Turtle (TTL) format.

2. Key components

- sh:NodeShape: Defines constraints for a specific class
- sh:targetClass: Specifies which class the shape applies to
- sh:property: Defines constraints on properties
- sh:path: Indicates which property is being constrained
- sh:minCount: Requires at least N values for a property
- sh:class: Restricts property values to specific classes
- sh:or: Allows alternative valid classes/properties
- sh:message: Custom error messages for violations

Validator

This module is developed to act as a SHACL validator. It is designed to compare SHACL shapes to RDF-compliant data. It produces a report of errors, if they exist. The goal is to automate quality control for data pipelines or APIs. The module is generic and can be used on any SHACL and RDF data. The validator will be published separately as its own repository.