ISO/IEC 5087-2 CD 2

ISO/IEC JTC 1

Secretariat: ANSI

Information technology — City data model— Part 2: City level concepts

CD

Warning for WDs and CDs

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Revision History

Version	Changes
0.2	• - Section 1: updated Figure 1 to refer to the "service level" rather than "application level" • - Section 1: added a note on the iterative scope of the standard and updated the list of ontologies in the scope • - Section 4: added a note describing the table format used to formalize the ontologies. • - Throughout: updated IRI references to iso5087-1: http://ontology.eil.utoronto.ca/5087/5087-1/ • - Throughout: moved tables into Formalization Section • - Throughout: updated representation of change according to revised, simplified approach in 5087-1 • - Section 5.6: Added the concept "Municipality" to the Organization Ontology • - Removed direct references to TOVE Organization Ontology terms • - Updated govstat reference in Land Use Ontology to ISO21972 • - Added Contract Ontology (Section 6.13~~0~~) • - Added City Resident Ontology (6.8~~5.6~~) • - Added City Service Ontology (Section 0) • - Added City Ontology (Section 6.11~~5.9~~) • - Added Indicator Ontology (Section 5.9) • - Added Bylaw Ontology (Section 6.14~~5.13~~)
0.2.1	• Added PersonID to Person ontology • Updated Contact Ontology to remove direct reference to icontact ontology by copying classes and properties into this version • Added ForProfitOrganization, NonProfitOrganization and GovernmentOrganization as sub classes of Organization • Made City a subclassof Parcel • Made Person a subclass of Agent • Updated Figure 1 to make language more consistent • Updated Figure 2 to make language more consistent and added missing patterns • Throughout: updated section headings to identify ontology "Patterns" rather than label each as an Ontology • Moved Use Cases and Mappings (previously Sections 7 and 8) to appendix A and Appendix B • Renamed "references" to "bibliography" • Updated definitions in 3.1 through 3.10 per recommendations in document N904 • Added Use Case on epidemic tracking • Added audience statement to Introduction

	• Added hasProgram object property to City and CityDivision • Contact Pattern: Updated text referencing the use (not import) of the iContact.owl ontology • Added hasContract property to Contract Pattern • Added “hasProgram Program” to NonProfitOrganization and GovernmentOrganization • Deleted “partOf ImpactModel” in Service • Deleted “hasImpact Impact” in Outcome • Added Input and Output classes to Service Pattern • Removed consistsOf from Organization – replaced by hasSubOrganization inherited from 5087-1 • Replaced buildingHasLocation with hasLocation in Building • Changed hasRooms to hasNumberOfRooms in BuildingUnit • Added Facility in Building Pattern formalization table • Added Building properties: hasBuildingFootprintArea, hasBuildingFloorArea, hasBuildingHeight, hasNumberOfFloors
0.22	• Added a note on what is outside of the scope • Added Transportation Infrastructure Pattern • Added Infrastructure Pattern • Added Occupation to Organization; changed memberofDivision to 5087-1:member OrganizationalUnit • Added hasUse and hasFunction properties in Building Pattern • Renamed Parcel to LandArea • Updated Land Use Pattern: extensions with specific classification systems have been moved out of the standard and are now provided as an Appendix • Updated the definition of Person to include the possibility to specify Gender • Updated the definition of City Resident to be more general • Added CityDivision to Bylaw jurisdiction • Updated City definition to have hasLandArea and hasGovernment property to link to the landarea it occupies and the government organization that manages the city. • Removed taxonomy of Employee in Organization
0.23	• Updated Person pattern: ○ to use schema.org naming and birth/death dates ○ property description to conform to standard format ○ Updated PersonID to contain the type of ID and the time interval it is valid ○ Added “individual” property to Sex and IDType • Updated Organization pattern: ○ Added hasAddress property to Organization ○ Added hasTelephone property to Organization ○ Added hasOperatingHours to Organization ○ Added OperatingHours to pattern • Updated Contact pattern: ○ Added PhoneNumber class ○ Added hasAddress property ○ Added hasTelephone property • Added prefixes to all classes used by a pattern, but not defined in the pattern. • Corrected prefixes in Epidemic use case/example

2.4	• Added status to InfrastructureElement • ~~Add drones to sensors~~ – Not added as SSN ontology includes the class Platform which a Drone is a subclass of • Added contact prefix • Added PersonName and hasName to Person • Added email, citizenship, education to Person • DwellingUnit – ○ removed hasValue as inherited; ○ defined Tenure; ○ changed dwellingAddress to hasAddress • Deleted Family in Household pattern • Organization: did not add ID – inherited from organizationstructure:Organization • City: ~~JurisdictionalEntity (government), JurisdictionalAuthorty (connects entity to area), JurisdictionalArea~~ – did not change as GovernmentOrganization already contains property hasJurisdiction.
CD 1	• JP 44-001, JP-45-002: Added labels to all tables and references to tables in the text (where missing) • Replaced outdated labels of “Ontology” with “Pattern” for consistency • JP 46-003: checked and updated use of “may” and “can” as appropriate according to ISO/IEC Directives Part 2 (sub-clause 7.4 Permission, 7.5 Possibility and capability) • JP 03-007: changed normative reference to ISO 5087-1 to ISO/IEC 5087-1 • JP 01-006: revised the Scope text to match the approved NP scope • KR 2-044: revised the Scope text to clarify the choice of categories • KR 2-044: added a NOTE to the City Resident pattern to explain why it is distinguished from Person • KR 2-044: added a NOTE to the City Services pattern to explain why Applications have not been included • KR 2-044: added a NOTE to the Sensors pattern to explain why Actuators have not been included • JP 04-008: corrected the namespace definition for i72 • JP 05-008: corrected the namespace definition for org, added a new namespace definition for org_city (the Organization Pattern defined in this standard) • JP 06-010: corrected the namespace prefix for org_s • JP 07-011: removed old references to the prefix iso5087-1 • JP 08-012,…: changed hanging paragraphs to subclauses titled “General” and renumbered subsequent subclauses • JP 10-014,…: changed references to “ISO 5087-1” to “ISO/IEC 5087-1” • JP 39-043: updated text reference to SSN ontology • JP 34-039: Corrected section title of Contact Pattern • JP 34-039: Added a section on the mapping to ISO 19160-1 in Annex B
CD 2	• Enhanced interpretation of administrative areas in a city ○ CityDivision → CityAdministrativeArea ○ Added ▪ City hasAdministrativeArea CityAdministrativeArea ▪ CityAdministrativeArea hasAdministrativeArea CityAdministrativeArea ▪ CityAdministrativeArea AdministrativeAreaOf CityAdministrativeArea ○ Removed ▪ CityAdministrativeArea hasProperPart ▪ CityAdministrativeArea properPartOf ▪ City hasProperPart

	• Updated scope text • Added title before scope section to conform to ISO template • Moved listing of owl file locations to annex • Updated IRIs to use ISO-issued namespaces
	• Replaced ResidentialRelationship with HousingTenure • Added forTime to Household allowing for the temporal limitation of a household, i.e., it may no longer exist. • Added hasAddress Address to InfrastructureELement to allow for the assignment of an address

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2. www.iso.org/directives

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received. www.iso.org/patents

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1.

ISO/IEC ##### is based on work developed in the Enterprise Integration Laboratory of the University of Toronto.

Introduction

Cities today face a challenge of how to integrate data from multiple, unrelated sources where the semantics of the data are imprecise, ambiguous and overlapping. This is especially true in a world where more and more data of interest is being openly published from various organizations. A morass of data is increasingly becoming available to support city planning and operations activities. In order to be used effectively, the data must be unambiguously understood so that it can be correctly combined, avoiding data silos. Early successes in data "mash-ups" relied upon an independence assumption, where unrelated data sources were linked based solely on geospatial location, or a unique identifier for a person or organization. More sophisticated analytics projects that require the combination of datasets with overlapping semantics entail a significantly greater effort to transform data into something useable. It has become increasingly clear that integrating separate datasets for this sort of analysis requires an attention to the semantics of the underlying attributes and their values.

A common data model enables city software applications to share information, plan, coordinate, and execute city tasks, and support decision making within and across city services, by providing a precise, unambiguous representation of information and knowledge commonly shared across city services. This requires a clear understanding of the terms used in defining the data, as well as how they relate to one another. This requirement goes beyond syntactic integration (e.g. common data types and protocols), it requires semantic integration: a consistent, shared understanding of the meaning of information.

To motivate the need for a standard city data model, consider the evolution of cities. Cities deliver physical and social services that traditionally have operated as silos. If during the process of becoming smarter, transportation, social services, utilities, etc. were to develop their own data models, then we would have smarter silos. To create truly smart cities data must be shared across these silos which can only be accomplished through the use of a common data model. For example, "Household" is a category of data that is commonly used by city services. Members of Households are the source of transportation, housing, education, and recreation demand. It represents who occupies a home, age, occupations, where they work, abilities, etc. Though each city service can gather and/or use different aspects of a Household, much of the data needs to be shared with each other.

Supporting this interoperability among city datasets is particularly challenging due to the diversity of the domain and the heterogeneity of its data sources. The purpose of this document is to support the precise and unambiguous specification of city data using the technology of Ontologies [1, 2] as implemented in the Semantic Web [3]. By doing so it will:

- enable the computer representation of precise definitions thereby reducing the ambiguity of interpretation,
- remove the independence assumption, thereby allowing the world of Big Data, open source software, mobile apps, etc., to be applied for more sophisticated analysis,
- achieve semantic interoperability, namely the ability to access, understand, merge and use data available from datasets spread across the semantic web,
- enable the publishing of city data using Semantic Web and ontology standards, and
- enable the automated detection of city data inconsistency, and the root causes of variations.

With a clear semantics for the terminology, it is possible to perform consistency analysis, and thereby validate the correct use of the standard.

Figure 1~~Figure 1~~ identifies the three levels of the standard. The lowest level, defined in Part 1 of this standard provides the classes, properties and logical, computational definitions for representing the concepts that are foundational to representing any data. The middle level, defined in part 2 of this standard, provides the classes, properties and logical, computational definitions for representing urban specific concepts common to all city services but not specific to any service. The top level provides the classes, properties and logical, computational definitions for representing service specific concepts that are used by other services across the city. Part 3 of this standard defines the Transportation concepts. In the future, additional parts will be added to the standard covering services such as Education, Water, Sanitation, Energy, etc.

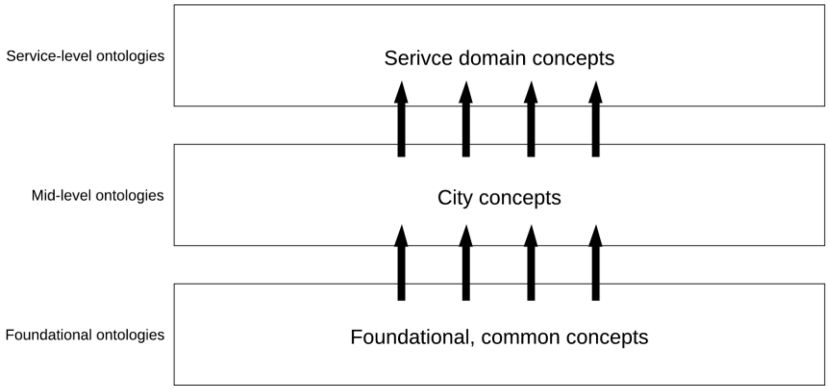

Figure 1: Stratification of City Data Model.

Figure 2 depicts example concepts for the three levels. Level 1, as defined in part 1 of this standard includes concepts of Location, Time, Unit of Measure, Change, etc. Level 2, as defined in part 2 of this standard includes concept of Land Use, Building, Household, etc. Level 3, as defined in part 3 of this standard defines transportation concepts such as Vehicle, Trips, Transportation Network, etc.

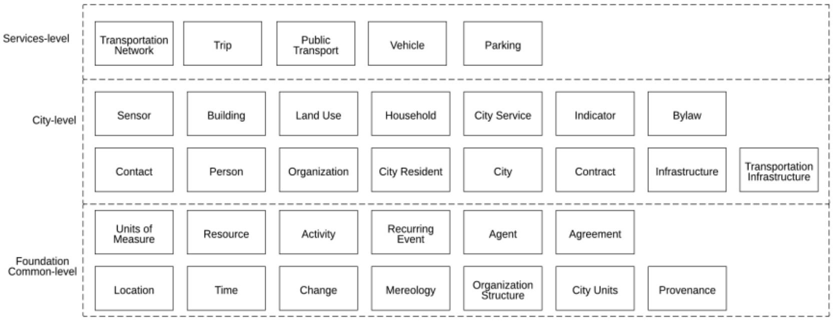

Figure 2: Example Concepts for each Level

The audience for this standard includes municipal information systems departments, municipal software designers and developers, and organizations that design and develop software for municipalities.

Information technology — City data model— Part 2: City level concepts

1 Scope

This document defines an ontology for city-level concepts defined using terms specified in ISO/IEC 5087-1. City-level concepts are used to represent data that is shared across multiple services and stakeholders in the city. City-level concepts are distinguished by their data being read and updated by multiple city services and stakeholders.

2 Normative References

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced documents (including any amendments) applies.

SEMANTIC SENSOR NETWORK ONTOLOGY. W3C Recommendation 19 October 2017, https://www.w3.org/TR/vocab-ssn/

ISO/IEC 5087-1, *Information technology — City Data Model — Part 1: Foundation Level Concepts*

3 Terms and Definitions

For the purposes of this document, the following apply. ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https: //www .iso .org/obp
— IEC Electropedia: available at http://www.electropedia .org/

3.1
cardinality
number of elements in a set

[SOURCE: ISO/TS 21526:2019, 3.11]

3.2
description logic (DL)
family of formal knowledge representation languages that are more expressive than propositional logic but less expressive than first-order logic

[SOURCE:ISO/IEC 21972:2020, 3.2]

3.3
manchester syntax
compact, human readable syntax for expressing Description Logic descriptions

[SOURCE: https://www.w3.org/TR/owl2- manchester-syntax/]

3.4

measure

value of the measurement (via the numerical_value property) which is linked to both Quantity and Unit_of_measure

[SOURCE: ISO/IEC 21972:2020, 3.4]

3.5

namespace

collection of names, identified by a URI reference, that are used in XML documents as element names and attribute names

Note 1 to entry: names may also be identified by a IRI reference.

[ISO/IEC 21972:2020, 3.5, modified – Note 1 to entry has been added]

3.6

ontology

formal representation of phenomena of a universe of discourse with an underlying vocabulary including definitions and axioms that make the intended meaning explicit and describe phenomena and their interrelationships

[SOURCE: ISO 19101-1:2014, 4.1.26]

3.7

ontology web language

ontology language for the semantic Web with formally defined meaning

Note 1 to entry: OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents.

[ISO/IEC 21972:2020, 3.7]

3.8

quantity

property of a phenomenon, body, or substance, where the property has a magnitude that can be expressed by means of a number and a reference

Note 1 to entry: Quantities can appear as base quantities or derived quantities.

EXAMPLE 1: Length, mass, electric current (ISQ base quantities).

EXAMPLE 2: Plane angle, force, power (derived quantities).

[SOURCE: ISO 80000-1:2009, 3.1, modified — NOTEs 1 to 6 have been removed; new Note 1 to entry and two EXAMPLEs have been added.]

3.9

Semantic Web

W3C’s vision of the Web of linked data

Note 1 to entry: Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data.

[SOURCE: https://www.w3.org/standards/semanticweb/]

3.10
unit_of_measure
actual units in which some quantity is measured

[SOURCE: ISO 11179-3:2003, 3.3.1334 modified.]

4 Symbols and Abbreviated Terms

DL	description logic
OWL	ontology web language
RDF	resource description framework
RDFS	resource description framework schema
IRI	international resource identifier

The following namespace prefixes are used in this document:

- activity: https://standards.iso.org/iso-iec/5087/-1/ed-1/en/ontology/Activity/
- agent: https://standards.iso.org/iso-iec/5087/-1/ed-1/en/ontology/Agent/
- agreement: https://standards.iso.org/iso-iec/5087/-1/ed-1/en/ontology/Agreement/
- building: https://standards.iso.org/iso-iec/5087/-2/ed-1/en/ontology/Building/
- bylaw: https://standards.iso.org/iso-iec/5087/-2/ed-1/en/ontology/Bylaw/
- city: https://standards.iso.org/iso-iec/5087/-2/ed-1/en/ontology/City/
- cityresident: https://standards.iso.org/iso-iec/5087/-1/ed-1/en/ontology/CityResident/
- cityunits: https://standards.iso.org/iso-iec/5087/-1/ed-1/en/ontology/CityUnits/
- code: https://standards.iso.org/iso-iec/5087/-2/ed-1/en/ontology/Code/
- contact: https://standards.iso.org/iso-iec/5087/-2/ed-1/en/ontology/Contact/
- contract: https://standards.iso.org/iso-iec/5087/-2/ed-1/en/ontology/Contract/
- genprop: https://standards.iso.org/iso-iec/5087/-1/ed-1/en/ontology/GenericProperties/
- geo: http://www.opengis.net/ont/geosparql#
- i72: http://ontology.eil.utoronto.ca/ISO21972/iso21972/
- infras: https://standards.iso.org/iso-iec/5087/-2/ed-1/en/ontology/Infrastructure/
- landuse: https://standards.iso.org/iso-iec/5087/-2/ed-1/en/ontology/Landuse/
- org: http://www.w3c.org/ns/org#
- org_s: https://standards.iso.org/iso-iec/5087/-1/ed-1/en/ontology/OrganizationStructure/
- org_city: https://standards.iso.org/iso-iec/5087/-2/ed-1/en/ontology/Organization/
- owl: http://www.w3.org/2002/07/owl#
- partwhole: https://standards.iso.org/iso-iec/5087/-1/ed-1/en/ontology/Mereology/
- person: https://standards.iso.org/iso-iec/5087/-2/ed-1/en/ontology/Person/
- rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
- rdfs: http://www.w3.org/2000/01/rdf-schema#

- recurringevent: https://standards.iso.org/iso-iec/5087/-1/ed-1/en/ontology/RecurringEvent/
- resource: https://standards.iso.org/iso-iec/5087/-1/ed-1/en/ontology/Resource/
- loc: https://standards.iso.org/iso-iec/5087/-1/ed-1/en/ontology/SpatialLoc/
- service: https://standards.iso.org/iso-iec/5087/-2/ed-1/en/ontology/CityService/
- transinfras: https://standards.iso.org/iso-iec/5087/-2/ed-1/en/ontology/TransportationInfrastructure/
- time: http://www.w3.org/2006/time#
- xsd: http://www.w3.org/2001/XMLSchema#

The formalization of the classes in this document is specified using the following table format, which is a simplification of DL where the first column identifies the class name, the second column its properties and the third column each property's range restriction It shall be read as: The <Class> is a subClassOf the conjunction of the associated <property>s with their <value>s. Range restrictions are specified using the Manchester syntax. For example, **Error! Reference source not found.**~~Table 1~~ specifies that Agent is a subclass of the intersection of (Person or Organization) and org:memberOf only Organization.

Table 1: Example formalization of the Agent class

Class	Property	Value Restriction
Agent	rdfs:subClassOf	Person or org_s:Organization
	org_s:memberOf	only Organization
	individual	{joe, frank}

CamelCase is used for specifying classes, properties and instances. For example, "legalName" instead of "legal_name". The first letter of a class name is capitalized. The first letter of a property and instance name are not capitalized. An instance of a class shall satisfy the class's definition. The instance's properties and values shall satisfy the value restrictions of the class it is an instance of.

The formalization of the properties in this document is done similarly, using the following table format that allows for the identification of properties and their sub-properties, inverse properties, or other characteristics. It is to be read as: The <property> is <characteristic> of <value>, or simply the <property> is <characteristic> if no value is applicable. For example, in Table 2 hasPrivilege is a sub-property of the agentInvolvedIn property. Characteristics are specified using the Manchester syntax.

Table 2: Example property formalization

Property	Characteristic	Value (if applicable)
hasPrivilege	rdfs:subPropertyOf	agentInvolvedIn
	Irreflexive	

In the case of DL definitions of classes where the simplified table representation is insufficient, the DL specification will be supplied.

The patterns defined in this document have also been implemented in OWL and made available online. The location of these encodings is identified in Annex D.

5 Unique identifiers

All classes, properties and instances of classes have a unique identifier that conforms to Linked Data/Semantic Web standards. The unique identifier is an IRI. When using ISO/IEC 5087-1 (this document) in an application, a class is identified by the IRI for the pattern of which it is a member, followed by the class name. In the Agent example in clause 4, the Agent class's unique identifier would be:

https://standards.iso.org/iso-iec/5087/-1/ed-1/en/ontology/Agent/Agent

Breaking the IRI down:

— "5087" identifies the series number

— "-1**"** identifies the part number

— "ed-1" indicates that the class is defined in edition 1 of the standard

— "en" indicates that the class is defined in a pattern implemented in English

— The first "Agent" identifies the Agent Pattern

— The second "Agent" identifies the Agent class within the Agent Pattern

The IRI can be shortened using the prefix's defined in Clause 4:

agent:Agent

where agent: is the prefix for the Agent Pattern.

Properties are identified in the same manner. The IRI's of individuals created by an application of ISO/IEC 5087-1 would have IRI's unique to the application.

6 City-Level Ontologies

6.1 General

Much like the foundational concepts defined in Part 1 are common across arbitrary domains, there are concepts that are generic across all cities. These concepts define the domain upon which city services operate – they are common for all cities but not specific to any one service. The City Data Model defines seven city-level ontologies to capture these concepts. These are described in the following sections.

Note that the cardinality restrictions specified in the pattern formalizations are weakened intentionally in many cases to support the pragmatic implementation of the ISO/IEC 5087 series of standards. For example, while common sense dictates that all persons should have at exactly one legal name, in practice there are many possible applications of city data where persons may be represented with no name at all. Users of this document are encouraged to extend and strengthen these restrictions where possible or warranted by the application.

6.2 Code Pattern

6.2.1 General

The Code Pattern provides a structure to address the challenge of value enumeration with a general approach. In city data there are many classes of things that are intended to be instantiated using a set list of values (e.g., classification systems), however these values may change based on application or context. In such cases it is not desirable for a standard to prescribe a restricted set of possible values which may not satisfy the needs of all applications. On the other hand, leaving the values completely open-ended provides no utility for interoperability. The Code Pattern provides an intermediate solution for this challenge by introducing a generic set of classes and properties that can be used to extend such classes to define various classification systems in an integrated way.

Instead of enumerating value sets for classes in this document, values can be defined with an associated Code that specifies additional metadata about the value and its origins. This allows these classes to be extended with various value-systems as required by a particular application, while providing the necessary information to support interpretation and integration as needed.

6.2.2 Key Classes & Properties

The key classes and properties are formalized in Table 3 and Table 4, respectively. A Code is introduced to capture the possible value of an object, according to some predefined system of values. It has the following key properties:

- definedBy: identifies the Organization that defined the code.
- ~~hasSpecification~~specification: specifies a URI where the definition of the code can be found.
- hasIdentifier: identifies a unique identifier for the code.
- genprop:hasName: specifies a name or title for the code.
- genprop:hasDescription: specifies a description of the code.

6.2.3 Formalization

Table 3: Key classes in the Code Pattern

Class	Property	Value Restriction
Code	definedBy	max 1 org_s:Organization
	~~hasSpecification~~specification	only xsd:string
	genprop:hasIdentifier	max 1 xsd:string
	genprop:hasName	only xsd:string
	genprop:hasDescription	only xsd:string

Table 4: Key properties in the Code Pattern

Property	Characteristic	Value (if applicable)
hasCode	rdfs:range	Code

6.3 Infrastructure Pattern

6.3.1 General

The Infrastructure pattern defines the concepts needed to capture various types of city infrastructure, such as buildings and roads. The Infrastructure pattern reuses the Spatial Location pattern (from ISO/IEC 5087-1) in order to capture the location of these infrastructure elements.

It is extended by the Building pattern, and the Transportation Infrastructure pattern. It can be extended with other types of infrastructure as required.

6.3.2 Key Classes & Properties

The key classes are formalized in Table 5~~Table 2~~. An Infrastructure Element is a generic representation of a city structure of interest. All infrastructure elements may have some defined location and shall be associated with some location, where locations are spatial Features as defined in ISO/IEC 5087-1. The Mereology pattern (from ISO/IEC 5087-1) is also reused in order to support the possible representation of infrastructure parts (e.g. road segments) and their associated wholes (e.g. the entire road). The following are its properties:

- **loc:hasLocation**: specifies the location of the element as a loc:Location.
- **partwhole:hasProperPart**: specifies any sub-parts of the element.
- **genprop:hasIdentifier**: specifies identifiers for the element provided by the city.
- **genprop:hasName**: specifies names of the element.
- **genprop:hasDescription**: specifies descriptions of the element.
- **contact:hasAddress**: specifies the address of the element as a contact:Address.
- **i72:hasValue**: identifies the value of a Building as a cityunits:MonetaryValue. Note that distinctions may be made between different types of value, such as the government-assessed value of a building or the purchase price. Subproperties of hasValue may be introduced to distinguish these types as required. Suggested possible extensions include: hasCost (purchase price), hasGovernmentAssessedValue (government assessed value for tax purposes), hasCollateralValue (value assessed in relation to a loan), hasInsuredValue (value determined by insurance policy).

6.3.3 Formalization

Table 5: Key classes in the Infrastructure Pattern

Class	Property	Value Restriction
InfrastructureElement	loc:hasLocation	only loc:Location
	partwhole:hasProperPart	only InfrastructureElement
	genprop:hasIdentifier	only xsd:string
	genprop:hasName	only xsd:string
	genprop:hasDescription	only xsd:string
	contact:hasAddress	only contact:Address
	i72:hasValue	only i72:MonetaryValue

6.4 Transportation Infrastructure Pattern

6.4.1 General

The Transportation Infrastructure pattern defines the concepts that are relevant in describing the physical transportation infrastructure and their characteristics. This includes the concepts of a Road, Bridge, and Tunnel. The Infrastructure Pattern is reused here, as these transportation structures are all defined as types of (subclasses) Infrastructure Elements.

6.4.2 Key Classes & Properties

The key classes are formalized in Table 6~~Table 3~~. They include the common structural elements that comprise the physical (land) transportation infrastructure: Roads, Rail Lines, Bridges and Tunnels.

A Travelled Way is a type of Infrastructure Element enables travel. It is defined with the following property:

- **aggregationOf**: identifies a Travelled Way Link that is aggregated to form the Travelled Way. Note that aggregationOf is distinct from parthood in that a Travelled Way Link does not depend on the Travelled Way in order to exist.

A Travelled Way Link represents a (continuous) length of a Travelled Way. The start and end of a Travelled Way is typically identified according to operational or managerial significance. It has the following property:

- aggregateOf: identifies a Travelled Way that aggregates the Travelled Way Link.

Travelled Way Links can be decomposed into Travelled Way Segments. Travelled Way Segments are typically defined based on some physical characteristics of the infrastructure (e.g. lane additions/removals, intersections). A Travelled Way Segment has the following property:

- partwhole:properPartOf: identifies the Travelled Way Link that it is part of.

A Road is a type of Travelled Way. It describes a part of the physical transportation infrastructure that has been improved to allow travel by motor vehicles, persons, bicycles, and similar methods of conveyance. It is identified as a Road as such by some governing body. It is defined with the following property:

- **aggregationOf**: identifies the Road Link that a Road may be decomposed into in order to represent its lengths at a finer granularity.

A Road Link is a type of Travelled Way Link that is represents a length of a Road. It has the following property:

- **aggregateOf**: identifies the Road that aggregates the Road Link

A Road Segment is a type of Travelled Way Segment that represents a part of a Road Link. It has the following property:

- **partwhole:properPartOf**: identifies the Road Link that the Road Segment is a part of.
- **networkType**: identifies the RoadNetworkType of the Road Segment. The RoadNetworkType identifies the modes of travel permitted or otherwise supported on a particular Road Segment (e.g. bicycles, motorized vehicles). Its values may be defined more precisely with the use of the code:hasCode property. The network types for Road Links and Roads may be defined based on the Road Segments they contain.

A Rail Line is a type of Travelled Way. It describes a part of the physical transportation infrastructure that has been fitted with tracks to allow travel by trains and other sorts of rail vehicles. No distinction is made between Rail Line types at this level. A Rail Line is identified as such by some governing body. It is defined with the following property:

- **aggregationOf**: identifies the Rail Link that a Rail may be decomposed into in order to represent its lengths at a finer granularity.

A Rail Link is a type of Travelled Way Link that represents a length of a Rail Line. It has the following property:

- **aggregateOf**: identifies the Rail Line that the Rail Link is an aggregate of.

A Rail Segment is a type of Travelled Way Segment that represents part of a Rail Link. It has the following property:

- **partwhole:properPartOf**: identifies the Rail Link that the Rail Segment is a part of.

A Bridge is a type of Infrastructure Element. It describes a part of the physical transportation infrastructure that enables travel over some area. It may contain some Road Segments or Rail Line Segments. A Bridge is identified as such by some governing body. It is defined with the following properties:

- **partwhole:hasProperPart:** identifies the Bridge Segments that a Bridge may be decomposed into to represent its lengths at a finer granularity.
- **supports:** identifies Travelled Way Segments that are supported by (i.e. travel over) the Bridge, if any.

A Bridge Segment is another type of Infrastructure Element. It is part of a Bridge and is defined with the following property:

- **supports:** identifies Travelled Way Segments that are supported by (i.e. travel over) the Bridge Segment, if any.

A Tunnel is a type of Infrastructure Element. It describes a part of the physical transportation infrastructure that enables travel underneath some area. It may contain some Road or Rail Line segments. A Tunnel is identified as such by some governing body. It is defined with the following properties:

- **partwhole:hasProperPart:** identifies the Tunnel Segments that a Tunnel may be decomposed into to represent its lengths at a finer granularity.
- **supports:** identifies Travelled Way Segments that are supported by (i.e. travel through) the Tunnel, if any.

A Tunnel Segment is another type of Infrastructure Element, it is part of a Tunnel and is defined with the following property:

- **supports:** identifies Travelled Way Segments that are supported by (i.e. travel over) the Tunnel Segment, if any.

6.4.3 Formalization

Table 6: Key classes in the Transportation Infrastructure Pattern

Class	Property	Value Restriction
TravelledWay	rdfs:subClassOf	infras:InfrastructureElement
	aggregationOf	only TravelledWayLink
TravelledWayLink	rdfs:subClassOf	infras:InfrastructureElement
	aggregateOf	only TravelledWay
TravelledWaySegment	rdfs:subClassOf	infras:InfrastructureElement
	partwhole:properPartOf	some TravelledWayLink
Road	rdfs:subClassOf	TravelledWay
	aggregationOf	only RoadLink

RoadLink	rdfs:subClassOf	TravelledWayLink
	aggregateOf	only Road
RoadSegment	~~Rdfs~~rdfs:subClassOf	TravelledWaySegment
	~~Partwhole~~partwhole:properPartOf	some RoadLink
	networkType	only RoadNetworkType
RoadNetworkType	code:hasCode	only code:Code
RailLine	rdfs:subClassOf	TravelledWay
	aggregationOf	only RailLink
RailLink	rdfs:subClassOf	TravelledWayLink
	aggregateOf	only RailLine
RailSegment	rdfs:subClassOf	TravelledWaySegment
	partwhole:properPartOf	some RailLink
Bridge	rdfs:subclassOf	InfrastructureElement
	partwhole:hasProperPart	only BridgeSegment
	supports	only TravelledWaySegment
BridgeSegment	rdfs:subClassOf	InfrastructureElement
	supports	only TravelledWaySegment
Tunnel	rdfs:subClassOf	InfrastructureElement
	partwhole:hasProperPart	TunnelSegment
	supports	only TravelledWaySegment
TunnelSegment	rdfs:subClassOf	InfrastructureElement
	supports	only TravelledWaySegment

6.5 Building Pattern

6.5.1 General

The Building pattern defines the concepts to capture information about individual buildings, thus describing land use from a different perspective and at a finer level of granularity than typical land use classifications. The Infrastructure Pattern is reused here, as Buildings and Building Units are defined as types of (subclasses) Infrastructure Elements. The Building pattern also reuses the Spatial Location pattern in order to capture the location of a building. Other attributes of a building are also captured, such as the type of building, units contained in a building, their monetary value, and so on. The Mereology pattern from ISO/IEC 5087-1 is referenced to capture the disaggregate parts of a building, and the City Units pattern from ISO/IEC 5087-1 is referenced to capture attributes required for land value considerations, such as sale prices and square footage.

6.5.2 Key Classes & Properties

The key classes and properties are formalized in Table 7~~Table 4~~ and Table 8~~Table 5~~, respectively. A Building is a structure with a roof and walls. It is defined as a type of Infrastructure Element and has the following key properties:

- ~~**loc:associatedLocation**: A Building has a Location in space – this may be described in terms of both actual (e.g. 2D and 3D space) location occupied by the building and associated locations (e.g. a point coordinate).~~
- **buildingFacility**: identifies a Facility(s), e.g,, kitchen, bath, or air conditioning that is contained in the Building.
- **hasBuildingUnit**: identifies the Building Unit(s), if any, that are contained by the Building.
- **numBuildingUnits**: identifies the total number of Building Unit(s) in a Building as a (non-negative) integer value.
- **hasBuildingFootprintArea**: identifies the footprint occupied by the Building as a cityunits:Area quantity.
- hasBuildingFloorArea: identifies the floor area occupied by the Building as a cityunits:Area quantity. The floor area accounts for the area of each floor of the building. However, floor area excludes unoccupied areas such as basements.
- **floorAreaRatio**: identifies the building's floor area ratio as a cityunits:RatioIndicator quantity. This is a ratio of the gross floor area (the value of hasBuildingFloorArea) to its "buildable area" (i.e. lot size).
- **hasBuildingHeight**: identifies the Building's height as a cityunits:Length quantity.
- **numFloors**: identifies the number of floors in a Building as a (non-negative) integer value. This represents a total count of all floors in the building (including those below ground).
- **numAboveGroundFloors**: identifies the number of above ground floors in a Building a (non-negative) integer value.
- **windowToWallRatio**: identifies the percentage area of a building's exterior envelope that is made up of glazing (i.e., glass installed in fixed openings such as windows and doors). The value is specified as a cityunits:RatioIndicator quantity.
- **builtAccordingToConstructionCode**: identifies the name of any construction code(s) applicable during the construction of the Building. Construction codes refer to a set of rules or instructions that specify the standards for constructed objects such as buildings and nonbuilding structures. Buildings must conform to the code throughout the construction of the building.
- ~~**contact:hasAddress**: identifies the address of the Building, as defined in the Contact Pattern.~~
- **use**: identifies the use of the Building as a BuildingUse object(s), typically associated with some classification system.
- **hasConstructionStatus**: identifies the Construction Status of a Building.
- **yearOfConstruction**: identifies the year that construction on the Building was completed as a time:DateTimeDescription object.
- **propertyRegistrationID**: identifies the unique identifier of the real estate in the property register of authority where the Building is located.

Construction Status identifies the construction status of a building more precisely. For example, distinctions could be made between new construction and renovation, but also on-site vs off-site (prefabrication) construction. This class is intended to be extended based on existing classification systems. Its values may be defined more precisely with the use of the code:hasCode property.

Building Use identifies the use of a building, e.g. based on occupancy (business, treatment, residential). This class is intended to be extended according to one or more existing classification systems. Its values may be defined more precisely with the use of the code:hasCode property.

A BuildingUnit refers to a part of a Building that is physically separate (i.e., has its own entrance). It should have its own identifier within the building and has a number of other characteristics that may be defined with the following properties:

- **unitInBuilding**: identifies the Building that the Building Unit belongs to (is contained in).
- **loc:hasLocation**: identifies the individual location of the Building Unit, as defined in the Location Pattern.
- **contact:hasAddress**: identifies the address of the Building Unit, as defined in the Contact Pattern.
- **hasRent**: identifies the rental fee for the Building Unit, if any.
- **hasUnitSize**: identifies the size of the Building Unit as an Area, defined in the City Units Pattern.
- **numberOfRooms**: identifies the number of rooms in the Building Unit.
- **numberOfBedrooms**: identifies the number of bedrooms in the Building Unit.
- **floorToCeilingHeight**: identifies the floor to ceiling height in the Building Unit as a Length, defined in the City Units Pattern.

- **buildingFacility**: identifies any Facility(s) contained in the Building Unit. Note that this is distinct from any Facilities that the owners of the unit would have access to (e.g. provided by the building to its occupants) and includes only those that are part of the Building Unit.

Different types (subclasses) of Building such as House, Apartment Building, or Office Building may be defined as required. It is recommended to avoid confusing type of building structure with building use. For example, a “Detached House” is a type of building whereas an “Office” is not. A Building also requires some degree of permanence; a “Duplex” or a “Garage” may be defined as types of buildings, whereas a tent may not. Also note that a Building refers only to the structure, not the surrounding area; an airport terminal is a building, an aircraft hangar is a building, but the entire airport complex is not.

6.5.3 Formalization

Table 7: Key classes in the Building Pattern

Class	Property	Value Restriction
Building	rdfs:subClassOf	infras:InfrastructureElement
	buildingFacility	only Facility
	hasBuildingUnit	only BuildingUnit
	hasBuildingFootprintArea	only cityunits:Area
	hasBuildingFloorArea	only cityunits:Area
	hasBuildingHeight	only cityunits:Length
	numFloors	max 1 xsd:nonNegativeInteger
	contact:hasAddress	max 1 contact:Address
	use	only BuildingUse
	hasConstructionStatus	max 1 ConstructionStatus
	yearOfConstruction	max 1 time:DateTimeDescription and time:day exactly 0 rdfs:Literal and time:month exactly 0 rdfs:Literal and time:year exactly 1 rdfs:Literal
	numBuildingUnits	max 1 xsd:nonNegativeInteger
	propertyRegistrationID	max 1 rdfs:Literal
	floorAreaRatio	max 1 i72:RatioIndicator and (i72:hasD~~d~~enominator only i72:Area) and (i72:hasN~~n~~umerator only i72:Area)
	numAboveGroundFloors	max 1 xsd:nonNegativeInteger
	windowToWallRatio	max 1 i72:RatioIndicator and (i72:hasD~~d~~enominator only i72:Area) and (i72:hasN~~n~~umerator only i72:Area)
	builtAccordingToConstructionCode	only xsd:string
ConstructionStatus	code:hasCode	only code:Code
Facility	loc:hasLocation	max 1 loc:Location
BuildingUnit	rdfs:subClassOf	infras:InfrastructureElement

	unitInBuilding	max 1 Building
	hasRent	only cityunits:MonetaryValue
	~~hasUnitSize~~unitSize	only cityunits:Area
	numberOfRooms	max 1 xsd:~~int~~nonNegativeInteger
	numberOfBedrooms	max 1 xsd:nonNegativeInteger~~int~~
	floorToCeilingHeight	only cityunits:Length
	buildingFacility	only Facility
BuildingUse	code:hasCode	only code:Code

Table 8: Key properties in the Building Pattern

Property	Characteristic	Value (if applicable)
~~hasBuildingFacility~~buildingFacility	subPropertyOf	partwhole:hasComponent
hasBuildingUnit	inverseOf	unitInBuilding
	subPropertyOf	partwhole:hasComponent
unitInBuilding	inverseOf	hasBuildingUnit
	subPropertyOf	partwhole:componentOf

6.6 Land Use Pattern

6.6.1 General

The Land Use Pattern provides the necessary concepts to describe a particular classification(s) applied to some area of land. It introduces the generic concept of a LandUseClassification which may be extended to capture specific classification systems as required. Annex C illustrates possible extensions with classifications from Land-Based Classification Standards (LBCS), Canada Land Use Monitoring Plan (CLUMP), and Agriculture and Agri-Food Canada (AAFC). Such extensions could be used to define multiple such systems such that relationships between classifications in different systems can be inferred. The Land Use Pattern imports the Spatial Location Pattern (defined in ISO/IEC 5087-1); in particular, Land Areas are defined as a type of Location, which may be described as a geometry. They may also be related to other Land Areas (or arbitrary Locations) by the spatial relations such as containment, contact, overlaps, and so on.

6.6.2 Key Classes & Properties

The key classes are formalized in Table 9~~Table 6~~. A Land Area is a way of defining some area in an urban system. It is defined as a type (subclass) of loc:Location and has the following key properties:

- **landUse**: identifies a Land Use Classification that is identified for the Land Area.
- **hasArea**: identifies the size of a Land Area as cityunits:Area quantity.
- **hasPopulation**: identifies the population of a Land Area as an i72:Population quantity.

Land Use Classifications provide a means of describing the land use in a standard way. The Land Use Classification class is intended to serve as the root class that may be extended with various classification systems as required for a particular application. Its values may be defined more precisely with the use of the code:hasCode property.

6.6.3 Formalization

Table 9: Key classes in the Land Use Pattern

Class	Property	Value Restriction
LandArea	rdfs:subClassOf	loc:Location
	landUse	min 1 LandUseClassification
	hasArea	only cityunits:Area
	hasPopulation	only ~~i72~~cityunits:Population
LandUseClassification	code:hasCode	only code:Code

6.7 Person Pattern

6.7.1 General

Persons are a key category in city models. They are the focus of many city services, and it is the combination of decisions of persons in the population that result in changes characteristics of the city such as road congestion. For example, consider a person's decision to change places of employment. Among other things, this change will likely impact their daily travel behaviour. The Person Pattern enables the representation of persons and their attributes of interest. Factors such as a person's age, income, and place of residence are defined as properties of a person.

6.7.2 Key Classes & Properties

The key classes and properties are formalized in Table 10~~Table 8~~ and Table 11, respectively. The core class is Person, which is defined as a type (subclass) of agent:Agent and has the following properties:

- **hasPersonID**: identifies any instances of a Person's PersonId. Examples include a driver's license or passport.
- **name**: identifies the instance of PersonName legally associated with a Person.
- **alias**: identifies any additional instances of PersonName associated with a Person. A Person may have one legal name, but multiple alias names.
- **contact:hasAddress**: identifies any instances of a contact:Address associated with the Person.
- **contact:hasPhoneNumber**: identifies any instances of PhoneNumbers associated with the Person.
- **hasEmail**: specifies zero or more email addresses as xsd:string values.
- **birthdate**: identifies the time:Instant when the person was born.
- **birthplace**: identifies the contact:Address where the person was born.
- **deathdate**: identifies the time:Instant when the person died.
- **deathplace**: identifies the contact:Address where the person died.
- **citizenOf**: specifies one or more Citizenships, each specifying the country (Country) and time interval (time:ProperInterval) the person is a citizen. A person can be a citizen of more than one country and for different time intervals. It is recommended that a Country be defined with using ISO 3166-2 alpha-2 2 letter country code, however different systems may be accommodated with the Code Pattern.
- **parent**: identifies any parents of the Person. Note that we define the parent relation in a general sense (including either the legal or biological relation). This property may be specialized and restricted, for example hasBiologicalMother: exactly 1 Person.

- **spouse**: identifies any spouses of the Person.
- **children**: identifies any children of the Person.
- **sex**: identifies a Person’s sex. A Person is associated with at most one sex. The definition of sex is distinct from that of a person’s gender: “Sex refers to sex assigned at birth. Sex is typically assigned based on a person's reproductive system and other physical characteristics.”[1] Its values may be defined more precisely with the use of the code:hasCode property.
- **genderIdentity**: identifies a Person’s associated Gender identity. The value of this may or may not differ from a person’s sex at birth. Precisely how Gender is defined and instantiated varies based upon context and so may be defined by the user of this standard as appropriate. Its values may be defined more precisely with the use of the code:hasCode property.
- **income**: specifies the Person’s annual income.
- **hasSkill**: identifies the Skills the Person has.
- **hasEducation**: identifies the Education the Person has.

Note the Skill and Education classes are not defined and are left to the application to define. Their values may be defined more precisely with the use of the code:hasCode property.

PersonName represents the parts of a person’s name. It has the following properties:

- **givenName**: identifies a string that is the given or first name of the Person.
- **additionalName**: identifies a string that is the middle or additional names of the Person.
- **familyName**: identifies as single string that is the family or last name of the Person.

PersonID represent a unique identifier for the Person. It has the following properties:

- **genprop:hasIdentifier**: identifies a string that encodes the unique id of the person.
- **hasIDType**: specifies the type of ID, including passport an driversLicense.
- **photoID**: specifies whether the ID contains a photo as a Boolean value.
- **validityPeriod**: a subproperty of time:hasTime, this identifies the time interval during which the ID is valid.
- **issuedBy**: identifies the agent:Agent that issued the ID.

6.7.3 Formalization

Table 10~~8~~: Key classes in the Person Pattern

Class	Property	Value Restriction
Person	rdfs:subClassOf	agent:Agent
	hasPersonID	only PersonId
	n~~N~~ame	max 1 PersonName
	a~~A~~lias	only PersonName
	contact:hasAddress	only contact:Address
	contact:hasTelephone	only contact:PhoneNumber
	email	only xsd:string
	birthDate	max 1 time:Instant

[1] http://www23.statcan.gc.ca/imdb/p3Var.pl?Function=DEC&Id=24101

	birthplace	max 1 contact:Address
	deathDate	max 1 time:Instant
	~~deathPlace~~deathplace	max 1 contact:Address
	citizenOf	only Citizenship
	~~P~~parent	only Person
	spouse~~Spouse~~	only Person
	children	only Person
	sex	max 1 Sex
	hasGenderIdentity	only Gender
	income	only cityunits:MonetaryValue
	hasSkill	only Skill
	hasEducation	only Education
PersonName	givenName	only xsd:string
	additionalName	only xsd:string
	familyName	max 1 xsd:string
PersonID	genprop:hasIdentifier	exactly 1 xsd:string
	hasIDType	only IDType
	photoID	max 1 xsd:boolean
	~~time:hasTime~~validityPeriod	max 1 time:Interval
	issuedBy	max 1 agent:Agent
Citizenship	forCountry	max 1 ~~address~~contact~~s~~:Country
	~~time:hasTime~~validityPeriod	max 1 time:ProperInterval
Sex	code:hasCode	only code:Code
Gender	code:hasCode	only code:Code
IDType	code:hasCode	only code:Code
Skill	code:hasCode	only code:Code
Education	code:hasCode	only code:Code

Table 11: Key properties in the Person Pattern

Property	Characteristic	Value (if applicable)
validityPeriod	subPropertyOf	time:hasTime

6.8 City Resident Pattern

6.8.1 General

NOTE Resident has been defined in a separate pattern because it is a specialization of the Person category defined above. Combining the two concepts into a single category would not be incorrect from a logical perspective, however the categories are separated to define a core Person pattern, independent of any specializations, to support ease of use of the standard.

As different cities have different definitions of who is that city’s Resident, the City Resident Pattern contains the core properties required by each. For example, the city of Toronto’s definition of a city resident includes the concept of owning property or owning or operating a business in the city. For Beijing, nationality is a unique aspect. Central to all the definitions is the concept of residing. Variously referred to as a home or domicile in which the resident spends significant amounts of time; they can own it, rent it, or just stay in it. Legally, “reside means to dwell permanently or continuously. It expresses an idea that a person keeps or returns to a particular dwelling place as his fixed, settled, or legal abode. The meaning of reside implies a continuous arrangement”[2]; reside has both a temporal and spatial dimension.

6.8.2 Key Classes & Properties

The key classes are formalized in Table 12~~Table 10~~. The CityResident class is a subclass of Person. The properties of the CityResident class are used to construct the definition of a resident for a particular city, which may then be defined as a subclass in an extension to this document. These properties are:

Formatted: Font: Cambria

- **hasResidence**: specifies one or more individuals of Residence, where each individual specifies a residence distinguished by city, address and/or time interval. A resident can have more than one residence.
- **owns**: specifies an EntityOwnership where entities owned include include buildings, land areas, or organizations.
- **operates**: specifies an EntityOperation where the entity is an Organization that the resident operates.

All specializations (subclasses) of Resident shall have at least one hasResidence property that identifies where they reside. The remaining properties are optional and their specifications are intended to constrain their use in the context of specializations of Resident. For example, if an optional property is used in the definition of Toronto Resident, then its range is restricted to what is specified in Resident.

In the following definition of CityResident, the properties identified fall into two types: properties that are required in all specializations of Resident, e.g., Toronto Resident, Beijing Resident, and properties that are optional, but if used by a specialization of Resident, have their ranges restricted. A major part of determining whether a person is a resident of a city is the specification of where and when they have resided. The hasResidence property is required and links a CityResident to a Residence. The cardinality of the property is greater than one as over time a person may reside in more than one place/address, in the same city and/or different cities. The Residence class identifies the following key properties:

- **forCity**: identifies the city (City) of the residence.
- **time:hasTime** specifies the time interval during which the residence was held.
- **hasHomeType**: identifies the type of home, such house, apartment, or shelter. Its values may be defined more precisely with the use of the code:hasCode property.
- **contact:hasAddress**: identifies the address of the residence, if any.
- **hasResidentialRelationship**: specifies whether the residence is held by ownership, rental, or other arrangement. Its values may be defined more precisely with the use of the code:hasCode property.

[2] https://definitions.uslegal.com/b/reside/

The temporal intervals of individuals of Residences can used to determine a total or partial ordering of a person's residencies, as a person may reside in more than one place at the same time.

The ControlledEntity and Citizenship classes are necessary to capture the time interval during which an entity is owned or operated, or the person is a citizen of a country. The Controlled Entity class identifies the following key properties:

- **entity**: identifies a thing that is controlled.
- **time:hasTime**: identifies the time interval during which controlled relationship is held for the entity (i.e. during which the person controls it).

The ControlledEntity class is further specialized with the EntityOwnership and EntityOperation subclasses to indicate the entity control more precisely. EntityOwnership indicates an instance of ownership and is specialized with the following properties:

- **entity**: identifies a land area, organization, or building that is owned.
- **percentOwnership**: identifies the percent of the entity that is owned.

The EntityOperation class indicates an instance of entity control via operation. It is specialized with the following property:

- **entity**: indicates an Organization that is controlled. Note in some implementations it might be possible to infer entity operation on the basis of a person's role within an organization.

6.8.3 Formalization

Table 12~~10~~: Key classes in the City Resident Pattern

Class	Property	Value Restriction
CityResident	rdfs:subClassOf	person:Person
	hasResidence	only Residence
	residentOf	only city:JurisdictionalArea
	owns	only EntityOwnership
	operates	only EntityOperation
Residence	forCity	max 1 city:City
	time:hasTime	max 1 time:ProperInterval
	hasHomeType	max 1 HomeType
	contact:hasAddress	max 1 contact:Address
	~~hasTenure~~hasResidentialRelationship	max 1 ~~HousingTenure~~ResidentialRelationship
ControlledEntity	time:hasTime	max 1 time:ProperInterval
	entity	some owl:Thing
EntityOwnership	rdfs:subClassOf	ControlledEntity
	entity	max 1 (landuse:LandArea or org:Organization or building:Building)

	percentOwnership	max 1 xsd:decimal
EntityOperation	rdfs:subClassOf	ControlledEntity
	entity	max 1 org:Organization
HomeType	code:hasCode	only code:Code
~~HousingTenure~~ResidentialRelationship	code:hasCode	only code:Code

6.9 Household Pattern

6.9.1 General

A household is an important concept in many areas of the city. Households are distinct, though often closely related to families and residences. This standard does not provide an explicit representation of a Family, though one could be inferred from the properties of some Persons (as defined in the Person pattern), residence is described in the City Resident Pattern, and household is described in this pattern. The behaviour of a household can be represented by the collective activities of its members, thus membership is a key property of a household.

6.9.2 Key Classes & Properties

The key classes are formalized in Table 13~~Table 11~~. A Household refers to a collection of persons occupying a shared place of residence. Households may or may not be comprised of family members. Different, more precise definitions of Household may be adopted as required for different contexts and applications through extensions to this class. A Household has the following key properties:

- householdOccupies: identifies the cityresident:Residence that it occupies, i.e. the residence that is shared by its members. A Household is defined by this residence, such that if the members move (even collectively), the new residence constitutes a new Household.
- org_s:hasMember: identifies a person:Person who is a member of the Household.
- ~~i72:for_time_interval~~time:hasTime: specifies the time interval the household exists in its current configuration.

Formatted: Indent: Left: 0 cm, Hanging: 1.27 cm

6.9.3 Formalization

Table 13~~11~~: Key classes in the Household Pattern

Class	Property	Value Restriction
Household	householdOccupies	max 1 cityresident:Residence
	org_s:hasMember	only person:Person
	~~i72:for_time_interval~~time:hasTime	~~time:DateTimeInterval~~only time:ProperInterval

6.10 City Organization Pattern

6.10.1 General

An organization is defined broadly as a formal or semi-formal group for which structure and behaviour are defined. Organizations such as schools and businesses are particularly important for cities as they determine regular travel patterns and other activities for much of the population. The Organization Pattern is drawn from the TOVE model of an Organization, originally presented in [7]. It is an extension of the Organization Structure Pattern defined in ISO/IEC 5087-1 that introduces city-specific concepts related to Organizations such as Students and Employees.

6.10.2 Key Classes & Properties

The key classes and properties are formalized in Table 14~~Table 13~~ and Table 15~~Table 12~~, respectively. An Organization is a company or other sort of formal or informal group of individuals in the urban system with some identified structure and behaviour. It is an extension of the org_s:Organization class defined in ISO/IEC 5087-1 to include properties relevant for city services. It includes the following key properties:

- orgAddress: a specialization to contact:hasAddress that is also defined according to the org_s:Site address specified in the Organization Structure Pattern defined in ISO/IEC 5087-1. Specifies the contact:Address associated with the Organization.
- contact:hasTelephone: identifies the contact:PhoneNumber for the Organization.
- o~~hasO~~peratingHours: identifies the OperatingHours of the Organization.
- hasGoal: identifies the Goal(s) of the Organization. This allows for the representation of various groups' responsibilities and intents.
- loc:hasLocation: identifies the loc:associatedLocation(s) of the Organization. This is more general than the contact address, and allows for the representation of any number of spatial locations that the organization is associated with occupying (e.g. office spaces or other facilities).

An Organization may be further classified as a For Profit Organization, Government Organization, or Non Profit Organization. A For Profit Organization has the following additional key properties:

- hasIndustryType: specifies an Industry Type assigned to the organization based on the kind of business conducted. There are different classifications of Industry Types, the inclusion of each is outside the scope of this document. Its values may be defined more precisely with the use of the code:hasCode property.
- hasEstablishment: specifies a Business Establishment where the organization conducts business.

A Government Organization has the following additional key properties:

- hasProgram: specifies a service:Program defined for the organization.
- jurisdiction: specifies the Jurisdictional Area that the organization is responsible for.

A Non Profit Organization has the following additional key properties:

- hasProgram: specifies a service:Program defined for the organization.

A Role is a specialization of org_s:Role as defined in the Organization Structure Pattern in ISO/IEC 5087-1. It extends the class with the following properties:

- hasGoal: specifies a Goal defined for the Role. Any agent in the Role will have this as a Goal.
- hasProcess: specifies an activity:Activity that is performed as part of the Role.
- hasResource: specifies a resource:Resource that is (expected to be) allocated for the Role.

A Goal describes a desired state for an Organization, Organization Agent, or Role. It is defined as a kind of activity:State.

A Business Establishment is a physical location where a For Profit Organization conducts business. It has the following properties:

- loc:hasLocation: specifies the loc:Location of the establishment.
- contact:hasAddress: specifies the contact:Address of the establishment.

An Organization Agent is a member of an organization. It has the following properties:

- org_s:memberOf: specifies the Organization that the Organization Agent is a member of.
- playsRole: specifies a Role that the Organization Agent is assigned in the context of the Organization.
- hasGoal: specifies the Goal of the Organization Agent in the context of the Organization. Goals may be inherited by the agent's Role, or defined directly for the Agent.
- hasEmployment: specifies the details of an agent's Employment, if applicable.

An Employment is a type of organizational membership where the agent receives monetary compensation for their membership in an Organization. It has the following properties:

- employedAs: identifies the occupation that the agent is employed as. An Occupation describes the type of work performed by some employee. Different classifications of occupations may be defined, such as: General Office / Clerical, Manufacturing / Construction / Trades, Professional / Management / Technical, Retail Sales and Service. Enumeration of this categorization is outside of the scope of this document in an extension as required for a given application. Its values may be defined more precisely with the use of the code:hasCode property.~~The inclusion of such classification systems is outside the scope of this document and may be captured in an extension as required for a given application.~~
- hasCompensation~~Pay~~: identifies the monetary compensation received by the agent~~ as a Wage or Salary~~.
- employedBy: identifies the Organization that the agent is employed by.
- hasEmploymentStatus: identifies the Employment Status of the employee. An employment status may be categorized as one of: full-time regular, part-time regular, full-time-work-at-home, part-time-work-at-home. Enumeration of this categorization is outside of the scope of this document in an extension as required for a given application. Its values may be defined more precisely with the use of the code:hasCode property.

Compensation is a generalization of monetary compensation (received for employment). It is further defined as either a Wage or a Salary and has the following property:

- hasPay: identifies the cityunits:MonetaryValue of the Compensation.

Wage is a type of compensation that is defined on an hourly basis. It specializes Compensation with the following properties:

- hourlyPay: identifies the cityunits:MonetaryValue of compensation to be paid per hour.
- overtimePay: identifies an additional rate of pay as cityunits:MonetaryValue to be paid per hour in the case of overtime compensation.

Salary is a type of compensation that is defined on an annual basis. It specialized Compensation with the following property:

- a~~hasA~~nnualPay: identifies the cityunits:MonetaryValue of compensation to be paid for a year.

Operation specifies the time during which an organization regularly conducts business (i.e., its hours of operation). It uses the Recurring Event Pattern (defined in ISO/IEC 5087-1) to define its operation as a recurring event. Operation specializes a Recurring event with the following properties:

- recurringevent:hasDayOfWeek: specifies the day of the week for this recurring instance of Operation.
- hasOpeningTime: specifies the opening time for this instance of Operation.

hasClosingTime: specifies the closing time for this instance of Operation.

6.10.3 Formalization

Table 14~~11~~: Key classes in the City Organization Pattern

Class	Property	Value Restriction
Organization	rdfs:subClassOf	org_s:Organization
	orgAddress	only contact:Address
	contact:hasTelephone	only contact:PhoneNumber
	o~~hasO~~peratingHours	only Operation
	hasGoal	only Goal
	loc:hasLocation	only loc:Location
ForProfitOrganization	rdfs:subClassOf	Organization
	hasIndustryType	only IndustryType
	hasEstablishment	only BusinessEstablishment
GovernmentOrganization	rdfs:subClassOf	Organization
	hasProgram	only service:Program
	jurisdiction	only city:JurisdictionalArea
NonProfitOrganization	rdfs:subClassOf	Organization
	hasProgram	only service:Program
Role	rdfs:subClassOf	org_s:Role
	hasGoal	only Goal
	hasProcess	only activity:Activity
	hasResource	only resource:Resource
Goal	rdfs:subClassOf	activity:State
BusinessEstablishment	loc:hasLocation	max 1 loc:Location
	contact:hasAddress	only contact:Address
OrganizationAgent	rdfs:subClassOf	agent:Agent
	org_s:memberOf	only Organization
	playsRole	only Role
	hasGoal	only Goal
	hasEmployment	only Employment
Employment	employedAs~~rdfs:subClassOf~~	only Occupation~~OrganizationAgent~~
	hasCompensation~~employedAs~~	some Compensation~~some Occupation~~

Formatted: Font: Cambria

Formatted: Highlight

Formatted: Highlight

Formatted: Highlight

Commented [MK1]: Should we put things like Authority back in? They need more of a definition than was included previously.

	employedBy~~hasPay~~	some Organization~~some Wage or Salary~~
	hasEmploymentStatus~~employedBy~~	only EmploymentStatus~~some Organization~~
Compensation	hasPay	max 1 cityunits:MonetaryValue~~Unit~~
Wage	rdfs:subClassOf	Compensation
	hourlyPay	max 1 cityunits:MonetaryValue
	overtimePay	only cityunits:MonetaryValue
Salary	rdfs:subClassOf	Compensation
	a~~hasA~~nnualPay	max 1 cityunits:MonetaryValue
Operation	rdfs:subClassOf	recurringevent:RecurringEvent
	recurringevent:hasDayofWeek	max 1 {friday, monday, saturday, sunday, thursday, tuesday, wednesday}
	hasOpeningTime	max 1 xsd:time
	hasClosingTime	max 1 xsd:time
IndustryType	code:hasCode	only code:Code
EmploymentStatus	code:hasCode	only code:Code
Occupation	code:hasCode	only code:Code

Table 15~~12~~: Key properties in the City Organization pattern.

Property	Characteristic	Value (if applicable)
org_s:hasSite o org_s:siteAddress	rdfs:subPropertyOf	orgAddress
hourlyPay	rdfs:subPropertyOf	hasPay
overtimePay	rdfs:subPropertyOf	hasPay
hasAnnualPay	rdfs:subPropertyOf	hasPay
hasOpeningTime	rdfs:subPropertyOf	recurringevent:start~~beginsRecurring~~Time
hasClosingTime	rdfs:subPropertyOf	recurringevent:end~~sRecurring~~Time

Formatted Table

6.11 City Pattern

6.11.1 General

The City Pattern combines information captured in several patterns, specifically: the land areas occupied by cities, government organizations and administrative areas, and associated bylaws.

6.11.2 Key Classes & Properties

The key classes are formalized in Table 16~~Table 14~~. More general than a City is the concept of a JurisdictionalArea. A JurisdictionalArea is an abstract entity that is characterized not only by its location, but by the objects that occupy it (persons, buildings, etc), the governing body(s) it is subject to, and the activities that occur within it. It has the power to implement administrative or policy decisions (made and enacted on its behalf by its governing body). A JurisdictionalArea has the following properties:

- **genprop:hasName**: The name assigned to the Jurisdictional Area.
- **hasLandArea**: identifies the spatial area occupied by the JurisdictionalArea
- **residentPopulation**: number of residents of the area. Note in many cases that this number may be distinct from the population associated with the Land Area that an Administrative Area occupies. The resident population is determined based on the definition of a resident which is often times more specific than an occupant of an area.
- **hasGovernment**: identifies the GovernmentOrganization that manages the JurisdictionalArea.
- **hasBylaw**: identifies the bylaws in existence in the JurisdictionalArea.
- **administrativeArea**: identifies administrative areas within the JurisdictionalArea. They do not have to be distinct and can be hierarchical.
- **administrativeAreaOf:** identifies an administrative area that the JurisdictionalArea is part of (e.g., a ward is part of a city).

Formatted: Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

A City is a specialization of a Jurisdictional Area that is formally identified as such. It has the following additional property:

- **legalName:** identifies the legal name of the City.

A CityAdministrativeArea is specialization of a Jurisdictional Area that has been identified for use by a City to reflect its unique areas such as districts, wards, neighbourhoods, or prefectures. It specializes the following property:

- **administrativeAreaOf**: identifies a Jurisdictional Areas that this administrative area is part of. A City Administrative Area must be identified as an administrative area of a City.

6.11.3 Formalization

Table 16~~14~~: Key classes in the City Pattern

Class	Property	Value Restriction
JurisdictionalArea	genprop:hasName	max 1 xsd:string
	landuse:hasLandArea	only landuse:LandArea
	residentPopulation	max 1 i72:Population
	hasGovernment	only org:GovernmentOrganization
	hasBylaw	only bylaw:Bylaw
	administrativeArea	only JurisdictionalArea
	administrativeAreaOf	only JurisdictionalArea
City	rdfs:subClassOf	JurisdictionalArea
	legalName	max 1 xsd:string
CityAdministrativeArea	rdfs:subClassOf	JurisdictionalArea
	administrativeAreaOf	some City

Formatted: Highlight

6.12 City Service Pattern

6.12.1 General

NOTE In some contexts (namely, computing) the inclusion of an Application category may also be expected, given that concept of an application is closely tied with that of a service, however the technology-oriented concepts of Application and Service have not been identified as appropriate for this part of the standard. The concept of service defined by the City Service Pattern is more generic, as described below.

Cities provide a variety of services to residents and businesses, including health and social services. The city service pattern, is based on the Canadian Government Reference Model (CGRM). It identifies the following concepts as a basis for understanding the services that governments (Wiseman, 2015):

- **Programs** are major city initiatives that address the needs of their constituents (citizens, clients). They are a mandate to achieve Outcomes by delivering Services. For example, ending homelessness.
- **Services** deliver outputs to clients that contribute to program outcomes. For example, providing shelters for the homeless.
- The **processes** are activities that deliver services. For example, homeless person registration, bed allocation, etc.
- **Resources** are used in carrying out processes. For example, shelter space, beds, and personnel.

6.12.2 Key Classes & Properties

The key classes are formalized in Table 17~~Table 15~~. A Program defines a set of services that focus on a shared set of Outcomes. For example, a "poverty reduction program" can be made up of a set of Services such as mobiles services that provide food and clothing to those that live on the street, and a training service that provides basic skills for those living on the street. A Program has a set of Stakeholders that can contribute or benefit. A program is defined as a type of Activity; it is a high level activity, made up of all of the more granular activities (e.g. Services) involved in it. A Program has the following properties:

- genprop:hasName: identifies the name of the Program. All Programs must have a name.
- genprop:hasDescription: identifies a description of the Program.
- hasService: identifies the Services that make up the Program.
- hasOutcome: identifies the Outcomes that the program is trying to achieve.
- hasContributingStakeholder: identifies the stakeholders that contribute to the Program.
- hasBeneficialStakeholder: identifies the stakeholders that benefit from the Program.
- hasInput: identifies the Inputs to the Program.
- hasOutput: identifies the Outputs of the Program.

A Program is composed of one or more Services. In turn, each Services may be comprised of different activities, Inputs, Outputs and Outcomes. The following are the Service properties:

- activity:hasSubActivity: identifies more specific Activities that comprise the Service.
- hasInput: identifies the Inputs to the Service.
- hasOutput: identifies the Outputs of the Service.
- hasOutcome: identifies the Outcomes that are specific to the Service.
- hasContributingStakeholder: identifies the stakeholders that contribute to the Service.
- hasBeneficialStakeholder: identifies the stakeholders that benefit from the Service.

Outcomes are what stakeholders experience as a result of a Program or Service. Outcomes capture positive and negative, intended and unintended results. Outcome contains the following properties:

- hasIndicator: identifies an i72:Indicator to measure to the Outcome.
- genprop:hasDescription: identifies a general description of the Outcome as a string.

- hasBeneficialStakeholder: identifies the Stakeholder affected.
- fromPerspectiveOf: identifies the Stakeholder who is determining the importance of the Impact.
- hasImportance: specifies how important the Impact is (e.g., high, medium, low). Its values may be more precisely defined using the code:hasCode property.
- intendedImpact: identifies the intended direction of the change (e.g. positive, negative). Its values may be more precisely defined using the code:hasCode property.

Stakeholder is a person or organization that either contributes to or benefits from a Program and/or Service. It has the following properties:

- genprop:hasName: identifies a title for the stakeholder as a string.
- genprop:hasDescription: identifies a general description of the stakeholder as a string.
- hasCatchmentArea: identifies the regional span of the stakeholders as a loc:Location
- hasCatchmentAreaType: identifies the type of regional span of the stakeholders (e.g. local, provincial). Its values may be defined more precisely with the use of the code:hasCode property.
- performs: identifies activities performed by the stakeholder.

Input is a type of resource:TerminalResourceState and defines the resources and the stakeholders that contribute them that are input to an Activity:

- hasContributingStakeholder: The Stakeholders that contribute the resources as input.
- i72:for_time_interval: Specifies the time interval over which the input is used.
- genprop:hasName: Name for the Input.
- genprop:hasDescription: Description for the Input.

Output is a type of resource:TerminalResourceState that provides a quantitative summary of an activity. For example, if the activity is 'we provide training' the output could be 'we trained 50 people to NVQ level 3'. Or a production output could produce 100 meals for the homeless. Basic to these outputs is "what" has been produced and the quantity.

- usedByIndicator: identifies the Indicators that use this Output in determining the value of the Indicator.
- genprop:hasName: identifies a name for the Output.
- genprop:hasDescription: identifies a description for the Output.

In addition, the pattern introduces the hasProgram property to support the reference to a Program by classes in other patterns.

Formatted: No bullets or numbering

6.12.2.1 Formalization

Table 17~~15~~: Key classes in the City Services Pattern

Class	Property	Value Restriction
Program	rdfs:subClassOf	activity:Activity
	genprop:hasName	max 1 xsd:string
	genprop:hasDescription	only xsd:string
	hasService	only Service
	hasOutcome	only Outcome
	hasContributingStakeholder	only Stakeholder
	hasBeneficialStakeholder	only Stakeholder
	hasInput	only Input
	hasOutput	only Output

Service	rdfs:subClassOf	activity:Activity
	genprop:hasName	max 1 xsd:string
	genprop:hasDescription	max 1 xsd:string
	hasInput	only Input
	hasOutput	only Output
	hasOutcome	only Outcome
	hasContributingStakeholder	only Stakeholder
	hasBeneficialStakeholder	only Stakeholder
Outcome	hasIndicator	only i72:Indicator
	genprop:hasDescription	only 1 xsd:string
	hasBeneficialStakeholder	max 1 Stakeholder
	fromPerspectiveOf	max 1 Stakeholder
	hasImportance	max 1 Importance
	intendedImpact	max 1 ImpactDirection
ImpactDirection	code:hasCode	only code:Code
Importance	code:hasCode	only code:Code
Stakeholder	rdfs:subClassOf	(org:Organization or person:Person)
	genprop:hasName	max 1 xsd:string
	genprop:hasDescription	only xsd:string
	hasCatchmentArea	only loc:Location
	hasCatchmentAreaType	only CatchmentAreaType
	performs	some activity:Activity
CatchmentAreaType	code:hasCode	only Code
Input	rdfs:subClassOf	resource:TerminalResourceState
	hasContributingStakeholder	only Stakeholder
	i72:for_time_interval	only time:DateTimeInterval
	genprop:hasName	max 1 xsd:string
	genprop:hasDescription	max 1 xsd:string
Output	rdfs:subClassOf	resource:TerminalResourceState
	usedByIndicator	only i72:Indicator
	genprop:hasName	max 1 xsd:string
	genprop:hasDescription	max 1 xsd:string

6.13 Contract Pattern

6.13.1 General

A contract is a legal document that specifies some agreement(s) between two or more parties. The aim of the Contract Pattern is not to formalize the semantics of all possible involved legal concepts, but rather to enable to representation of the general structure and contents of a particular contract. A Contract is defined as a type of Document and is distinct from the Agreement it specifies.

6.13.2 Key Classes & Properties

The key classes are formalized in Table 18~~Table 17~~. A Contract is a document with the additional following properties:

- specifiesAgreement: identifies the Agreement(s) that are specified by some Contract.
- hasParty: identifies the person(s) and/or organization(s) that are involved in the Contract.
- hasSignatory: identifies the Person(s) responsible for signing the Contract.
- hasContractualElement: identifies the decomposition of a Contract into more precise parts.
- isValidFor: identifies the Interval in time over which the Contract is valid for, if applicable.
- isExecutedOn: identifies the Interval or Instant in time at which the terms in the Contract are executed, if applicable.

A ContractualElement represents a part of a contract. Therefore, it can only exist in the context of some Contract.

- hasContractText: identifies the excerpt of text from the Contract that corresponds to the Contractual Element.

A ContractualElement may be more precisely identified as:

- ConditionPrecedent: a part of the contract that identifies conditions that must be met in order for the contract to take effect.
- ContractualCommitment: a part of the Contract that identifies some Agreement between the parties. It has the following property:
  specifiesAgreement: indicates an Agreement specified by this part of the Contract.
- ContractualDefinition: a part of the Contract that defines key terms that are referred to in the Contract.
- NonBindingTerm: a part of the Contract that is not legally binding.
- Representation: a part of the Contract that specifies some assertions that are taken to be true at the time of the contract and serve to influence a party's decision to enter into the Contract.
- Warranty: a part of the Contract that promises some indemnification if an assertion made in the Contract is false.

A general hasContract property is also defined to associate other objects (e.g. services, employment) with a Contract object.

6.13.3 Formalization

Table 18~~17~~: Key classes in the Contract Pattern

Class	Property	Value Restriction
Contract	specifiesAgreement	some agreement:Agreement
	hasParty	min 2 (person:Person or org:Organization)
	hasSignatory	min 2 person:Person
	hasContractualElement	some ContractualElement

Formatted: Highlight

	isValidFor	only time:Interval
	isExecutedOn	only time:TemporalEntity
ContractualElement	inverse (hasContractualElement)	some Contract
	hasContractText	some xsd:string
ConditionPrecedent	rdfs:subClassOf	ContractualElement
ContractualCommitment	rdfs:subClassOf	ContractualElement
	specifiesAgreement	some agreement:Agreement
ContractualDefinition	rdfs:subClassOf	ContractualElement
NonBindingTerm	rdfs:subClassOf	ContractualElement
Representation	rdfs:subClassOf	ContractualElement
Warranty	rdfs:subClassOf	ContractualElement

Property	Characteristic	Value (if applicable)
hasContract	Range	Contract

The full implementation of the ontology encoding in OWL is available at http://ontology.eil.utoronto.ca/5087/2/Contract.owl

6.14 Bylaw Pattern

6.14.1 General

“Municipal by-laws are public regulatory laws which apply in a certain area. The main difference between a by-law and a law passed by a national/federal or regional/state body is that a by-law is made by a non-sovereign body, which derives its authority from another governing body, and can only be made on a limited range of matters. A local council or municipal government derives its power to pass laws through a law of the national or regional government which specifies what things the town or city may regulate through by-laws. It is therefore a form of delegated legislation.” (Wikipedia, 2020)

“A municipal by-law is no different than any other law of the land, and can be enforced with penalties, challenged in court and must comply with higher levels of law. Municipal bylaws are often enforceable through the public justice system, and offenders can be charged with a criminal offence for breach of a bylaw.” (Alberta, 2017)

The intent of the Bylaw Pattern is to capture the major components of a city bylaw, such as dates, geographic areas of application, penalties, etc. It is not intended to provide a legal semantics with which to codify a particular bylaw. The following three types of bylaws are represented in the Bylaw Pattern (Alberta, 2017):

1. Main bylaws;
2. Amending bylaws, which reflect material changes, in principle or substance, to an existing bylaw; and
3. Revision bylaws, which reflect limited changes to an existing bylaw.

6.14.2 Use Cases

- A commercial organization operating within the city is considering storing or manufacturing hazardous materials. They wish to know what restrictions existing on the handling, storage and manufacturing of hazardous materials. The various search mechanisms are able to precisely find the information as the bylaw explicitly specifies the resources and/or activities that are the focus of the bylaw, as the pattern provides the appropriate tagging of the information.
- A homeowner in the city wishes to remove a tree for their property. They wish to know whether there are any restrictions regarding tree removal. They use their favourite search engine to find the relevant bylaw as the bylaw was tagged using the pattern making it easy for the search engine to find the relevant law amongst the myriad of information found.

6.14.3 Key Classes & Properties

The key classes are formalized in Table 19~~Table 18~~. The core concept of the Bylaw Pattern is Law, of which a Bylaw is a specific type. Its properties decompose a Law into conceptually relevant components that support connection with other city concepts. The following are a Law's properties:

- genprop:name: identifies a short name for the Bylaw, to be used in other descriptions to refer to the Law.
- legislationJurisdiction: identifies the jurisdiction that enacted the Law.
- legislationIdentifier: specifies a unique identifier for the bylawLawThe range is a xsd:string.
- abstract: specifies a brief statement of the Lawpurpose. The range is a xsd:string.
- keywords: identifies keywords used to categorize the Lawfor search purposes. The range is zero of more xsd:string.
- legislationLegalForce: Specifies whether the Law is currently in force, not in force or partially in force.
- hasDefinition: Words or phrases that are defined to have a specific meaning within the context of the Law. The range is restricted to Definition.
- impacts: identifies what is impacted by the Law. Not restricted to any class type.
- legislationDate: Date which the Lawis officially adopted/signed by the legislationJurisdiction city:JurisdictionalArea.
- datePublished: Date the Lawis officially published.
- dateInEffect: Date after which the Lawis in effect.
- expires: Date the Lawexpires.
- hasClause: Clauses that make up the body of the Law. Restricted to Clause.
- hasPenaltyClause: Clauses that specify penalties for not adhering to the Law.
- hasSeveranceClause: Clauses that specify that the bylaw remains valid if any portion of the Law is found to be invalid by a higher jurisdiction.
- hasTransitionClause: Clauses that cover the period during which entities affected by the Lawc an do things to conform to the new conditions.
- hasRepealClause: Clauses that specify all previous Laws that deal with subjects that are addressed in the new bylaw must either be repealed or amended.
- hasSchedule: Schedules that are attached to the Law and are referenced by the Law. A Schedule is part of the Bylaw.

A Bylaw is a type of Law put in place by city. It adds and specializes the following properties:

- legislationJurisdiction: identifies the city:City that enacted the Bylaw.
- legislationType: Type of bylaw chosen from bylawMain, bylawAmending or bylawRevision.
- impacts: who and/or what is impacted by the Bylaw. Restricted to Person, Organization, LandArea, city:JurisdictionalArea, and/or Activities.

A MainBylaw is a subclass of Bylaw and has the following additional properties:

- legislationType: value bylawMain

An AmendingBylaw is a subclass of Bylaw and has the following additional properties:

- legislationChanges: The Bylaw that is being amended.
- legislationType: value bylawAmending

An RevisionBylaw is a subclass of Bylaw and has the following additional properties:

- legislationChanges: The Bylaw that is being amended.
- legislationType: value bylawRevision

A Definition concept that specifies the defined terms used in the Bylaw. It has the following properties:

- genprop:hasName: the formal term being defined. It is an xsd:string.
- genprop:hasDescription: the definition of the genprop:hasName.
- partwhole:properPartOf: the law that the definition is part of.

A Clause is a statement of a rule, provision, requirement, etc. that is part of the body of the Bylaw, or its schedules, penalties, etc. It has the following properties:

- genprop:hasIdentifier: Unique identifier/number for the clause. It is an xsd:string.
- genprop:hasName: title or name of the clause, if any. It is an xsd:string.
- genprop:hasDescription: the content of the clause.
- clauseType: one of (severance or penalty or transition or repeal or schedule or bylaw)
- hasClause: links to any subclauses of this clause.
- partwhole:properPartOf: The part of the l~~Byl~~aw or clause this clause is contained in.

A Schedule is attached to the Bylaw and is part of the Bylaw. It has the following properties:

- genprop:hasIdentifier: Unique identifier/number for the schedule. It is an xsd:string.
- genprop:hasName: title or name of the schedule, if any. It is an xsd:string.
- genprop:hasDescription: an introductory description of the Schedule. It is an xsd:string.
- hasClause: links to all clauses contained in the Schedule.
- partwhole:properPartOf: The l~~Byl~~aw this Schedule is part of.

6.14.4 **Formalization**

Table 19~~18~~: Key classes in the Bylaw Pattern

Class	**Property**	**Value Restriction**
Law	genprop:has~~n~~Name	only xsd:string
	legislationJurisdiction	max 1 city:JurisdictionalArea
	legislationIdentifier	max 1 xsd:string
	abstract	max 1 xsd:string
	keywords	only xsd:string
	legislationLegalForce	only {InForce, NotInForce, PartiallyInForce}
	hasDefinition	only Definition
	legislationDate	max 1 xsd:DateTime
	datePublished	max 1 xsd:DateTime

	dateInEffect	max 1 xsd:DateTime
	expires	max 1 xsd:DateTime
	hasClause	only Clause
	hasPenaltyClause	only Clause
	hasSeveranceClause	only Clause
	hasTransitionClause	only Clause
	hasRepealClause	only Clause
	hasSchedule	only Schedule
Bylaw	rdfs:subClassOf	Law
	legislationJurisdiction	max 1 city:City
	legislationType	only {mainBylaw, amendingBylaw, revisionBylaw}
	impacts	only (person:Person or org:Organization or landuse:LandArea or city:JurisdictionalArea or activity:Activity)
MainBylaw	rdfs:subClassOf	Bylaw
	legislationType	value mainBylaw
AmendingBylaw	rdfs:subClassOf	Bylaw
	legislationType	value amendingBylaw
	legislationChanges	exactly 1 Bylaw
RevisionBylaw	rdfs:subClassOf	Bylaw
	legislationType	value ~~bylawRevision~~revisionBylaw
	legislationChanges	exactly 1 Bylaw
Definition	genprop:hasName	max 1 xsd:string
	genprop:hasDescription	max 1 xsd:string
	partwhole:properPartOf	exactly 1 L~~ Byl~~aw
Clause	genprop:hasIdentifier	max 1 xsd:string
	genprop:hasName	max 1 xsd:string
	genprop:hasDescription	max 1 xsd:string
	clauseType	some {severance,penalty,transition, repeal, schedule, bylaw}
	hasClause	only Clause
	partwhole:properPartOf	exactly 1 {L~~(Byl~~aw or Clause)
Schedule	genprop:hasIdentifier	max 1 xsd:string
	genprop:hasName	max 1 xsd:string

Formatted: Highlight

Formatted: Highlight

Formatted: Highlight

	genprop:hasDescription	max 1 xsd:string
	hasClause	only Clause
	partwhole:properPartOf	exactly 1 L~~Byl~~aw

6.15 Contact Pattern

6.15.1 General

Contact information is relevant for a range of concepts in the city domain. For example, a building can have some associated address, similarly a person or an organization can have some contact address (or phone number, email address, and so on). Note that a person's contact address can differ from their place of residence.

Rather than define these attributes separately for persons, organizations, and so on, it makes sense to capture the general concepts associated with contact information in a separate pattern.

The iContact ontology, accessed at http://ontology.eil.utoronto.ca/icontact.owl, is the basis of the core concepts and properties identified as necessary to define this type of information.

6.15.2 Key Classes & Properties

The key classes are formalized in Table 20~~Table 19~~. Both Address and PhoneNumber classes are designed to accommodate international versions. In addition to drawing from the iContact ontology, the pattern reuses concepts from the Spatial Location Pattern (defined in ISO/IEC 5087-1) to associate an address with a location.

Formatted: Font: Cambria

The Address Class has the following properties:

- hasAddressType: specifies the type of address, e.g., home, work. The values for AddressType may be defined more precisely with the use of the code:hasCode property.
- hasStreetNumber: specifies the number on the street for the address.
- minStreetNumber: a subproperty of hasStreetNumber, specifies the minimum street number of an address in the case that it is defined a street number range.
- maxStreetNumber: a subproperty of hasStreetNumber, specifies the maximum street number of an address in the case that it is defined with a street number range.
- hasStreet: specifies the name of the street for the address.
- hasStreetType: specifies the type of the street for the address, e.g., road, drive. The values for StreetType may be defined more precisely with the use of the code:hasCode property.
- hasStreetDirection: specifies the direction of the street for the address, e.g., east, west. The values for StreetDirection may be defined more precisely with the use of the code:hasCode property.
- hasUnitNumber: specifies the unite or suite number of the address.
- hasPostalBox: specifies the box number for the address.
- hasBuilding: specifies the name of the building for the address.
- hasCitySection: specifies the section of the city for the address.
- hasCity: specifies the name of the city for the address.
- hasProvince: specifies the state or province for the address. Its values may be defined more precisely with the use of the code:hasCode property.
- hasPostalCode: specifies the zip or postalcode for the address.
- hasCountry: specifies the country for the address. The values of Country may be defined more precisely with the use of the code:hasCode property. Use of the ISO 3166-2 alpha-2 2 letter country code is recommended.
- loc:hasLocation: specifies a placename (e.g., geonames.org) and geometry for the address.
- wgs84:lat: Specifies the latitude for the address.

- wgs84:long: Specifies the longitude for the address.

The PhoneNumber class has the following properties:

- hasCountryCode: specifies the country code for the number.
- hasAreaCode: specifies the area code for the number.
- hasPhoneNumber: specifies the remaining digits of the number after the area code.
- hasPhoneType: specifies the type of phone number, e.g., cell phone, home phone, etc. PhoneType values may be defined more precisely with the use of the code:hasCode property.

6.15.3 Formalization

Table 20~~19~~: Key classes in the Contact Pattern

Class	Property	Value Restriction
Address	hasAddressType	only AddressType
	hasStreetNumber	max 1 xsd:~~nonNegativeInteger~~string
	minStreetNumber	max 1 xsd:nonNegativeInteger
	maxStreetNumber	max 1 xsd:nonNegativeInteger
	hasStreet	max 1 xsd:string
	hasStreetType	max 1 StreetType
	hasStreetDirection	max 1 StreetDirection
	hasUnitNumber	max 1 xsd:~~nonNegativeInteger~~string
	hasPostalBox	max 1 xsd:string
	hasBuilding	max 1 xsd:string
	hasCitySection	max 1 xsd:string
	hasCity	max 1 city:City
	hasProvince	max 1 State
	hasPostalCode	max 1 xsd:string
	hasCountry	max 1 Country
	loc:hasLocation	max 1 loc:Location
	wgs84:lat	max 1 xsd:decimal
	wgs84:long	max 1 xsd:decimal
AddressType	code:hasCode	only code:Code
StreetDirection	code:hasCode	only code:Code
StreetType	code:hasCode	only code:Code
State	code:hasCode	only code:Code
Country	code:hasCode	only code:Code
PhoneNumber	hasCountryCode	max 1 xsd:nonNegativeInteger

Commented [MK3]: Should these be string values as well? (As with street number?)

Commented [MK4]: Suggest to change this to "buildingName" to avoid confusion? "hasBuilding" seems like it should be associated with a Building object

	hasAreaCode	max 1 xsd:nonNegativeInteger
	hasPhoneNumber	max 1 xsd:nonNegativeInteger
	hasPhoneType	max 1 PhoneType
PhoneType	code:hasCode	only code:Code

6.16 **Sensors Pattern**

6.16.1 **General**

NOTE Sensors are included in the city-level of this standard due to their broad relevance across city services. Actuators and other categories of knowledge specific to control systems are not currently identified as relevant for multiple services and so will likely be defined in standards at the Service Level, if at all.

The Sensors Pattern is included in this document due to the importance of data collection for all manner of city services. Data collection efforts take various forms – whether through manual canvassing or the use of sensors. With a growing access to the Internet of Things, data from available sensors will continue to expand, likely to include observations about persons, vehicles, and so on. It is important to not only capture the collected data, but the source of the observations.

The representation of sensors shall conform to the ontology specified in The W3C Recommendation "The SSN (Semantic Sensor Network) Ontology" [6]. It is included in its entirety with the prefix 'ssn'. The W3C standard ontology for sensors and their observations, the SSN (Semantic Sensor Network) Ontology, accessed from http://www.w3.org/ns/ssn/ shall be used in the context of this document.

Annex A
(informative)

Example Use Case – Epidemic Tracking

Data and distribution map of confirmed cases, deaths, etc. Data are from each communities and districts in the city and summarized by city government and released to the public, including the number of new confirmed cases, asymptomatic infected cases, cumulative confirmed cases, cumulative cured cases, cumulative deaths, regional risk levels, etc. The classes required for this use case are formalized in Table 21~~Table 20~~. Example data are presented in Table 22~~Table 21~~.

Competency questions:

1. Currently, how many cumulative confirmed cases of COVID-19 are there in the city?

2. How many deaths of COVID-19 were in the city in 2020-02-08?

3. Currently, what is the COVID-19 infection rate (number of confirmed cases in the district/population of the district) in Lixia district in this city?

4. Which districts do the new cases in the city in 2020-02-08 come from?

5. What is the current population of the city?

6. What is the cure rate (number of cure cases in the district/ number of confirmed cases in the district) in Lixia district in this city in 2020?

7. How many districts are there in the city?

8. What is the average daily confirmed cases of COVID-19 in the city From February 6th to February 12th in 2020?

9. What is the current number of confirmed cases (have infected but have not cured and not dead yet) of COVID-19 in the city?

10. What are the current medium-risk districts in the city?

11. Which day in the city had the most confirmed new cases of COVID-19 in the first three months in 2020?

12. What color is this person's X health code?

Concepts & Properties:

Table 21~~20~~: Classes required for the Epidemic Tracking use case

Class	Property	Value Restriction
cov:City	rdfs:subClassOf	city:City
	city:legalName	exactly 1 xsd:string
	partwhole:hasProperPart	only city:CityAdministrativeArea

	loc:hasLocation	exactly 1 loc:Location
	city:hasPopulationSize	exactly 1 (i72:Quantity and i72:Unit_of_measure value i72:population_cardinality_unit)
	cov:hasOrganization	only org:Organization
cov:District	rdfs:subClassOf	city:CityAdministrativeArea
	genprop:hasName	exactly 1 xsd:string
	partwhole:properPartOf	only (city:JurisdictionalArea)
	loc:hasLocation	exactly 1 loc:Location
	city:hasPopulationSize	exactly 1 (i72:Quantity and i72:unit_of_measure value i72:population_cardinality_unit)
	cov:hasRiskLevel	exactly 1 of {low, medium, high}
	cov:hasCaseReport	only cov:CaseReport
cov:Person	rdfs:subClassOf	person:Person
	person:hasPersonID	only person:PersonID
	person:personHasBirthDate	exactly 1 time:Instant
	person:personHasDeathDate	max 1 time:Instant
	person:hasAge	exactly 1 xsd:positiveInteger
	person:hasSex	exactly 1 person:Sex
	person:parent	only person:Person
	person:spouse	only person:Person
	person:children	only person:Person
	person:hasIncome	only cityunits:MonetaryValue
	contact:address	some contact:PostalAddress
	cityresident:hasResidence	only cityresident:Residence
	cov:hasHealthCode	exactly 1 of {red, yellow, green}
	foaf:firstName	only xsd:string
	foaf:lastName	only xsd:string
cov:Patient	cov:patientOf	exactly 1 cov:Person
	cov:hasDisease	only cov:DiseaseDescription
	cov:hospitalized	only cov:Hospital
	cov:hasConfirmTime	exactly 1 time:Instant
	cov:hasSickPeriod	exactly 1 time:Interval
	cov:hasCureTime	max 1 time:Instant
	cov:hasDeathTime	max 1 time:Instant

	loc:hasLocation	only loc:Location
cov:CaseReport	cov:reportedBy	only cov:Hospital
	cov:reportedTo	only cov:DistrictGovernment
	cov:forPatient	only cov:Patient
	cov:hasReportDate	exactly 1 time:Instant
	cov:hasUpdateInterval	exactly 1 time:Interval
	loc:hasLocation	only loc:Location
org:Organization	rdfs:subClassOf	change:Manifestation
	contact:address	some contact:PostalAddress
	org:hasGoal	only org:Goal
	org:consistsOf	only city:CityDivision
	loc:hasLocation	only loc:Location
	org_s:hasMember	only city:Employee
cov:MunicipalGovernment	rdfs:subClassOf	city:GovernmentOrganization
	cov:distribute	some cov:MedicalResource
	cov:publishReport	some cov:CaseReport
	cov:publishPolicy	some cov:PreventionPolicy
	partwhole:hasProperPart	some cov:DistrictGovernment
	loc:hasLocation	only loc:Location
cov:DistrictGovernment	rdfs:subClassOf	city:GovernmentOrganization
	cov:summarizeCaseReportTo	only cov:MunicipalGovernment
	loc:hasLocation	only loc:Location
cov:MedicalResource	rdfs:subClassOf	resource:Resource
	genprop:hasDescription	exactly 1 xsd:string
	resource:hasCapacity	only cityunits:CapacitySize
cov:DistributionSpecification	cov:distributeContent	some cov:MedicalResource
	cov:distributedBy	only cov:MunicipalGovernment
	resource:usedBy	some org:Organization
	cov:distributeDate	exactly 1 time:Instant
	cov:distributeNumber	exactly 1 i72:Measure
cov:PreventionPolicy	rdfs:subClassOf	bylaw:Bylaw
	cov:publishedBy	some city:GovernmentOrganization
	bylaw:datePublished	exactly 1 xsd:DateTime
	cov:associatedDisease	only cov:Disease

	bylaw:legislationJurisdiction	exactly 1 city:City
cov:Disease	genprop:hasDescription	exactly 1 xsd:string
	genprop:hasName	exactly 1 xsd:string
	cov:associatedPolicy	some cov:PreventionPolicy
	cov:associatedCase	only cov:DiseaseDescription
cov:DiseaseDescriptio n	cov:hasSymptom	only xsd:string
	cov:forDisease	only cov:Disease
	cov:hasDiseaseStage	only xsd:string
	cov:hasTest	only cov:Test
	cov:forPatient	exactly 1 cov:Patient
	cov:DescriptionTime	exactly 1 time:Instant
	cov:hasCurrentState	exactly 1 of {Confirmed,Cured,Death}
cov:Test	cov:associatedDisease	only cov:Disease
	cov:hasTestDate	exactly 1 xsd:DateTime
	cov:operatedIn	exactly 1 cov:Hospital
	cov:hasResult	exactly 1 xsd:string
	cov:forPatient	exactly 1 cov:Patient
cov:Hospital	rdfs:subClassOf	org:Organization
	cov:hasBuilding	only building:Building
	resource:hasCapacity	only cityunits:CapacitySize
	contact:address	some contact:PostalAddress
	cov:hasPatient	only cov:Patient
	service:hasService	some cov:MedicalService
	org_s:hasMember	only org:Employee
	cov:hasType	exactly 1 of {ordinary, designated}
cov:MedicalService	rdfs:subClassOf	service:Service
	cov:providedBy	some cov:Hospital
5087-2:Employee	rdfs:subClassOf	person:Person
	org:employedAs	some org:Occupation
	org:hasPay	some org:Wage or org:Salary
	org:worksAt	some loc:Location
	org:hasEmploymentStatus	only org:EmploymentStatus
cov:MedicalWorker	rdfs:subClassOf	org:Occupation
	org:worksAt	some cov:Hospital

	cov:treat	some cov:Patient

Example:

Table 22~~21~~: Example data for the Epidemic Tracking use case

instance	property	value
Jinan	rdf:type	cov:City
	person:legalName	“Jinan” ^^xsd:string
	partwhole:hasProperPart	Lixia District, Shizhong District ......
	loc:hasLocation	jinanLocation
	city:hasPopulationSize	jinanPopulationSize
	cov:hasOrganization	Jinan Municipal Government, Lixia District Government, Shizhong District Government, Shandong Provincial Chest Hospital, Jinan Infectious Disease Hospital, Shandong Provincial Qianfoshan Hospital ......
jinanPopulationSize	rdf:type	i7s:Measure
	i72:unit_of_measure	i72:population_cardinality_unit
	i72:numerical_value	“7000000”^^xsd:integer
jinanLocation	rdf:type	geo:Point
	geo:lat	“36.40”^^xsd:decimal
	geo:long	“117.00”^^xsd:decimal
LixiaDistrict	rdf:type	city:CityAdministrativeArea
	genprop:hasName	“Lixia District”^^xsd:string
	partwhole:properPartOf	Jinan
	loc:hasLocation	lixiaLocation
	city:hasPopulationSize	lixiaPopulationSize
	cov:hasRiskLevel	medium
	cov:hasCaseReport	CaseReport_200208 CaseReport_200210 ......
lixiaPopulationSize	rdf:type	i72:Measure

	i72:unit_of_measure	i72:population_cardinality_unit
	i72:numerical_value	“1000000”^^xsd:integer
lixiaLocation	rdf:type	geo:Point
	geo:lat	“36.67”^^xsd:decimal
	geo:long	“117.08”^^xsd:decimal
ShizhongDistrict	rdf:type	city: CityAdministrativeArea
	genprop:hasName	“Shizhong District”^^xsd:string
	partwhole:properPartOf	Jinan
	loc:hasLocation	shizhongLocation
	city:hasPopulationSize	shizhongPopulationSize
	cov:hasRiskLevel	low
shizhongPopulationSize	rdf:type	i72:Measure
	i72:unit_of_measure	i72:population_cardinality_unit
	i72:numerical_value	“600000”^^xsd:integer
shizhongLocation	rdf:type	geo:Point
	geo:lat	“35.40”^^xsd:decimal
	geo:long	“116.58”^^xsd:decimal
LiLi	rdf:type	cov:Person
	foaf:firstName	“Li”^^xsd:string
	foaf:lastName	“Li”^^xsd:string
	person:hasPersonID	“001”^^xsd:string
	person:personHasBirthDate	“1996-11-06”^^xsd:date
	person:hasAge	“P24Y”^^xsd:duration
	person:hasSex	Female
	cov:hasHealthCode	green
	contact:address	“No.75, Weiwu Road, Shizhong District, Jinan City”^^xsd:string
	rdf:type	5087-2:Employee
	org:employedAs	MedicalWorker_LiLi
	org:worksAt	Jinan Infectious Disease Hospital
TaoZhou	rdf:type	cov:Person
	foaf:firstName	“Tao”^^xsd:string
	foaf:lastName	“Zhou”^^xsd:string
	person:hasPersonID	“002”^^xsd:string
	person:personHasBirthDate	“1988-09-28”^^xsd:date

	person:hasAge	““P32Y”^^xsd:duration
	person:hasSex	Male
	cov:hasHealthCode	green
	contact:address	“No.19, Keyuan Road, Lixia District, Jinan City”^^xsd:string
	rdf:type	org:Employee
	org:employedAs	MedicalWorker_TaoZhou
	org:worksAt	Shandong Provincial Chest Hospital
YingWu	rdf:type	cov:Person
	foaf:firstName	“Ying”^^xsd:integer
	foaf:lastName	“Wu”^^xsd:integer
	person:hasPersonID	“003”^^xsd:string
	person:personHasBirthDate	“2012-05-30”^^xsd:date
	person:hasAge	“P8Y”^^xsd:duration
	person:hasSex	Female
	cov:hasHealthCode	red
	contact:address	“Yaotou Road, Lixia District, Jinan City”^^xsd:string
	rdf:type	cov:Patient
	cov:patientOf	YingWu
	cov:hasDisease	DiseaseDescription_YingWu_Day 1
	cov:hospitalized	Jinan Infectious Disease Hospital
	cov:hasConfirmTime	“2020-2-10”^^xsd:date
	cov:hasSickPeriod	“P23D”^^xsd:duration
	person:parent	QizhengWu
	loc:hasLocation	Jinan
	loc:hasLocation	Lixia District
QizhengWu	rdf:type	cov:Person
	foaf:firstName	“Qizheng”^^xsd:string
	foaf:lastName	“Wu”^^xsd:string
	person:hasPersonID	“004”^^xsd:string
	person:personHasBirthDate	“1984-07-15”^^xsd:date
	person:hasAge	“P36Y”^^xsd:duration
	person:hasSex	Male

	cov:hasHealthCode	yellow
	contact:address	“Yaotou Road, Lixia District, Jinan City”^^xsd:string
	rdf:type	cov:Patient
	cov:patientOf	QizhengWu
	cov:hasDisease	DiseaseDescription_QizhengWu_Day1 ...... DiseaseDescription_QizhengWu_Day14
	cov:hospitalized	Shandong Provincial Chest Hospital
	cov:hasConfirmTime	“2020-02-08”^^xsd:date
	cov:hasSickPeriod	“P14D”^^xsd:duration
	cov:hasCureTime	“2020-02-21”^^xsd:date
	person:children	YingWu
	loc:hasLocation	Jinan
	loc:hasLocation	Lixia District
CaseReport_200208	rdf:type	cov:CaseReport
	cov:reportedBy	Jinan Infectious Disease Hospital
	cov:reportedTo	Lixia District Government
	cov:forPatient	YingWu
	cov:hasReportDate	“2020-02-08”^^xsd:date
	loc:hasLocation	Lixia District
CaseReport_200210	rdf:type	cov:CaseReport
	cov:reportedBy	Shandong Provincial Chest Hospital
	cov:reportedTo	Lixia District Government
	cov:forPatient	QizhengWu
	cov:hasReportDate	“2020-02-10”^^xsd:date
	loc:hasLocation	Lixia District
JinanMunicipalGovernment	rdf:type	cov:MunicipalGovernment
	genprop:hasName	“Jinan Municipal Government”^^xsd:string
	loc:hasLocation	Jinan
	cov:distribute	Mask, PPE,

		ECMO, Vaccine ......
	cov:publishReport	CaseReport_200208 CaseReport_200210 ......
	cov:publishPolicy	Covid-19_PreventionPolicy
	partwhole:hasProperPart	Lixia District Government ShiZhong District Government ......
LixiaDistrictGovernment	rdf:type	cov:DistrictGovernment
	genprop:hasName	“Lixia District Government”^^xsd:string
	loc:hasLocation	Lixia District
	cov:summarizeCaseReportTo	Jinan Municipal Government
ShizhongDistrictGovernment	rdf:type	cov:DistrictGovernment
	genprop:hasName	“Shizhong District Government”^^xsd:string
	loc:hasLocation	Shizhong District
	cov:summarizeCaseReportTo	Jinan Municipal Government
ECMO	rdf:type	MedicalResource
	genprop:hasDescription	“Extracorporeal membrane oxygenation machine.”^^xsd:string
	resource:hasCapacity	“20”^^xsd:integer
PPE	rdf:type	MedicalResource
	genprop:hasDescription	“Personal Protective Equipment”^^xsd:string
	resource:hasCapacity	“100000”^^xsd:integer
DistributionSpecification_ECMO_1	rdf:type	DistributionSpecification
	cov:distributeContent	ECMO
	cov:distributedBy	Lixia District Government
	resource:usedBy	Shandong Provincial Chest Hospital
	cov:distributeDate	“2020-02-01”^^xsd:date
	cov:distributeNumber	“3”^^xsd:integer
DistributionSpecification_PPE_1	rdf:type	DistributionSpecification
	cov:distributeContent	PPE

	cov:distributedBy	Lixia District Government
	resource:usedBy	Shandong Provincial Qianfoshan Hospital
	cov:distributeDate	“2020-02-01”^^xsd:date
	cov:distributeNumber	“1000”^^xsd:integer
Covid-19_PreventionPolicy	rdf:type	cov:PreventionPolicy
	cov:publishedBy	Jinan Municipal Government
	bylaw:datePublished	“2020-03-01”^^xsd:date
	cov:associatedDisease	“Covid-19”^^xsd:string
	bylaw:legislationJurisdiction	Jinan
	bylaw:abstract	“Those returning to Jinan from outside the province should take the initiative to register with the community where they live, and consciously and strictly implement isolation medical observation. The observation period shall not be less than 14 days from the date of return.”^^xsd:string
Covid-19	rdf:type	cov:Disease
	genprop:hasDescription	“COVID-19 is an infectious disease caused by a newly discovered coronavirus......”^^xsd:string
	cov:associatedPolicy	Covid-19_PreventionPolicy
	cov:associatedCase	DiseaseDescription_YingWu_Day1, DiseaseDescription_QizhengWu_Day1, DiseaseDescription_QizhengWu_Day1 ......
DiseaseDescription_YingWu_Day1	rdf:type	cov:DiseaseDescription
	cov:forDisease	Covid-19
	cov:hasSymptom	“Fever, fatigue,dry cough”^^xsd:string
	cov:hasDiseaseStage	Symptomatic
	cov:hasTest	Test_YingWu_1
	cov:forPatient	YingWu
	cov:DescriptionTime	“2020-02-10”^^xsd:date
	cov:hasCurrentState	“Confirmed”^^xsd:string

DiseaseDescription_QizhengWu_Day1	rdf:type	cov:DiseaseDescription
	cov:forDisease	Covid-19
	cov:hasSymptom	“None”^^xsd:string
	cov:hasDiseaseStage	Asymptomatic
	cov:hasTest	Test_QizhengWu_1
	cov:forPatient	QizhengWu
	cov:DescriptionTime	“2020-02-08”^^xsd:date
	cov:hasCurrentState	“Confirmed”^^xsd:string
DiseaseDescription_QizhengWu_Day14	rdf:type	cov:DiseaseDescription
	cov:forDisease	Covid-19
	cov:hasSymptom	“None”^^xsd:string
	cov:hasDiseaseStage	Asymptomatic
	cov:hasTest	Test_QizhengWu_2
	cov:forPatient	QizhengWu
	cov:DescriptionTime	“2020-02-21”^^xsd:date
	cov:hasCurrentState	“Cured”^^xsd:string
Test_YingWu_1	rdf:type	cov:Test
	cov:associatedDisease	Covid-19
	cov:hasTestDate	“2020-02-10”^^xsd:date
	cov:operatedIn	Jinan Infectious Disease Hospital
	cov:hasResult	Positive
	cov:forPatient	YingWu
Test_QizhengWu_1	rdf:type	cov:Test
	cov:associatedDisease	Covid-19
	cov:hasTestDate	“2020-02-08”^^xsd:date
	cov:operatedIn	Shandong Provincial Chest Hospital
	cov:hasResult	Positive
	cov:forPatient	QizhengWu
Test_QizhengWu_2	rdf:type	cov:Test
	cov:associatedDisease	Covid-19
	cov:hasTestDate	“2020-02-21”^^xsd:date
	cov:operatedIn	Shandong Provincial Chest Hospital
	cov:hasResult	Negative

	cov:forPatient	QizhengWu
JinanInfectiousDisease Hospital	rdf:type	cov:Hospital
	genprop:hasName	“Jinan Infectious Disease Hospital”^^xsd:string
	cov:hasPatient	YingWu
	service:hasService	NAT
	org_s:hasMember	LiLi
	cov:hasType	designated
	contact:address	“No.22029, Jingshi Road, Shizhong District, Jinan City”^^xsd:string
ShandongProvincialChestHospital	rdf:type	cov:Hospital
	genprop:hasName	“Shandong Provincial Chest Hospital”^^xsd:string
	cov:hasPatient	QizhengWu
	service:hasService	NAT
	org_s:hasMember	TaoZhou
	cov:hasType	designated
	contact:address	“No.46, Lishan Road, Lixia District, Jinan City”^^xsd:string
ShandongProvincialQianfoshanHospital	rdf:type	cov:Hospital
	genprop:hasName	“Shandong Provincial Qianfoshan Hospital”^^xsd:string
	cov:hasType	ordinary
	contact:address	“No.16766, Jingshi Road, Lixia District, Jinan City”^^xsd:string
NAT	rdf:type	cov:MedicalService
	genprop:hasName	“nucleic acid testing”^^xsd:string
MedicalWorker_LiLi	org:worksAt	Jinan Infectious Disease Hospital
	cov:treat	YingWu
MedicalWorker_TaoZhou	org:worksAt	Shandong Provincial Chest Hospital
	cov:treat	QizhengWu

Competency questions with answers:

```
PREFIX cov: <http://www.example.com/ontologies/covid19ontology/>
PREFIX co: <http://purl.org/ontology/co/core#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xml: <http://www.w3.org/XML/1998/namespace>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX i72: <http://ontology.eil.utoronto.ca/ISO21972/iso21972#>
PREFIX schema: <http://schema.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX fo: <http://www.w3.org/1999/XSL/Format#>
```

1. Currently, how many cumulative confirmed cases of COVID-19 are there in the city?

```
SELECT (count(distinct ?patient) as ?count)
WHERE {
?patient cov:patientOf ?person.
?patient loc:hasLocation cov:Jinan.
?patient cov:hasDisease ?disdescrip.
?disdescrip cov:forDisease cov:Covid-19.
?patient cov:hasConfirmTime ?time.
FILTER(?time>="2020-01-01T00:00:00"^^xsd:dateTime && ?time<=NOW())
}
```

Answer:

2

2. How many deaths of COVID-19 were in the city in 2020-02-08?

```
SELECT (count(distinct ?patient) as ?deathCount)
WHERE {
?patient cov:patientOf ?person.
?patient loc:hasLocation cov:Jinan.
?patient cov:hasDisease ?disdescrip.
?disdescrip cov:forDisease cov:Covid-19.
?patient cov:hasDeathTime ?time.
FILTER (?time>="2020-02-08T00:00:00"^^xsd:dateTime &&  ?time<="2020-02-08T23:59:59"^^xsd:dateTime).
}
```

Answer:

0

3. Currently, what is the COVID-19 infection rate (number of confirmed cases in the district/population of the district) in Lixia district in this city?

```
SELECT ((count(distinct ?patient))/MIN(?population))
WHERE {
cov:LixiaDistrict partwhole:properPartOf cov:Jinan.
cov:LixiaDistrict city:hasPopulationSize ?ps.
?ps i72:numerical_value ?population.
  {SELECT ?patient
  WHERE {
    ?patient cov:patientOf ?person.
    ?patient cov:hasDisease ?disdescrip.
    ?disdescrip cov:forDisease cov:Covid-19.
    ?patient loc:hasLocation cov:LixiaDistrict.
    ?patient cov:hasConfirmTime ?time.
    FILTER(?time>="2020-01-01T00:00:00"^^xsd:dateTime && ?time<=NOW())
     }
  }
}
```

Answer:

0.000002

4. Which districts do the new cases in the city in 2020-02-08 come from?

```
SELECT distinct ?patient ?district
WHERE {
?patient loc:hasLocation cov:Jinan.
?district partwhole:properPartOf cov:Jinan.
?patient loc:hasLocation ?district.
?patient cov:patientOf ?person.
?patient cov:hasDisease ?disdescrip.
?disdescrip cov:forDisease cov:Covid-19.
?patient cov:hasConfirmTime ?time.
FILTER (?time>="2020-02-08T00:00:00"^^xsd:dateTime &&  ?time<="2020-02-08T23:59:59"^^xsd:dateTime).
}
```

Answer:

patient	**district**
cov:QizhengWu	cov:LixiaDistrict

5. What is the current population of the city?

```
SELECT  ?population
WHERE {
cov:Jinan city:hasPopulationSize ?ps.
?ps i72:numerical_value ?population.
}
```

Answer:

7000000

6. What is the cure rate (number of cure cases in the district/ number of confirmed cases in the district) in Lixia district in this city in 2020?

```
SELECT  ((count(distinct ?curedpatient))/(count(distinct ?allpatient)))
WHERE {
?curedpatient loc:hasLocation cov:LixiaDistrict.
?curedpatient cov:patientOf ?person.
  {
    SELECT ?curedpatient (MAX(?time) AS ?latesttime)
    WHERE {
      ?curedpatient cov:hasDisease ?disdescrip.
      ?disdescrip cov:forDisease cov:Covid-19.
      ?disdescrip cov:DescriptionTime ?time.
    } GROUP BY ?curedpatient
  }
  ?curedpatient cov:hasDisease ?disdescrip.
  ?disdescrip cov:forDisease cov:Covid-19.
  ?disdescrip cov:DescriptionTime ?time.
  ?disdescrip cov:hasCurrentState ?state.
  FILTER(?time=?latesttime && ?state="Cured").
  FILTER(?time>="2020-01-01T00:00:00"^^xsd:dateTime &&  ?time<="2020-12-31T23:59:59"^^xsd:dateTime).

  {SELECT  ?allpatient
   WHERE {
    ?allpatient cov:patientOf ?person.
    ?allpatient cov:hasDisease ?disdescrip.
    ?disdescrip cov:forDisease cov:Covid-19.
    ?allpatient loc:hasLocation cov:LixiaDistrict.
    ?allpatient cov:hasConfirmTime ?time.
    FILTER (?time>="2020-01-01T00:00:00"^^xsd:dateTime &&  ?time<="2020-12-31T23:59:59"^^xsd:dateTime).
  }
}
}
```

Answer:

0.5

7. How many districts are there in the city?

```
SELECT  count(?district)
WHERE {
cov:Jinan partwhole:hasProperPart ?district.
}
```

Answer:

2

8. What is the average daily confirmed cases of COVID-19 in the city From February 6th to February 12th in 2020?

```
SELECT (count(distinct ?patient)/7)
WHERE {
?patient loc:hasLocation cov:Jinan.
?patient cov:patientOf ?person.
?patient cov:hasDisease ?disdescrip.
?disdescrip cov:forDisease cov:Covid-19.
?patient cov:hasConfirmTime ?time.
FILTER (?time>="2020-02-06T00:00:00"^^xsd:dateTime &&  ?time<="2020-02-12T23:59:59"^^xsd:dateTime).
}
```

Answer:

0.285714285714285714285714285714

9. What is the current number of confirmed cases (have infected but have not cured and not dead yet) of COVID-19 in the city?

```
SELECT  (count(distinct ?patient) as ?count)
WHERE {
?patient loc:hasLocation cov:Jinan.
?patient cov:patientOf ?person.
  {
    SELECT ?patient (MAX(?time) AS ?latesttime)
    WHERE {
      ?patient cov:hasDisease ?disdescrip.
      ?disdescrip cov:forDisease cov:Covid-19.
      ?disdescrip cov:DescriptionTime ?time.
    } GROUP BY ?patient
  }
  ?patient cov:hasDisease ?disdescrip.
  ?disdescrip cov:forDisease cov:Covid-19.
  ?disdescrip cov:DescriptionTime ?time.
  ?disdescrip cov:hasCurrentState ?state
  FILTER(?time=?latesttime && ?state="Confirmed")
}
```

Answer:

1

10. What are the current medium-risk districts in the city?

```
SELECT  ?district
WHERE {
cov:Jinan partwhole:hasProperPart ?district.
?district cov:hasRiskLevel ?rl.
FILTER (?rl="medium")
}
```

Answer:

cov:LixiaDistrict

11. Which day in the city had the most confirmed new cases of COVID-19 in the first three months in 2020?

```
SELECT (COUNT(distinct ?patient) AS ?count) ?time
WHERE {
?patient loc:hasLocation cov:Jinan.
?patient cov:patientOf ?person.
?patient cov:hasDisease ?disdescrip.
?disdescrip cov:forDisease cov:Covid-19.
?patient cov:hasConfirmTime ?time.
FILTER (?time>="2020-01-01T00:00:00"^^xsd:dateTime &&  ?time<="2020-03-31T23:59:59"^^xsd:dateTime).
}
GROUP BY ?time
ORDER BY DESC(?count)
```

Answer:

count	**time**
1	2020-02-08T00:00:00
1	2020-02-10T00:00:00

12. What color is this person's X health code?

```
SELECT  ?healthcode
WHERE {
cov:LiLi cov:hasHealthCode ?healthcode.
}
```

Answer:

"green"

Annex B
(informative)

Relationship to existing standards

B.1 CityGML

Scope

CityGML is an XML-based standard for representing 3D city models. Target application areas identified include: “urban and landscape planning; architectural design; tourist and leisure activities; 3D cadastres; environmental simulations; mobile telecommunications; disaster management; homeland security; vehicle and pedestrian navigation; training simulators and mobile robotics.” It is intended to capture the data necessary to generate 3D portrayals in appropriate tools, providing not only geometry but data regarding surface characteristics and objects of interest (e.g. buildings, water bodies). Data mappings between related terms are identified and summarized in Table 23~~Table 22~~.

Relevance

Although the purpose of CityGML is to support the capture and exchange of 3D city models, the thematic extension modules of CityGML capture information on city objects and so are relevant to 5087-2.

Data Mappings

Table 23~~22~~: Relationship between related terms in 5087-2 and CityGML

5087-2 Term	Corresponding CityGML Term	Relationship
contact:Address	AddressType	In CityGML, addresses are defined per the extensible address language (OASIS), xalAddress. Correspondences with this schema should be identified. CityGML also associates a geometry(s) with an Address object via the multiPoint property. This could be a useful inclusion in 5087-2.
building:Building	AbstractBuilding	In contrast to the 5087:Building definition, the CityGML representation focuses on physical aspects (e.g. roof and wall surface types) but also includes some relevant attributes such as height, storeys above ground, usage, and construction date. It could be worthwhile to adopt some of these attributes in 5087-2.
landuse: LandArea	LandUse	Land Use objects are used to identify physical and biological uses/characteristics of some area. "LandUse objects in CityGML ... can be employed to represent, parcels, spatial planning objects, recreational objects and objects describing the physical characteristics of an area, in 3D (e.g. wetlands)." 5087-2 defines LandArea in a similar way.

		5087-2 does not currently distinguish between usage and function, this could be something to incorporate in the future <> function attribute
LandUse	class attribute for LandUse	The land use classification of some area is represented with the associated LandUse class in 5087-2 whereas it is defined with a code assigned to the class attribute in CityGML.
-	function attribute for LandUse	The representation of the intended purpose of some area could also be captured by a particular subclass of LandUse in 5087, whereas in CityGML it is captured with a code assigned to the function attribute.
-	Use attribute for LandUse	5087-2 does not currently distinguish between usage and function, this could be something to incorporate in the future <> function attribute

B.2 INSPIRE

Scope

The INSPIRE directive is aimed at supporting the sharing of and access to spatial data throughout the EU, particularly those that can have an impact on the environment. INSPIRE aims to create an infrastructure to achieve this, part of which includes the specification of data models in UML. These specifications are defined according to 34 data themes, ranging from Addresses, to Geology, to Human Health and Safety. Data mappings between related terms are identified and summarized in Table 24~~Table 23~~.

Relevance

Out of all of the 34 data themes identified, the Cadastral Parcels, Land Cover, Land Use, and Buildings specifications are of particular relevance to the ontologies defined in 5087-2.

Data Mappings

The data models are defined in UML, however much of the intended semantics is captured in the accompanying description rather than the model itself. It is based on these descriptions that the mappings that follow are identified. A key distinction between INSPIRE and 5087-2 is that the classes identified in INSPIRE are all interpreted as spatial objects, whereas in 5087-2, the objects are defined as distinct classes that are related to spatial objects through a “hasLocation” property.

Table 24~~23~~: Relationship between related terms in 5087-2 and INSPIRE

5087-2 Term	Corresponding INSPIRE Term	Relationship
landuse: LandArea	CadastralParcel	The CadastralParcel term in INSPIRE is a specialization of LandArea as defined in 5087-2. INSPIRE describes the CadastralParcel as “…a single area of Earth surface (land and/or water), under homogeneous real property rights and unique ownership, real property rights and ownership being defined by national law.” In the context of 5087-2 a LandArea is a generic representation of an identified region

		of land (possibly but not necessarily with unique ownership). CadastralParcel has an associated cadastral reference and optional portrayal attributes whereas a LandArea does not. Both terms have some associated geometry and area.
landuse: LandArea	CadastralZoning	The CadastralZoning term in INSPIRE is a specialization of LandArea as defined in 5087-2. INSPIRE describes the CadastralZoning as “the intermediary areas (such as municipalities, sections, blocks, ...) used in order to divide national territory into cadastral parcels.” In the context of 5087-2 a LandArea is a generic representation of an identified region of land (possibly but not necessarily with unique ownership). CadastralZoning has an associated cadastral zoning reference and optional portrayal attributes whereas a LandArea does not. Both terms have some associated geometry and area.
-	AdministrativeUnit	Administrative units, authorities, and organizations are not clearly identified in 5087-2, but their addition should be considered.
landuse:LandArea, landuse:hasLandUse	LandCoverVector, LandCoverUnit	Both specifications adopt the same position for the representation of land use as that of INSPIRE: not to prescribe a particular land cover classification system but to provide a mechanism of capturing land cover data. 5087-2 associates land with various land use and classification systems with a single, hasLandUse property. Whereas this module of INSPIRE focuses solely on land cover, 5087-2 includes both types of classification systems in the Land Use Ontology. INSPIRE provides a generic structure (LandCoverNomenclature) for definining classification codes whereas 5087-2 does not. INSPIRE distinguishes between vector and raster land cover representations whereas 5087-2 does not. In 5087-2 land cover classes are associated with LandAreas in the same way as land use classes are assigned to areas. INSPIRE captures land over as observations and associates an observation date. It allows for the specification of a "covered percent" where in 5087-2 it is assumed that the land cover is defined to completely cover the identified area of land.
landuse:LandArea, landuse:hasLandUse	ExistingLandUseObject, ExistingLandUseSample, ExistingLandUseGrid,	5087-2 provides a generic approach to representing land use. Two land use classification systems are currently included, but it is intended to be easily extended with others. INSPIRE requires the use of the HILUCS (Hierarchical INSPIRE Land Use Classification System), but

	ZoningElement	also allows for the specification of other classification systems. INSPIRE clearly distinguishes between Land Use and Land Cover whereas 5087-2 does not, allowing for classification systems that combine the two notions. 5087-2 describes land use with the property hasLandUse that is associated to a LandArea. INSPIRE introduces different classes (geometries) to represent Existing Land Use, Sampled Land Use, Gridded Land Use, and Planned Land Use (zoning). Existing, Sampled, and Gridded Land Use are not identified explicitly in 508702, but can be captured with different spatial representations of the area being described. Unlike INSPIRE, the spatial plans from which some zoning originates are not captured in 5087-2 (but should be considered for future proposals).
building:Building	AbstractBuilding, BuildingAndBuildingUnitInfo,	In INSPIRE, AbstractBuilding (subclass of AbstractConstruction): allows for the representation of buildings and building parts. Like Building in 5087-2, it captures data on use, dwellings, etcetera, but unlike 5087-2 it represents only a count of these features, not actual relationships (e.g. between a person and a dwelling). INSPIRE includes classifications (code systems) for building nature (type of structure) and uses, whereas 5087-2 does not have such code systems. INSPIRE supports extended representations of an abstract building as 2D or 3D geometries. It also provides an extended representation (BuildingAndBuildingUnitInfo) that includes address representation and detailed physical characteristics such as materials and other building features (e.g. chimneys). The AbstractBuilding representation captures additional features, physical characteristics in particular, that are not addressed in 5087-2's definition of building. Construction information is not explicitly identified in 5087-2 but can be implicitly captured by changes in a Building's features.

B.3 ISO 19152

Scope

The Land Administration Domain Model (LADM) is a standard, global vocabulary for land administration. It includes a representation of parties (people and organizations), administrative units, and spatial units. Data mappings between related terms are identified and summarized in Table 25~~Table 24~~.

Relevance

A descriptive model with a goal of enabling the definition of a shared ontology for the domain. The LADM addresses concepts such as administrative areas, surveys, and policies that are pertinent to city activities such as land use modelling and simulation.

The parties involved in land administration are also relevant to the generic city concepts of persons and organizations. The administrative package of ISO 19152 can also serve to inform the representation of building occupancy and tenure.

Data Mappings

Table 25~~24~~: Relationship between related terms in 5087-2 and ISO 19152

5087-2 Term	Corresponding ISO 19152 Term	Relationship
iso:5087-1:Agent	LA_Party	TBD

B.4 ISO 19160-1

Scope

ISO 19160-1 defines a conceptual model for address information. It includes concepts needed to describe address information, along with concepts related to address metadata. The intent of the standard is to provide a common representation for address information that may be used to develop and enable translation between individual address specifications.

Relevance

ISO 19160-1 defines a conceptual model for addresses and is thus closely related to the ISO/IEC 5087-2 Contact Pattern, which represents contact information including addresses. ISO 19160-1 is formalized in UML and defines an address model at a higher, more abstract level of detail than that specified by the Contact Pattern. Some broader, city-related concepts may be identified using the ISO 19160-1 model with the use of predefined (or user-specified) codelists. In contrast, ISO/IEC 5087-2 includes patterns that define a city concepts, and these concepts may then be associated with an address, as defined by the Contact Pattern. Data mappings between related terms are identified and summarized in Table 23~~Table 22~~.

Data Mappings

Table 26~~22~~: Relationship between related terms in 5087-2 and ISO 19160-1

5087-2 Term	Corresponding ISO 19160-1 Term	Relationship
contact:Address	Address, AddressComponent	The contact:Address class in ISO/IEC 5087-2 defines an address with its properties, whereas the Address class in ISO 19160-1 is a higher-level construct. The properties defined for contact:Address would be represented with AddressComponent objects (below) related to a particular Address.

		The scope of the Address class is broader than contact:Address as it also aims to cover information such as an address' locale, parent and child addresses, and alias addresses. These attributes provide additional information about an address but are not necessary to define it for the purposes of contact.
Organization, Building, BuildingUnit, Person, Residence	AddressableObject, AddressComponent	ISO 19160-1 defines the generic concept of an AddressableObject. ISO/IEC 5087-2 identifies specific classes of AddressableObjects such as Organization and Building classes, with the contact:hasAddress property. Types of AddressableObjects may be specified with the use of a codelist in ISO 19160-1. In ISO 19160-1, an addressee may be defined as part of an Address with an AddressComponent. In ISO/IEC 5087-2, an addressee is not part of an address definition but may be identified through the definition of the contact:hasAddress property for a Person.
loc:Location	ReferenceObject	In ISO 19160-1, an AddressComponent, not the entire Address, may be defined as referring to a ReferenceObject. In ISO/IEC 5087-2, contact:Address may be defined as referencing a loc:Location with the loc:hasLocation property. In ISO 19160-1 the referenced object may be spatial or non-spatial (i.e., for addressees), whereas in ISO/IEC 5087-2, the referenced object may only be spatial.

B.5 ISO 19144-2

Scope

ISO 19144-2 is part of a standard series focused on the standardization of geographic classification systems. Specifically, ISO 19144-2 presents a metamodel for land cover classification systems defined in UML.

Relevance

ISO 19144-2 focuses on standardization of land cover classification systems and as such is conceptually related to the Land Use Pattern, as it may be extended to include land cover classification systems as well. Despite this, the focus of the metamodel presented in ISO 19144-2 is considerably different as it is intended to provide a structure to define the content of the classification systems such that they can be uniformly defined and compared with one another, whereas the focus on the Land User Pattern is on structuring the association of these classification systems to some Land Area. There are no direct mappings as the content of the model defined in ISO 19144-2 applies to the classification systems themselves, which are not included in the scope of this document.

Annex C
(informative)

Extending the Land Use Pattern with multiple classification systems

The Land Based Classification Standards (LBCS) Ontology[3] presented by [1] is reused for the representation of land use classifications. The LBCS recognizes different dimensions of Land Use: Activity, Function, Structure, Site, and Ownership Classifications. Each dimension is further defined by a taxonomy of specialized classifications. For each dimension, we introduce an equivalent class name for disambiguation, e.g. to distinguish between the Activity dimension of land use (we refer to this as ActivityClassification) and the foundational notion of an Activity.

Activity Classification: An Activity Classification identifies the activity use of some Land Parcel, examples include Residential Activities, Shopping Activities, and Industrial Activities.

Function Classification: A Function Classification identifies the economic function of some Land Parcel,

Structure Classification: A Structure Classification identifies the type of structure(s) on some Land Parcel.

Site Classification: A Site Classification identifies the state of the site development on some Land Parcel (e.g. is it developed or not?)

Ownership Classification: An Ownership Classification identifies any constraints on the use of the land and its ownership for some Land Parcel.

The LBCS Ontology can be imported in order to extend the Land Use Pattern as specified in the formalization in Table 27~~Table 25~~.

Table 27~~25~~: Extension of the Land Use Pattern with the LBCS.

Class	Property	Value Restriction
LBCSClassification		LandUseClassification
ActivityClassification	rdfs:subClassOf	LBCSClassification
	owl:equivalentClass	lbcs:Activity
FunctionClassification	rdfs:subClassOf	LBCSClassification
	owl:equivalentClass	lbcs:Function
StructureClassification	rdfs:subClassOf	LBCSClassification
	owl:equivalentClass	lbcs:Structure
SiteClassification	rdfs:subClassOf	LBCSClassification
	owl:equivalentClass	lbcs:Site
OwnershipClassification	rdfs:subClassOf	LBCSClassification

[3] Not available online

	owl:equivalentClass	lbcs:Ownership

CLUMPClassification: Canada Land Use Monitoring Program Classification is a type (subclass) of Land Use classification. CLUMP identifies 15 different types of land use, each with an associated code used in datasets. We have made the design decision that the code need not be unique to a particular land use classification, as a classification from one system can correspond to multiple classifications in CLUMP. CLUMP introduces the following land use classifications:

- B - Urban built-up area
- E - Mines, quarries, sand and gravel pits
    - Outdoor recreation
- H - Horticulture
- G - Orchards and vineyards
- A - Cropland
- P - Improved pasture and forage crops
- K - Unimproved pasture and range land
- T - Productive woodland
- U - Non-productive woodland
- M - Swamp, marsh or bog
- S - Unproductive land - sand
- L - Unproductive land - rock
    - Unmapped areas (technically not a CLUMP classification but it is used in land use data)
- Z - Water areas (technically not a CLUMP classification but it is used in land use data)

The Land Use Pattern can be extended to incorporate the CLUMP Classification system as specified in the formalization in Table 28~~Table 26~~.

Table 28~~26~~: Extension of the Land Use Pattern with the CLUMP Classification System.

Class	Property	Value Restriction
CLUMPClassification	rdfs:subClassOf	LandUseClassification
	equivalentTo	hasCLUMPCode min 1 xsd:string
UrbanBuiltUp	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "B"
MinesQuarriesSandGravelPits	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "E"
CLUMPCropland	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "A"
CLUMPWater	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "Z"
Horticulture	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "H"
ImprovedPasture	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "P"

NonProductiveWoodland	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "U"
OrchardsVineyards	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "G"
OutdoorRecreation	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "O"
ProductiveWoodland	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "T"
SwampMarshBog	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "M"
UnimprovedPasture	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "K"
Unmapped	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "8"
UnproductiveRock	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "L"
UnproductiveSand	rdfs:subClassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "S"

AAFCClassification: Agriculture and Agri-Foods Canada Classification is a type (subclass of) land use classification. The codes are based on the IPCC (International Panel on Climate Change) protocol. The code need not be unique to a particular land use classification, as a classification from one system can correspond to multiple classifications in AAFC. AAFC uses the following land use classifications:

- Unclassified
- Settlement
- Roads
- Water
- Forest
- Forest Wetland
- Trees
- Treed Wetland
- Cropland
- Grassland Managed
- Grassland Unmanaged
- Wetland
- Wetland Shrub
- Wetland Herb
- Other land
- TrafficZone: traffic zone is a kind of (subclass of) LandArea. It may be identified with a predefined set of identifiers, corresponding to its centroid node ID.

The Land Use Pattern can be extended to capture the AAFC Classification system as specified in the formalization in Table 29~~Table 27~~.

Table 29~~27~~: Extension of the Land Use Pattern with the AAFC Classification System.

Class	Property	Value Restriction
AAFCClassification	rdfs:subClassOf	LandUseClassification
	equivalentTo	hasAAFCCode min 1 xsd:string
Unclassified	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "11"
Settlement	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "21"
Roads	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "25"
Water	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "31"
Forest	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "41"
ForestWetland	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "42"
Trees	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "45"
TreedWetland	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "46"
AAFCCropland	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "51"
GrasslandManaged	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "61"
GrasslandUnmanaged	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "62"
Wetland	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "71"
WetlandShrub	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "73"
WetlandHerb	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "74"

OtherLand	rdfs:subClassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "91"

Annex D
(informative)

Location of Pattern Implementations

The patterns defined in this document are implemented as OWL files, available online at the following locations:

- Code Pattern: http://ontology.eil.utoronto.ca/5087/2/Code.owl
- Infrastructure Pattern: http://ontology.eil.utoronto.ca/5087/2/Infrastructure.owl
- Transportation Infrastructure Pattern: http://ontology.eil.utoronto.ca/5087/2/TransportationInfrastructure.owl
- Building Pattern: http://ontology.eil.utoronto.ca/5087/2/Building.owl
- Land Use Pattern: http://ontology.eil.utoronto.ca/5087/2/Landuse.owl
- Person Pattern: http://ontology.eil.utoronto.ca/5087/2/Person.owl
- City Resident Pattern: http://ontology.eil.utoronto.ca/5087/2/CityResident.owl
- Household Pattern: http://ontology.eil.utoronto.ca/5087/2/Household.owl
- Organization Pattern: http://ontology.eil.utoronto.ca/5087/2/Organization.owl
- City Pattern: http://ontology.eil.utoronto.ca/5087/2/City.owl
- City Service Pattern: http://ontology.eil.utoronto.ca/5087/2/CityService.owl
- Contract Pattern: http://ontology.eil.utoronto.ca/5087/2/Contract.owl
- Bylaw Pattern: http://ontology.eil.utoronto.ca/5087/2/Bylaw.owl
- Contact Pattern: http://ontology.eil.utoronto.ca/5087/2/Contact.owl
- Sensors Pattern: http://www.w3.org/ns/ssn/

Bibliography

[1] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition,* vol. 5, no. 2, pp. 199-220, 1993.

[2] M. Grüninger and M. S. Fox, "Methodology for the design and evaluation of ontologies," 1995.

[3] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific american,* vol. 284, no. 5, pp. 28-37, 2001.

[4] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, "OWL 2 web ontology language primer," *W3C recommendation,* vol. 27, no. 1, p. 123, 2009.

[5] W. W. W. Consortium, "RDF 1.1 Primer," 2014.

[6] *Semantic Sensor Network Ontology*, O. W3C, 2017. [Online]. Available: https://www.w3.org/TR/vocab-ssn/

[7] M. S. Fox, M. Barbuceanu, and M. Gruninger, "An organisation ontology for enterprise modelling: preliminary concepts for linking structure and behaviour," in *Proceedings 4th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'95)*, 1995: IEEE, pp. 71-81.

[8] M. Bennett, "The financial industry business ontology: Best practice for big data," *Journal of Banking Regulation,* vol. 14, no. 3-4, pp. 255-268, 2013.