OCS: A Foundational Ontology for Computational Sociology

Dr Edit Hlaszny
Dr Hlaszny Bioystems Engineering
Mail: edit@edithlaszny.eu
http://www.edithlaszny.eu/

Abstract

The history of computational sociology extends over the last 4.5 decades; its roots can perhaps be found in general systems theory and structural functionalism. Ontologies have been created in a wide range of subject areas and their number and application areas are dramatically growing. However, it can be considered quite well-founded to assume that no ontology has been created in the general sociological subject area so far. The *OCS* (Ontology for Computational Sociology) mentioned in the title makes a modest attempt at this, hoping that true experts in the subject area will find the topic itself (creating and further developing sociological ontologies) interesting. Therefore, let us quote modestly the esteemed Basel mathematician Johann Bernoulli, Opera Omnia, 67, Tom. I.: *"Problema novum ad cuius solutionem sociologi invitantur."*

Keywords

Computational sociology; ontology; Java-based application.

1. What's Actually in This Paper

1.1 For readers...

who'd rather cut to the chase than plough through the usual academic throat-clearing, here's what has actually been built:

- *The ontology itself:* 700+ classes covering general sociology, 254 object properties, 78 data properties, and 201 n-ary causal relations—all properly mapped under BFO 2020 top-level ontology classes.
- *A web-based browser* that lets you navigate the whole thing without getting lost in the conceptual weeds.
- *Development tools* for extending the system: Eclipse IDE integration, 60 Java classes, database management scripts, and a LaTeX-based documentation generator–because ontologies that can't evolve are rather pointless.

The paper walks through the theoretical foundations (skip if you're in a hurry), technical implementation (don't skip if you're planning to use this), and demonstrates why computational sociology might benefit from having its conceptual house in order.

Crack on—read what interests you, skip what doesn't :)

2. On computational Sociology

2.1 The Epistemological Foundations of Sociological Ontology

The fundamental methodological divide between social and natural sciences stems from their divergent subject matter and analytical challenges. Natural sciences examine phenomena governed by universal laws, enabling prediction and replication through controlled experimentation.

Social sciences confront human agency, cultural variation, and historical contingency, rendering absolute prediction impossible. Social phenomena emerge from complex interactions between individual choice and structural constraints, creating inherently interpretive challenges.

The observer-observed relationship further complicates social inquiry. Researchers cannot achieve complete detachment from their cultural context, whilst their subjects possess reflexive awareness that can alter behaviour under study.

Mathematical formalisation has historically correlated with scientific maturity across disciplines. Physics achieved predictive precision through mathematical modelling, whilst chemistry and biology developed rigorous quantitative frameworks as they matured.

However, this relationship requires nuanced evaluation. Mathematics provides analytical precision and enables hypothesis testing, yet its applicability varies across domains. Economics extensively employs mathematical methods whilst remaining contentious regarding predictive accuracy.

The presumption that mathematical sophistication equals scientific validity risks privileging quantification over explanatory depth. Complex social phenomena may resist meaningful reduction to mathematical representations without losing essential characteristics.

An elaborated sociological ontology would represent a significant advancement in computational sociology by providing systematic conceptual architecture for social phenomena. Traditional computational approaches often suffered from ad hoc categorisations and inconsistent terminology.

A rigorous ontological framework enables precise definition of social concepts, their relationships, and hierarchical organisation. This facilitates automated reasoning, knowledge integration, and comparative analysis across diverse sociological domains.

Ontological standardisation promises enhanced reproducibility in computational social research. Researchers can build upon shared conceptual foundations rather than constructing idiosyncratic frameworks for each investigation.

The *OCS* system demonstrates how formal ontological methods can capture sociological complexity whilst maintaining logical consistency. By grounding social concepts within established philosophical frameworks like BFO, it bridges humanistic insight with computational tractability.

Such developments suggest computational sociology's evolution from purely quantitative analysis towards sophisticated conceptual modelling. This represents methodological advancement rather than replacement of traditional sociological approaches.

The integration of ontological reasoning with empirical analysis may ultimately transcend the quantitative-qualitative divide by providing structured frameworks for both numerical data and interpretive understanding within unified analytical systems.

2.2 The Crucial Role of Ontologies in the Modern Era

In an age defined by the exponential growth of information, the discipline of ontology has transcended its philosophical origins to become a cornerstone of modern knowledge engineering. Ontologies, as formal specifications of a shared conceptualization, serve as foundational frameworks for structuring, organizing, and interpreting information in a machine-readable manner.

They provide a precise and unambiguous vocabulary of classes, properties, and relationships to model a domain of interest. This semantic rigor is essential or transforming unstructured data into meaningful, interconnected knowledge graphs. Beyond simple data categorisation, ontologies enable advanced forms of reasoning and inference, allowing computational systems to discover new relationships, validate logical consistency, and make informed decisions that would be impossible with traditional data models. The role of ontologies today is therefore not merely descriptive but is actively generative, creating the semantic infrastructure necessary for intelligent systems to operate effectively in an increasingly complex world.

3. The Semantic Web and Its Foundational Technologies

3.1 Technological overview

The vision of the Semantic Web, as an extension of the World Wide Web, is to make Internet data machine-readable and semantically

meaningful, facilitating seamless integration and automated reasoning. Its technical foundation is built upon a layered architecture of standards and languages designed to achieve this goal. At the core are resource description frameworks such as RDF (Resource Description Framework), which provides a simple, graph-based model for making statements about resources in the form of subject-predicate-object triples.

For the expression of more complex relationships and formal axioms, OWL (Web Ontology Language) serves as the primary language. OWL offers a rich set of constructors for defining classes, properties, and the intricate logical relationships between them, enabling sophisticated reasoning over the data.

The Web Ontology Language has three sublanguages (OWL Lite, OWL DL, and OWL Full) each offering different levels of expressiveness and corresponding reasoning capabilities. These ontologies are often encoded in standardised formats such as OWL/XML, RDF/XML, or JSON-LD, ensuring interoperability across different tools and platforms.

The ecosystem of semantic technologies includes reasoners (such as FaCT++ and HermiT) that perform logical inference and consistency checks, query languages like SPARQL that enable complex graph queries, and various API libraries and software frameworks that facilitate the development and manipulation of semantic data. Collectively, these technologies provide a robust and powerful toolkit for building and leveraging semantic representations of knowledge.

3.2 The Ontology for Computational Sociology in Practice

The *OCS* represents a formal and systematic conceptualization of the domain of sociology. It provides a foundational vocabulary of classes and properties for modeling the full spectrum of social phenomena, from micro-level interactions and individual dispositions to macro-level social structures, institutions, and global processes. By rigorously defining these concepts and their relationships, the *OCS* serves as a powerful instrument for:

Research:
It enables researchers to formalise hypotheses and theories in a machine-readable format, facilitating automated reasoning and the discovery of non-obvious connections between disparate social concepts. For instance, a researcher could use the ontology to query for all bfo:processes that occur in a bfo:realizable entity (like Law), or to analyse how different forms of Social_Control (realizable entity) are linked to various types of Deviance (process). This level of formalisation supports quantitative, qualitative, and mixed-methods research by providing a common semantic ground.

Education:
As a pedagogical tool, the *OCS* can be used to teach students the foundational concepts and theoretical frameworks of sociology in a structured and interconnected way. It visually represents the relationships between different schools of thought, key concepts, and their hierarchical organisation, providing a clear and comprehensive map of the discipline.

Interoperability:
The alignment of *OCS* with a robust upper ontology like BFO 2020 ensures that its concepts can be semantically integrated and reasoned over with other domain ontologies in fields such as public health, economics, political science, and environmental science. This allows for a trans-disciplinary understanding of complex societal issues, where sociological insights can be linked with data and knowledge from other fields to create a more holistic and powerful knowledge base for research, policy-making, and social analysis.

4. Basic considerations of *OCS* object properties

4.1 Object properties and their inverses

The existence of an object property does not logically entail the existence of its inverse - this would require an additional axiom or inference rule. However, the situation with reasoners is more subtle than it might initially appear.

The Reasoner Perspective
Some reasoners (particularly those implementing complete tableaux algorithms for description logics like ALCIQ or SHOIN) can indeed work with implicit inverse relationships without requiring explicit inverse property declarations. They achieve this through:

Query rewriting: When encountering a pattern that would benefit from an inverse property, they can reformulate queries to use the original property in reverse

Internal inverse handling: They maintain internal representations that treat $\mathtt{P}(\mathtt{x},\mathtt{y})$ and $\mathtt{P}^{-1}(\mathtt{y},\mathtt{x})$ as equivalent without requiring explicit declaration of $\mathtt{P}^{-1}$. However, this capability varies significantly across reasoners and reasoning tasks.

4.2 The Case for Explicit Inverse Properties

Despite some reasoners' capabilities, there are compelling reasons to define inverse properties explicitly:

Semantic Clarity:
Inverse properties often represent genuinely distinct conceptual relationships. In a sociology ontology, "hasChild" and "hasParent" aren't merely logical inverses - they capture different social and conceptual perspectives on kinship relations.

Reasoner Agnosticism:
Not all reasoners handle implicit inverses equally well, particularly when dealing with complex property chains, transitivity, or functional properties.

Query Expressiveness:
Explicit inverses enable more natural and efficient SPARQL queries and API interactions.

Ontological Completeness:
If a relationship is conceptually bidirectional and both directions are meaningful in your domain, explicit representation better captures the ontological structure.

A Pragmatic Approach:
Many people believe that some kind of pragmatic approach is worth taking: Rather than defining inverses for "every possible case," they use this heuristic. *Define explicit inverses when:* The inverse represents a conceptually distinct relationship (hasChild/hasParent), when the inverse is frequently queried, or when it participates in different axioms or property characteristics. *Rely on implicit inverses when:* The inverse is purely a logical convenience with no distinct conceptual content.

4.3 Computational overhead

For *OCS*, relationships like "belongsToOrganisation / hasMember" or "influencedBy / influences" likely warrant explicit inverse properties due to their conceptual significance in sociological analysis. The computational overhead of additional properties is generally negligible compared to the benefits in semantic expressiveness and reasoning reliability.

Nevertheless, the *OCS* includes inverse properties for each object property, ensuring that future ontologists developing the system have these inverses readily available and can choose whether to utilise them or not. This hierarchy provides a comprehensive foundation for sociological object properties. Each object property includes both the forward and inverse relationships, along with concise annotations explaining their sociological significance.

4.4 Object properties follow sociological dimensions

- *Communication:* the mechanism through which social reality is constructed and maintained
- *Cultural relations:* shared meaning systems that bind communities
- *Demographic properties:* essential for population and life course classes

- *Deviance and control properties:* crucial for extensive deviance and crime classes
- *Economic relations:* material basis of many social relationships and inequalities
- *Environmental properties:* important for your spatial and urbanization classes
- *Health/medical properties:* important for medical sociology concepts
- *Membership and belonging:* essential for group dynamics and identity formation
- *Organisational properties:* important for many institutional and organisational classes
- *Political properties:* necessary for governance and power-related classes
- *Power and influence:* fundamental to understanding social stratification and control
- *Research methodology properties:* essential for connecting research-related classes (Census, Interview, Observation, etc.)
- *Social control:* mechanisms for maintaining order and transmitting culture
- *Social relationships:* the building blocks of social networks and community
- *Spatial/temporal relations:* contextual factors shaping social interaction
- *Stratification properties:* vital for social class and inequality concepts
- *Technology/media properties:* increasingly crucial for contemporary sociology

These entities (the *OCS* contains a total of 254 object properties) work well with the class hierarchy (674 classes) of the ontology, particularly with entities like Social_Organisations, Social_Groups, Social_Processes, and the various institutional categories.

4.5 Object Property Characteristics

The ontology browser displays all the characteristic bits of the object properties in a table. Characteristics can be

Functional

A functional object property is a relationship that, for a given subject, can have only one object. In other words, if an entity $\mathcal{A}$ is related to an entity $\mathcal{B}$ via a functional property $\mathcal{P}$, then $\mathcal{A}$ cannot also be related to any other entity $\mathcal{C}$ via that same property $\mathcal{P}$ (unless $\mathcal{C}$ is the same as $\mathcal{B}$). This enforces a one-to-one or many-to-one relationship from subject to object.

Inverse Functional

An inverse functional object property is a relationship where the inverse of the property is functional. This means that if an entity $\mathcal{A}$ is related to an entity $\mathcal{B}$ via a property $\mathcal{P}$, then $\mathcal{B}$ cannot also be the object of that same property $\mathcal{P}$ for any other entity $\mathcal{C}$ (unless $\mathcal{C}$ is the same as $\mathcal{A}$. This enforces a one-to-one or one-to-many relationship from subject to object.

Transitive

A transitive object property is a relationship where if an entity $\mathcal{A}$ is related to $\mathcal{B}$ via the property $\mathcal{P}$, and $\mathcal{B}$ is related to $\mathcal{C}$ via $\mathcal{P}$, then the reasoner can automatically infer that $\mathcal{A}$ is also related to $\mathcal{C}$ via $\mathcal{P}$.

Symmetric

A symmetric object property is a relationship where if an entity $\mathcal{A}$ is related to $\mathcal{B}$ via the property $\mathcal{P}$, then the reasoner automatically infers that $\mathcal{B}$ is also related to $\mathcal{A}$ via $\mathcal{P}$.

Asymmetric

An asymmetric object property is a relationship where if an entity $\mathcal{A}$ is related to $\mathcal{B}$ via the property $\mathcal{P}$, it is automatically inferred that $\mathcal{B}$ cannot be related to $\mathcal{A}$ via $\mathcal{P}$. This is a stronger condition than simply being irreflexive, as it states that the inverse relationship is impossible.

Reflexive

A reflexive object property is a relationship where every individual is related to itself via that property. This is a less common characteristic in sociological modeling.

Irreflexive

no individual can be related to itself via that property. This is a very common characteristic for many relationships in sociology.

This interpretation of object properties is also intended to help shed more light on the ontology browser data.

↑ friendOf
Annotation: (Voluntary, reciprocal relationship characterized by mutual affection, trust, and companionship.)
Annotation source: (editor)
Annotation type: skos:definition
Language: en
Super object property: relatedTo
Inverse object property: hasFriendship
Sub-object property: –
Referred relation(s): –

Object property characteristics:	Functional	Inverse Functional	Symmetric	Asymmetric	Transitive	Reflective	Irreflective
	–	–	✔	–	–	–	–

Figure 1: Object property characteristics showing in the browser.

5. The N-ary Relation Framework

5.1 A Formal Architecture for Sociological Causation

The representation of complex sociological causation within ontological structures presents fundamental challenges to standard binary relationship models. The framework implemented addresses this through a rigorous reification pattern that preserves both the collective nature of causal mechanisms and the multiplicity of their consequences whilst maintaining computational tractability.

Sociological phenomena characteristically exhibit causal relationships wherein multiple factors operate conjointly to produce various effects. A binary predicate structure, limited to expressing relations between precisely two entities, proves inadequate for capturing such complexity. The n-ary relation framework transcends this limitation through systematic reification—the explicit representation of relationships themselves as first-class ontological entities.

5.2 Architectural Components

The framework comprises five interrelated structural elements:

1. Reified Causal Events: Each n-ary relationship instantiates a unique individual of the class Collective_Causal_Event. This reified entity serves as the ontological anchor for the entire causal structure, providing a singular referent for what is conceptually a complex, multi-participant relationship.
2. Contributory Linkages: Domain elements—those entities functioning as causal antecedents—connect to the reified event through the object property contributesToCausalEvent. This establishes their role as conjoint causal factors whilst maintaining their individual identities within the larger causal mechanism.
3. Consequential Linkages: Range elements—the effects or outcomes—connect from the reified event via the object property causalEventProduces. This directional relationship preserves the causal asymmetry inherent in sociological explanation whilst accommodating multiple simultaneous consequences.
4. Cartesian Product Assertions: The framework generates exhaustive binary assertions between each domain element and each range element using the primary relational predicate. This provides direct access paths for reasoning engines whilst preserving semantic transparency for domain experts unfamiliar with reification patterns.
5. Semantic Annotations: Each reified event individual carries structured annotations documenting the theoretical basis, empirical support, and sociological significance of the causal relationship, transforming the formal structure into interpretable scholarly knowledge.

5.3 Epistemological Advantages

This architecture offers several methodological benefits for computational sociology:

1. The reification pattern enables attachment of meta-level properties—certainty measures, evidential support, temporal scope—to relationships themselves rather than merely to participating entities. This captures the epistemological status of causal claims with appropriate granularity.
2. The dual representation—both reified events and direct Cartesian assertions—accommodates varying degrees of ontological sophistication amongst users. Domain experts may query direct relationships intuitively whilst formal reasoners exploit the richer reified structure.
3. The framework maintains extensibility: additional causal factors or consequences may be incorporated without restructuring existing relationships, supporting the iterative refinement characteristic of sociological theory development.

5.4 Formal Semantics

Let $D = \{d_1, d_2, \ldots, d_n\}$ represent domain entities and $R = \{d_1, d_2, \ldots, d_m\}$ represent domain entities represent range entities in a causal relationship mediated by predicate P. The framework generates:

1. A unique reified individual (Figure 2):
   $e \in Collective\_Causal\_Event$
2. Contributory assertions: (Figure 3):
   $contributesToCausalEvent(d_i, e)$ for all $d_i \in D$
3. Consequential assertions: (Figure 4):
   $causalEventProduces(e, r_j)$ for all $r_j \in R$
4. Direct assertions: (Figure 5):
   $P(d_i, r_j)$ for all $d_i \in D$ and $r_j \in R$

This generates n contributory linkages, m consequential linkages, and $n \times m$ direct assertions, providing multiple inference paths whilst maintaining logical coherence.

5.5 An OWL/XML encoded example

```
<!--
    n-ary causal relation:
        Youth Mobilization and Social Change

    {Youth_Culture, Social_Movement} → mobilizes →
    {Social_Change, Culture}
  -->

<!--  freshing reified individual   -->
<Declaration>
    <NamedIndividual IRI="#causal_Event_ID000201"/>
</Declaration>
<ClassAssertion>
    <Class IRI="#Collective_Causal_Event"/>
    <NamedIndividual IRI="#causal_Event_ID000201"/>
</ClassAssertion>

<AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="skos:definition"/>
     <IRI>#causal_Event_ID000201</IRI>
    <Literal>The causal_Event_ID000201 named individual is
             associated to the relation: 'Youth Mobilization
             and Social Change'. The individual represents
             a reified causal event capturing the sociological
             defined by the name of the relation and the
             symbolic description. The causal_Event_ID000201
             is named individual of the Collective_Causal_Event
             class, with identifiers reflecting their systematic
             role in modeling complex n-ary sociological
             relationships.
    </Literal>
</AnnotationAssertion>
```

Figure 2: n-ary causal relation: "Youth Mobilization and Social Change". Generating a unique reified individual for each n-ary causal relation. Each individual is annotated.

```
<!--  connecting causes to event  -->
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#contributesToCausalEvent"/>
    <NamedIndividual IRI="#youth_Culture_ID000002"/>
    <NamedIndividual IRI="#causal_Event_ID000201"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#contributesToCausalEvent"/>
    <NamedIndividual IRI="#social_Movement_ID000014"/>
    <NamedIndividual IRI="#causal_Event_ID000201"/>
</ObjectPropertyAssertion>
```

Figure 3: Connecting causes to event.

```
<!--  connecting event to effects  -->
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#causalEventProduces"/>
    <NamedIndividual IRI="#causal_Event_ID000201"/>
    <NamedIndividual IRI="#social_Change_ID000016"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#causalEventProduces"/>
    <NamedIndividual IRI="#causal_Event_ID000201"/>
    <NamedIndividual IRI="#culture_ID000009"/>
</ObjectPropertyAssertion>
```

Figure 4: Connecting event to effects.

```
<!--  direct Cartesian product assertions  -->
<ObjectPropertyAssertion>
    <ObjectProperty  IRI="#mobilizes"/>
    <NamedIndividual IRI="#youth_Culture_ID000002"/>
    <NamedIndividual IRI="#social_Change_ID000016"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty  IRI="#mobilizes"/>
    <NamedIndividual IRI="#youth_Culture_ID000002"/>
    <NamedIndividual IRI="#culture_ID000009"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty  IRI="#mobilizes"/>
    <NamedIndividual IRI="#social_Movement_ID000014"/>
    <NamedIndividual IRI="#social_Change_ID000016"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty  IRI="#mobilizes"/>
    <NamedIndividual IRI="#social_Movement_ID000014"/>
    <NamedIndividual IRI="#culture_ID000009"/>
</ObjectPropertyAssertion>
```

Figure 5: Direct Cartesian product assertions.

5.6 Implementation Significance

The successful deployment of this framework across 201 causal relationships within the Subject Ontology demonstrates its practical viability. The architecture proves computationally tractable—reasoning completes within acceptable timeframes—whilst providing the semantic richness required for sophisticated sociological analysis.

This reification pattern represents a principled solution to the representation of complex causation in formal ontologies, balancing theoretical rigour with practical utility. It provides computational sociology with infrastructure capable of expressing the nuanced, multi-factorial causal relationships that characterise social phenomena, thereby advancing the discipline's capacity for formal knowledge representation and automated reasoning.

After all, this constitutes the crux of ontological work where genuine domain expertise proves essential. The technical framework provides the infrastructure, but its true value emerges only when populated with relationships grounded in rigorous sociological theory and empirical research.

Whilst the author has endeavoured to implement a sound computational architecture, the substantive sociological content—the selection of causal factors, the theoretical justification of relationships, the nuanced understanding of social mechanisms—requires depth of disciplinary knowledge that can only come from established scholars in the field.

It is here, in the realm of sociological interpretation rather than technical implementation, that the author's limitations become most apparent, and where collaboration with domain experts would prove invaluable.

6. Technical Details

6.1 Ontology Development Structure

The development pipeline architecture instantiates a two-stage workflow separating domain conceptualisation from ontological formalisation. The interface is made up of the central data repository, which is a relational database management system (RDBMS).

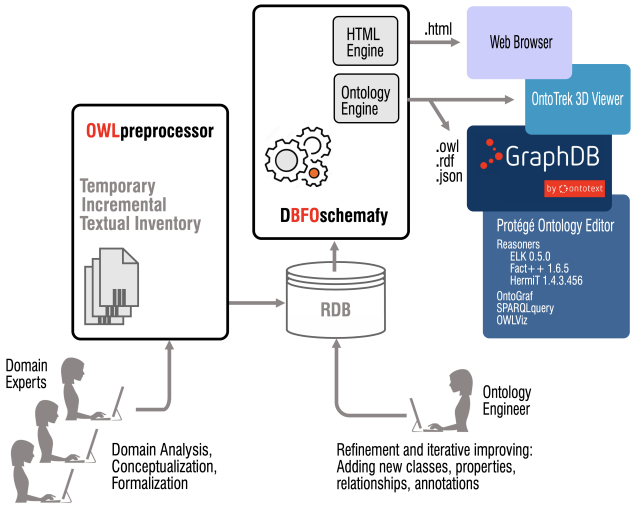

Figure 6: Development pipeline structure.

Domain experts interact with an OWL preprocessor that maintains a temporary incremental textual inventory—essentially structured text documents specifying classes, properties, relationships, and annotations in a simplified syntax accessible to non-technical sociologists. This preprocessor validates input consistency and translates domain specifications into database-compatible formats.

The central component, DBFOschemafy, functions as the ontology generation engine. It queries the relational database to retrieve validated domain specifications, then orchestrates two parallel output streams: an HTML engine produces human-readable documentation with collapsible hierarchical trees and cross-referenced relationship networks, whilst the ontology engine generates standards-compliant OWL/XML suitable for consumption by reasoning tools.

The system leverages the Basic Formal Ontology (BFO 2020) as upper-level scaffolding, ensuring ontological consistency with established scientific ontologies. BFO is an ISO standard (ISO/IEC 21838-2). As such, BFO serves as a baseline for information sharing practices, and enables coherent interoperability of heterogeneous data.

The right-hand ecosystem represents standard ontology engineering tools: Protégé provides visual editing and reasoning capabilities (ELK, FaCT++, HermiT), OntoTrek enables 3D visualisation of complex relationship networks, and GraphDB offers SPARQL querying with inference support. Critically, the architecture supports bidirectional workflow—ontology engineers can refine the generated OWL directly in Protégé as well, with modifications potentially feeding back into the database for subsequent iterations.

6.2 Ontology Metrics

The detailed *OCS* ontology metrics can be seen on the next figure, as the Protégé it presents.

Ontology metrics:

Metrics

Axiom	8,564
Logical axiom count	4,067
Declaration axioms count	2,073
Class count	701
Object property count	391
Data property count	79
Individual count	938
Annotation Property count	54

Class axioms

SubClassOf	739
EquivalentClasses	0
DisjointClasses	129
GCI count	0
Hidden GCI Count	0

Object property axioms

SubObjectPropertyOf	288
EquivalentObjectProperties	0
InverseObjectProperties	147
DisjointObjectProperties	0
FunctionalObjectProperty	10
InverseFunctionalObjectProperty	3
TransitiveObjectProperty	23
SymmetricObjectProperty	29
AsymmetricObjectProperty	10
ReflexiveObjectProperty	12
IrrefexiveObjectProperty	6
ObjectPropertyDomain	64
ObjectPropertyRange	64
SubPropertyChainOf	0

Data property axioms

SubDataPropertyOf	78
EquivalentDataProperties	0
DisjointDataProperties	0
FunctionalDataProperty	0
DataPropertyDomain	0
DataPropertyRange	78

Individual axioms

ClassAssertion	938
ObjectPropertyAssertion	1,443
DataPropertyAssertion	6
NegativeObjectPropertyAssertion	0
NegativeDataPropertyAssertion	0
SameIndividual	0
DifferentIndividuals	0

Annotation axioms

AnnotationAssertion	2,419
AnnotationPropertyDomain	0
AnnotationPropertyRangeOf	0

Figure 7: *CSO* — ontology metrics.

Ontology metrics are quantitative measures used to assess and evaluate an ontology's quality, complexity, and characteristics, enabling users to track its evolution and understand its suitability for a particular application. These metrics provide objective data by counting elements like classes, axioms (logical statements), and properties.

6.3 Using Basic Formal Ontology (BFO-2020)

The Basic Formal Ontology (BFO) serves as an upper-level, domain-neutral ontology designed to facilitate the development of specialised domain ontologies at lower levels. BFO enables formal logical reasoning and incorporates a collection of established formal theories—including mereotopology and qualitative spatial reasoning, with potential extensions to numerical theory—thereby eliminating the need to reconstruct these foundations for each new domain.

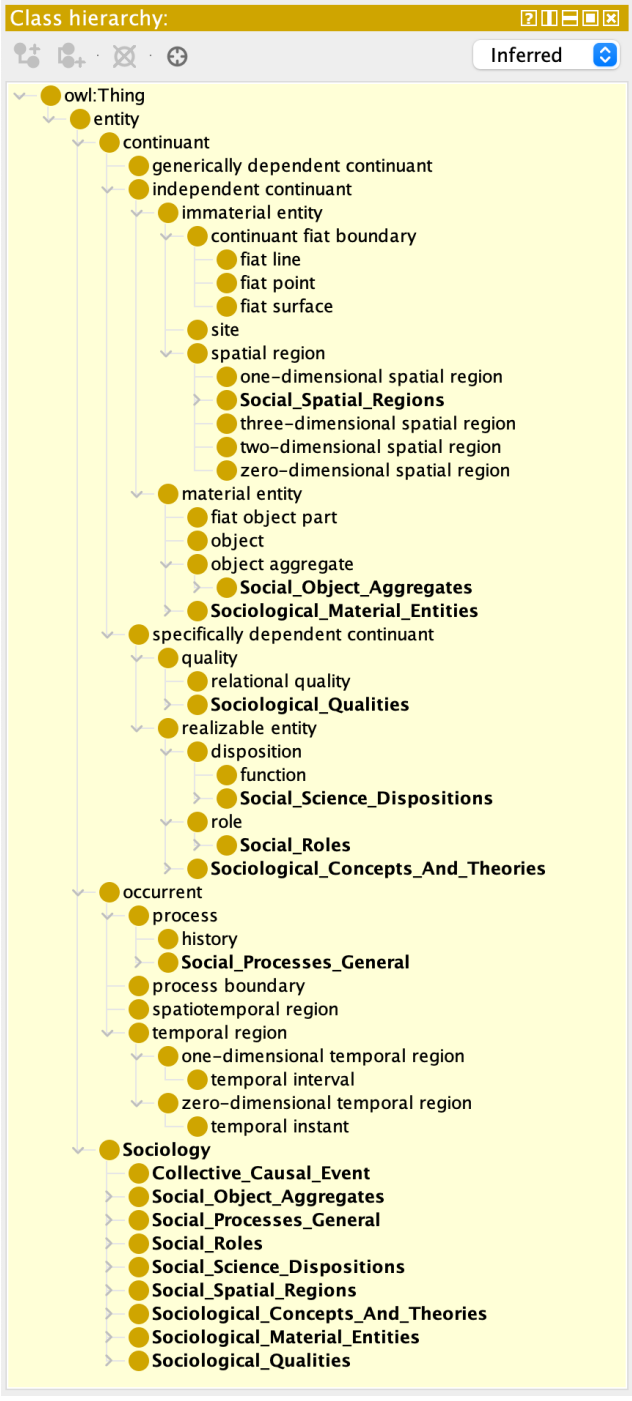

Figure 8: BFO 2020 Structure showing in Protégé.

Over 450 ontology initiatives have adopted BFO as their foundational framework, predominantly within biomedical ontology, security and defence intelligence ontology, and industrial ontology domains. The Ontology for Biomedical Investigations (OBI) provides a notable illustration of BFO's practical applications.

As shown in Figure 6, DBFOschemafy also generates output in HTML format, enabling BFO classes to be linked directly from the browser. These links direct users to the corresponding entries within the Ontology Lookup Service of the European Molecular Biology Laboratory's European Bioinformatics Institute.

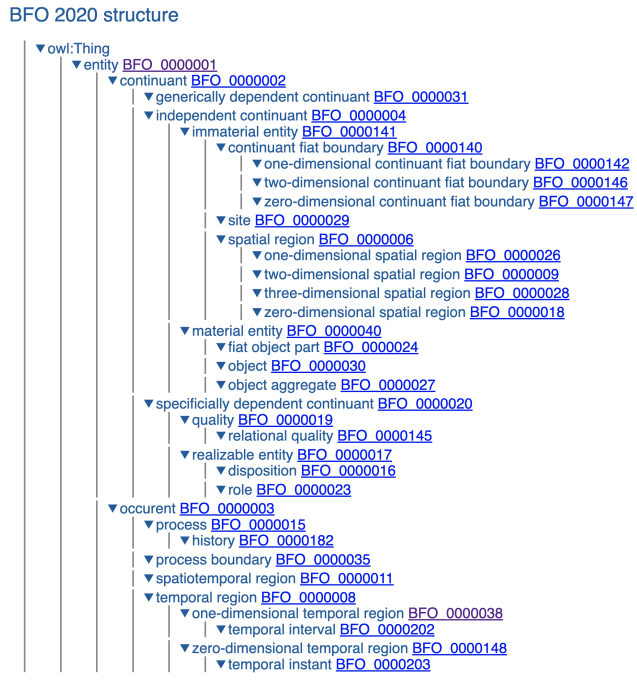

Figure 9: BFO 2020 Structure shown in browser.

BFO is used in various ways: some practitioners completely restructure lower-level domain ontologies by directly subclassing their classes under BFO classes, whilst others employ BFO class names within domain class annotations. The *OCS* ontology adopts a different approach, which the author believes aligns more closely with Barry Smith's original vision.

6.4 The Ontology Browser

OCS is sparsely annotated, containing only the most necessary data (title, author, timestamp, ...).

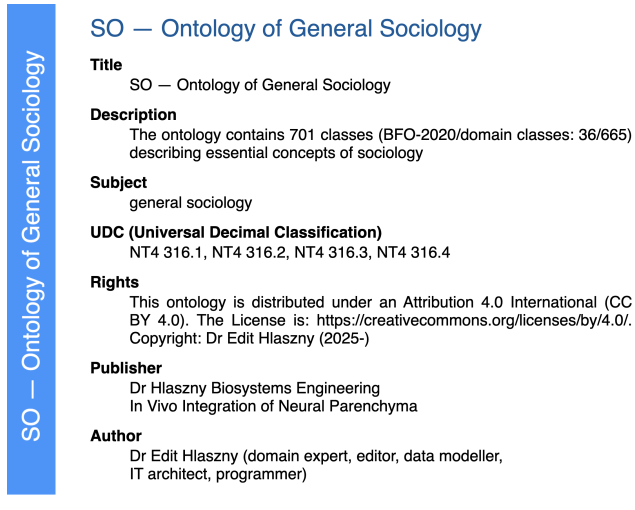

Figure 10: *OCS* introductory annotation.

The following two figures show an example of an OWL class, and one of the 201 n-ary causal relationships.

↑ Colonialism
Annotation: Colonialism is the maintenance of political, social, economic, and cultural dominance over a people by a foreign power for an extended period of time. This often involves the establishment of settlements and the exploitation of resources in the colonized territory for the benefit of the colonizing power. Colonialism has had a profound and lasting impact on the political, economic, and social structures of many regions around the world, leading to significant cultural changes and often resulting in inequalities and conflicts that persist even after the end of colonial rule. The motivations for colonialism have varied throughout history, including economic gain, political power, strategic advantage, and the spread of religious or cultural beliefs. The study of colonialism examines its historical roots, its various forms and impacts, and its enduring legacies in the contemporary world.
Annotation source: (editor)
Annotation type: skos:definition
Language: en
Subclass(es): —
Superclass(es):
Sociological_Social_Control_And_Collective_Action_Frameworks
BFO 2020 Superclass: BFO_0000017
Disjoint class(es): Informal_Social_Control, Institutional_Discrimination, New_Social_Movements

Figure 11: OWL class in the *OCS* browser.

↑ Bureaucratic Rationalization

Annotation: This relation captures Weber's concept of increasing emphasis on efficiency, calculability, and formal rules in modern organizations and societies, leading to systematic administrative control.
Parameters: {Bureaucratization, Rationalization → bureaucratizes → Bureaucracy, Formal_Organization}
Annotation source: (editor)
Language: en
Predicate: bureaucratizes
Relation Id: 11
Causal even individual: collective_Causal_Event_ID000011
S u b j e c t (s)
Class name: Bureaucratization
Data property name: —
Data property value: —
Data property type: —
Participant Id: 37
Relation Id: 11
Class name: Rationalization
Data property name: —
Data property value: —
Data property type: —
Participant Id: 38
Relation Id: 11
O b j e c t (s)
Class name: Bureaucracy
Data property name: —
Data property value: —
Data property type: —
Participant Id: 39
Relation Id: 11
Class name: Formal_Organization
Data property name: —
Data property value: —
Data property type: —
Participant Id: 40
Relation Id: 11

Figure 12: N-ary causal relation example.

Clicking on the arrow in front of the entity names takes the user to the beginning of the browser. The Figure 13 shows an example of an object property (254 in total).

↑ bureaucratizes
Annotation: (Implementation of formal rules, hierarchies, and procedures for organizational efficiency and control.)
Annotation source: (editor)
Annotation type: skos:definition
Language: en
Super object property: institutionalizes
Inverse object property: bureaucratizedBy
Sub-object property: —
Referred relation(s):
Relation name: **Bureaucratic Rationalization**
{Bureaucratization, Rationalization} → bureaucratizes → {Bureaucracy, Formal_Organization}
Relation name: **Bureaucratization and Social Rationalization**
{Bureaucracy, Rationalization} → bureaucratizes → {Alienation, Formal_Organization}
Relation name: **Politics Influence Organizations**
{Political_Systems_And_Governance_Forms} → bureaucratizes → {Formal_Organization}

Figure 13: Object property example.

The data properties (78 in total) have a layout similar to the object properties.

↑ *_causal_Certainty*
Annotation: This property quantifies the degree of confidence in a causal relationship's validity. It addresses the epistemological challenge that sociological causation often involves probabilistic rather than deterministic relationships. The property allows researchers to express varying levels of certainty about causal claims, acknowledging that some relationships are well-established through extensive empirical evidence while others remain tentative or contested within the discipline.
Annotation source: (editor)
Data property type: xsd:decimal
Language: en
Super data property: owl:topDataProperty
Sub-data property: —

Figure 14: Data property example.

The *OCS* ontology contains 654 classes, in addition to which the BFO has 36 classes. The resulting 701 classes are shown in Figure 7 (ontology metrics).

7. Software Background

7.1 Java on Eclipse IDE

The *OCS* development environment is simple, fairly homogeneous, reliable and explicitly hardware-independent: Java classes, bash shell scripts and MySQL DBMS are running under Eclipse.

The *OCS* system consists of six Eclipse projects, of which four are Java projects, one is Bash shell-based and one is capable of performing LaTeX typesetting.

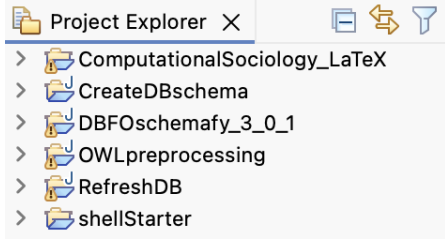

Figure 15: Eclipse projects.

The *OCS* distribution package contains all source files, configuration text files, and shell scripts: thus, users of the system have the ability to make all possible changes, also *OCS* is not merely a theoretical concept, a realised ontology, but a fully functional Java application. To facilitate further research and development, all components of the system are made available to the research community, including the Java source code, control files, and database schema and content.

The system itself can of course be used without the Eclipse IDE, it only requires a text-based dumb terminal. The menu system that can be activated on the terminal is quite easy to use, and ontological enhancements of the *OCS* system, such as adding new n-ary causal relationships, classes, object- or data properties, are fully feasible.

The menu system, accessible via a terminal, is available in two versions: so even those who prefer less IT slang can easily navigate the menu items, which are not too complicated anyway.

```
~/Desktop/ECLIPSEws2/shellStarter % ./SOworkflowManager.sh academic

     Ontological Data Processing Workflow Manager

  1  -   Initialize repository structure
  2  -   Launch semantic preprocessing module
  3  -   Integrate processed semantic data into repository
  4  -   Execute schema transformation and output generation
  5  -   Provide access to human-readable ontology interface
  6  -   Provide access to formal ontology representation
  7  -   EXIT SYSTEM
```

Figure 16: Workflow manager for ontologists.

```
~/Desktop/ECLIPSEws2/shellStarter % ./SOworkflowManager.sh technical

     Controlling the Computational Sociology Development Pipeline

  1  -   Initialize database schema
  2  -   Execute OWL preprocessing engine
  3  -   Synchronize database with preprocessed OWL data
  4  -   Generate OWL and HTML artifacts via DBFOschemafy
  5  -   Display ontology browser interface URL (HTML)
  6  -   Display ontology endpoint URL (OWL/XML format)
  7  -   EXIT MENU
```

Figure 17: Workflow manager for IT professionals.

7.2 Software Components

The software development environment consists of the following components:

- *Operating System:* macOS Tahoe (Version 26.0.1) running on Mac Studio (M3 ULTRA, 96GB, 8TB)
- *RDB:* MySQL Server (Version 8.0.30), MySQL Workbench (Version 8.0.31 build 2235049 CE (64 bits) Community)
- *Java Runtime:* Java™ SE Runtime Environment (build 13.0.2+8)
- *Java Virtual Machine:* Java HotSpot™ 64-Bit Server VM (build 13.0.2+8, mixed mode, sharing)
- *Integrated Development Environment:* Eclipse IDE for Enterprise Java and Web Developers (includes Incubating components) Version: 2025-03 (4.35.0) Build id: 20250306-0812 + TeXlipse 2.03 plugin
- *Protégé:* Open-source ontology editor and framework for building intelligent systems, Version 5.6.3

8. References

[1] Allemang, D., & Hendler, J.A. (2012). *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann.

[2] Arp, R., Smith, B., & Spear, A.D. (2015). *Building Ontologies with Basic Formal Ontology*. MIT Press.

[3] Baader, F., Horrocks, I., Lutz, C., & Sattler, U. (2017). *An Introduction to Description Logic*. Cambridge University Press.

[4] Boudon, R., & Bourricaud, F. (1989). *A Critical Dictionary of Sociology*. Routledge.

[5] Calhoun, C., Gerteis, J., Moody, J., Pfaff, S., & Virk, I. (2022). *Contemporary Sociological Theory* (4th ed.). Wiley-Blackwell.

[6] Giddens, A. (2009). *Sociology* (6th ed.). Polity Press, Cambridge UK.

[7] Horridge, M. (2011). *A Practical Guide to Building OWL Ontologies Using Protégé 4 and CO-ODE Tools*. The University of Manchester.

[8] Mueller, G.H. (1989). *Sociology and Ontology: The Analytical Foundations of Sociological Theory*. University Press of America.

[9] Turner, B.S. (Ed.). (2006). *The Cambridge Dictionary of Sociology*. Cambridge University Press.

[10] Turner, J.H. (2013). *Theoretical Sociology: 1830 to the Present*. SAGE Publications.

[11] W3C. (n.d.). *OWL – Semantic Web Standards*. Retrieved from `https://www.w3.org/OWL/`

Online Content

Any source and control data, extended data, pictures, supplementary information, and the LaTeX-source of this paper are available electronically.

Base URL:	https://www.hlaszny.com/OCS/
Present paper:	`paper/OCS.pdf`
LaTeXsource:	`paper/OCS.tex`
Eclipse Wokspaces:	`EclipseOCSwsWS/`
RDB content:	`EclipseOCSws/OWLpreprocessing/SQL/*.sql`
Java sources:	`EclipseOCSws/*/*/*/*.java`
HTML browser:	`data/OCS.html`
Ontology:	`data/OCS.*`

Technical Support and Collaboration

Readers are cordially invited to contact the author concerning any matters relating to the *OCS* system, including installation procedures, system enhancement, or the resolution of technical difficulties. The author is pleased to provide guidance and welcomes collaborative discussion regarding the system's ongoing development.

Correspondence may be directed to: edithlaszny@gmail.com