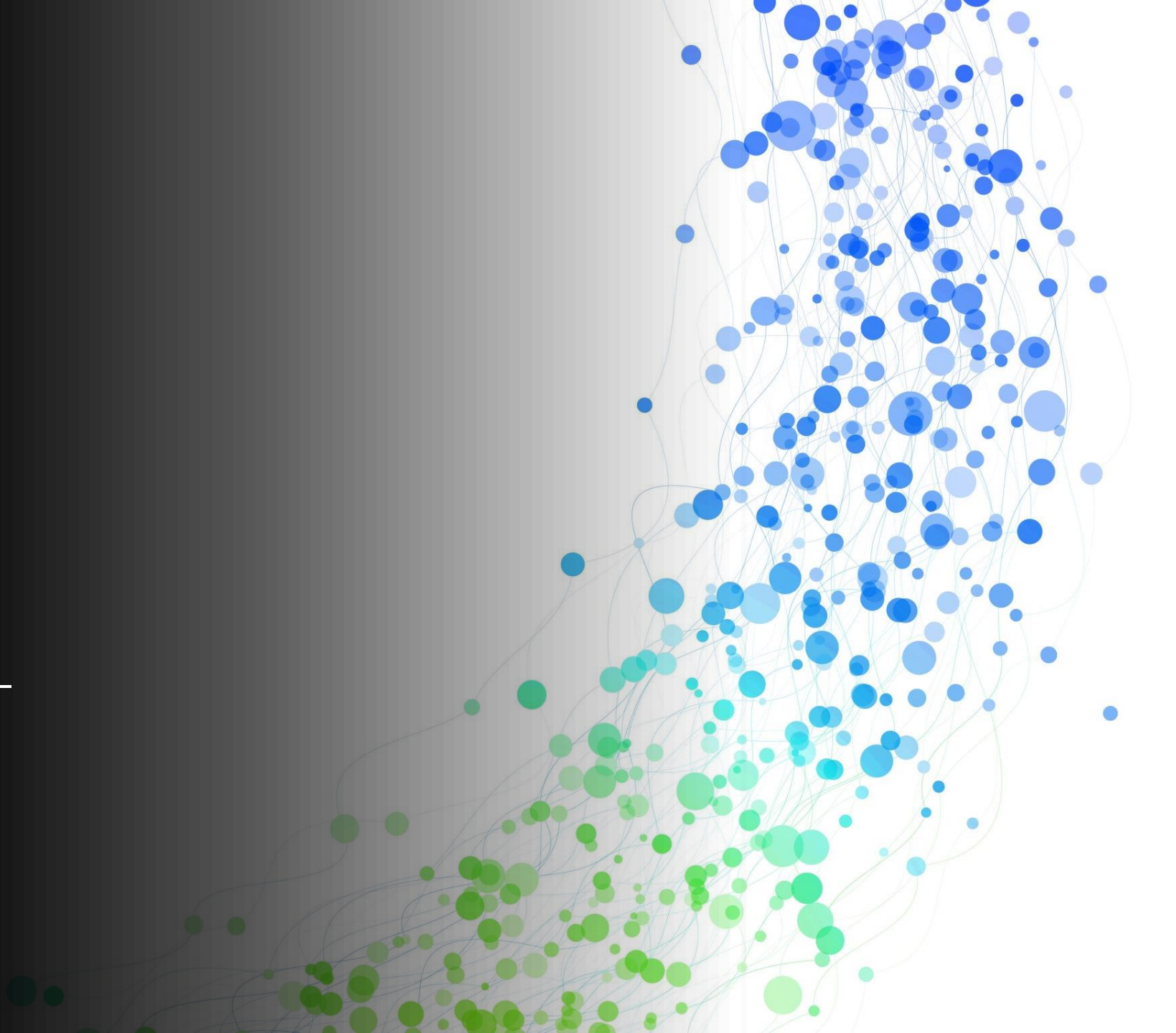




Applied Ontology Engineering: The Missing Pieces

Ontology Summit
March 15, 2023

Karl Hammar





Speaker Credentials

- Worked on:
 - eXtreme Design
 - CoModIDE
 - RealEstateCore
- Previously: Department Head, Jönköping University, Sweden
- Since September: Senior Technical PM, Microsoft



Why ontology?

- Adding functionality to some (software) system
- Broad strokes:
 - Reasoning over knowledge
 - Integrating data

Why does adoption matter?

- Academic success: typically, rational result of competence + well-structured hard work
- Industry success: stochastic; dependent on immeasurables; perception-based
- Why is the latter important?
 - Real-world impact
 - In-situ evaluation opportunities
 - Funding
 - Fun



Tech Industry Adoption Requires DevEx

"Culture trumps strategy"

Successful tech institutions
sport dev cultures

Devs choose technologies

Some criteria:

First run impression

Easy wins – time to first
success

Explorability

Reasonable defaults

Evolution trajectory

My Grievances (A Subset)

Ambiguity in basic node-edge-node modelling

Overuse of global fields

Contextualising statements

Tooling insufficiency

Reusability

Node-Edge-Node modelling



OWL:

1

```
:owns rdf:type owl:ObjectProperty ;  
      rdfs:domain :Person ;  
      rdfs:range :Car .
```

2

```
:Person rdf:type owl:Class ;  
        rdfs:subClassOf [ rdf:type owl:Restriction ;  
                           owl:onProperty :owns ;  
                           owl:someValuesFrom :Car  
                        ] ,  
        [ rdf:type owl:Restriction ;  
          owl:onProperty :owns ;  
          owl:allValuesFrom :Car  
        ] .
```

SHACL

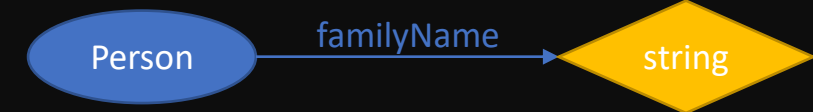
```
:Person  
rdf:type sh:NodeShape ;  
sh:property [  
    rdf:type sh:PropertyShape ;  
    sh:path :owns ;  
    sh:class :Car ;  
    sh:name "owns" ;  
    sh:nodeKind sh:IRI ;  
] ;
```

3

```
sh:property [  
    rdf:type sh:PropertyShape ;  
    sh:path :owns ;  
    sh:name "owns" ;  
    sh:node :Car ;  
    sh:nodeKind sh:IRI ;  
] ;
```

4

Global Fields



example:Person

```
  rdf:type rdfs:Class ;
  rdf:type sh:NodeShape ;
  rdfs:subClassOf rdfs:Resource ;
  sh:property [
    rdf:type sh:PropertyShape ;
    sh:path example:familyName ; ← i.e., https://example.com/ontology#familyName
    sh:datatype xsd:string ;
    sh:name "family name" ;
  ] ;
```

In the large majority of data modelling cases, local fields are sufficient

Why are Global Field Semantics Bad?

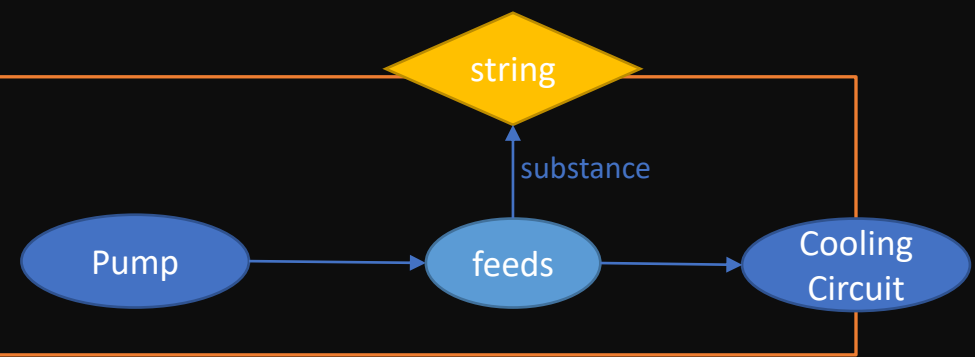
Name
collisions for
ambiguous
but
common
words

- "Name", "Time",
"Member", ...

Lack of type
specific
property
adaptation

- E.g., Equipment
can only have
other Equipment
as part

Statement-level Metadata



- We need to express property graph:isms
- E.g.:
 - Duration of some assignment
 - Substance fed via some interconnection
- Sometimes can be reified somewhat naturally – e.g., “Role”
- Other solutions are hacks:
 - RDF reification vocabulary
 - Singleton property + property/individual punning
 - Subproperty hierarchy abuse
 - Named graphs
- RDF* to the rescue?

Tooling Challenges

- IDE
- Visualization
 - Models and data
- Documentation
- Data store
- Model translation
- Model and data querying
- Versioning
- Packaging

How long to value creation?

© 2014 Pearson Education, Inc. or its affiliate(s). All rights reserved. Pearson Education, Inc., 501 Boylston Street, Boston, MA 02116

Node label

★ (11) **Data**

Relationship Types

* (10) CONTAINS (5)

Displaying 11 nodes, 10 relationships

© 2014 Pearson Education, Inc. or its affiliate(s). All rights reserved.

Node label

* (17) OS (

Router (1) Service (1) Egress (1)

Process (1) Machine (1) Rack (1)

Type (1) DataCenter (1) Switch (1)

Version (1) Network (1)

Software (1) Application (1)

Interface (1)

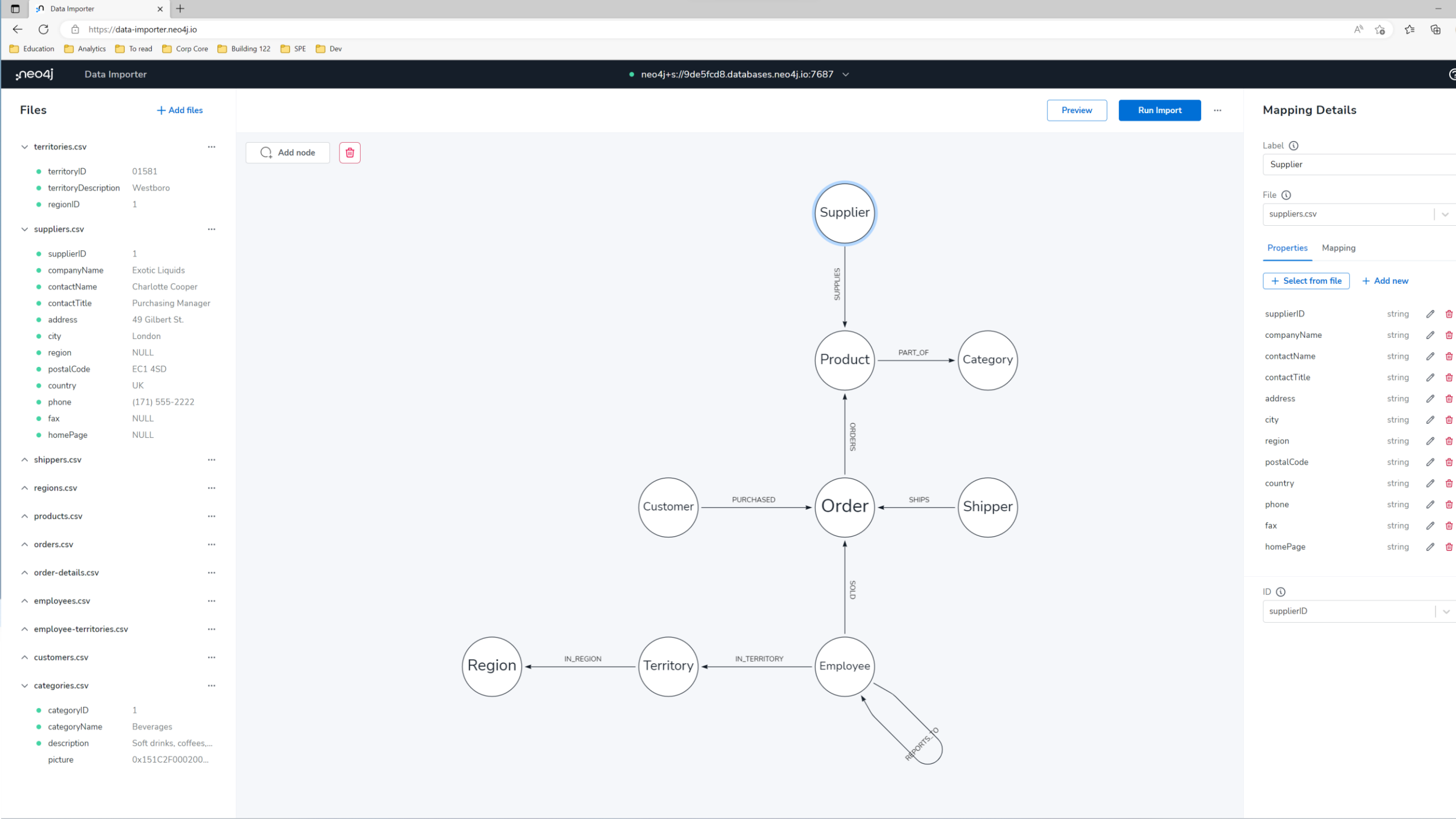
Relationship Types

* (63) ROUTES (6) VERSION (3)

DEPENDS_ON (20) LISTENS (3)

CONNECTS (3) INSTANCE (16)

Downloaded from ascelibrary.org by New York University on 06/01/15. Copyright ASCE, For All Rights Reserved, No part of this document may be reproduced without written permission from ASCE.



Tools are
out there –
but poorly
integrated

Ontology + data graph
integration

GUI integration

Workflow integration

Multi-user integration

Reuse Challenges

Reusable components need to be large enough to not be obvious, yet small enough to not be overwhelming.

Good examples:

- Martin Fowler's Analysis Patterns
- Building Topology Ontology
- FOAF

The importance is the conceptual solution, NOT the exact serialization.

- Diagrams rather than OWL!

Serialization doesn't matter



Addressing the Problems

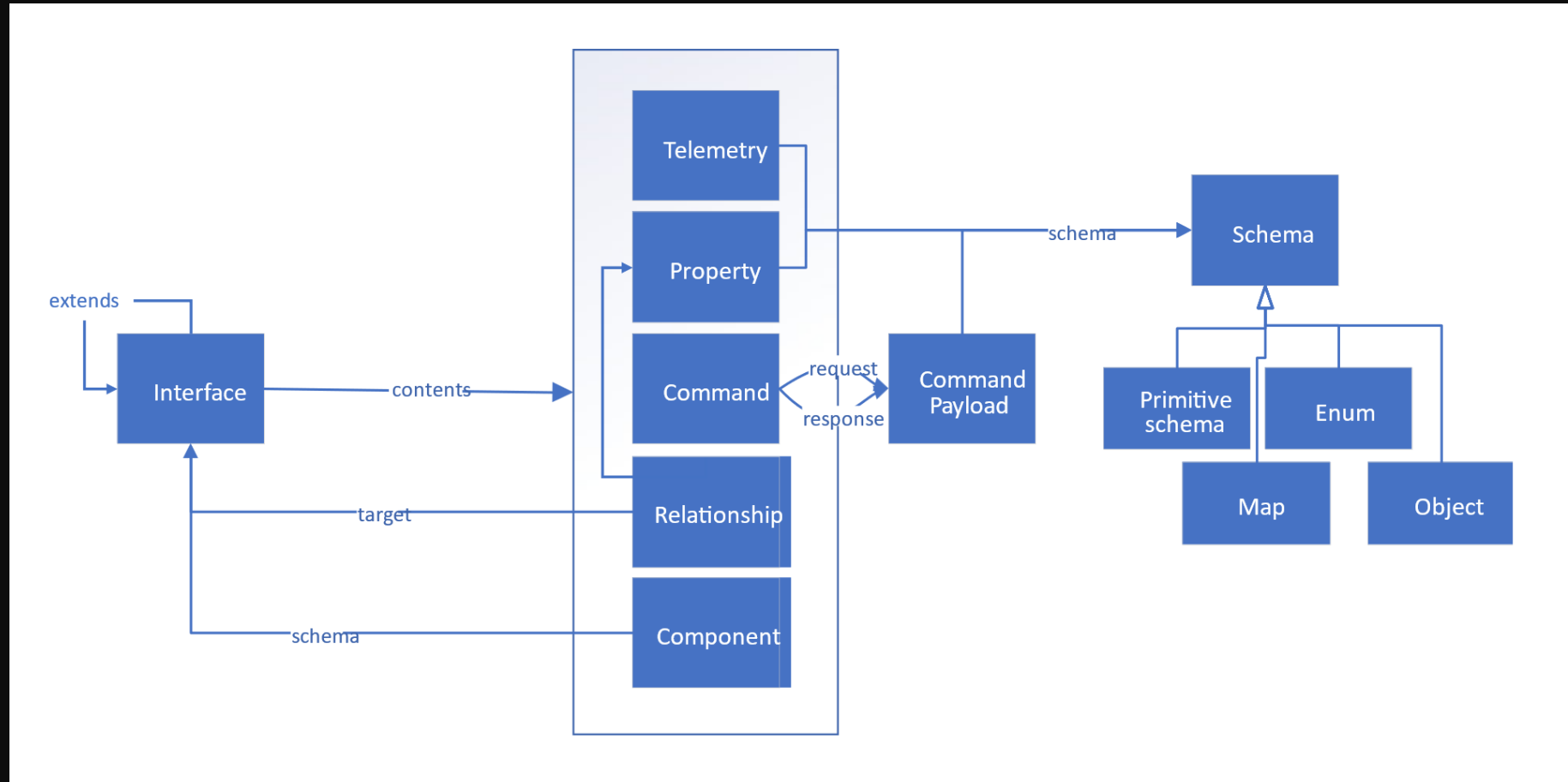


Digital Twin Definition Language -- DTDL

```
{
  "@id": "dtmi:com:example:Room;1",
  "@type": "Interface",
  "contents": [
    {
      "@type": "Property",
      "name": "occupied",
      "schema": "boolean",
      "@id": "dtmi:example:occ;1"
    }
  ],
  "@context": "dtmi:dtdl:context;2"
}
```

```
{
  "@id": "dtmi:com:example:ConferenceRoom;1",
  "@type": "Interface",
  "extends": "dtmi:com:example:Room;1",
  "contents": [
    {
      "@type": "Property",
      "name": "capacity",
      "schema": "integer"
    },
    {
      "@type": "Relationship",
      "name": "bookingResponsible",
      "target": "dtmi:com:example:Person;1"
    }
  ],
  "@context": "dtmi:dtdl:context;2"
}
```

Digital Twin Definition Language -- DTDL



Digital Twin Definition Language -- DTDL

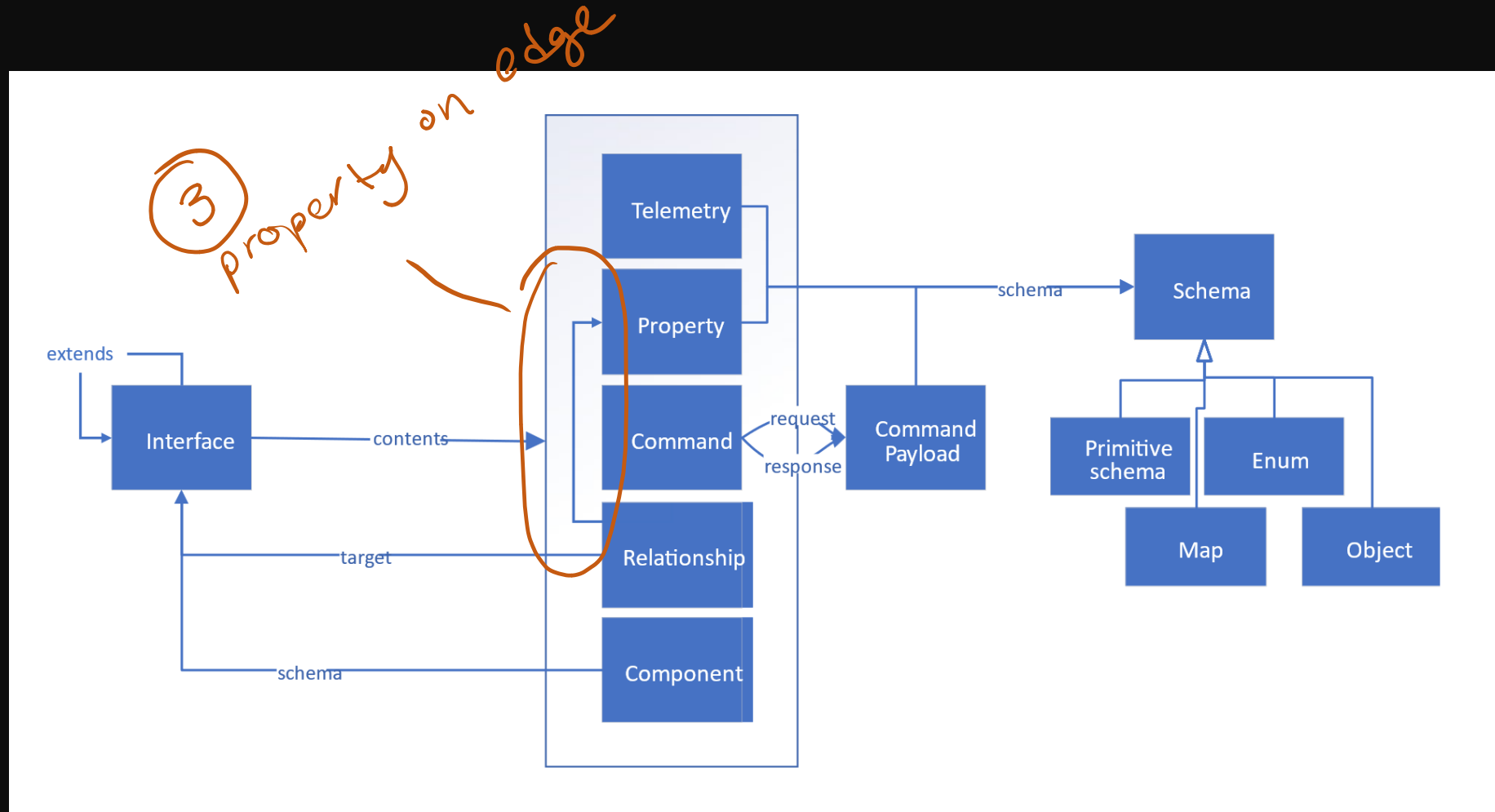
```
{
  "@id": "dtmi:com:example:Room;1",
  "@type": "Interface",
  "contents": [
    {
      "@type": "Property",
      "name": "occupied",
      "schema": "boolean",
      "@id": "dtmi:example:occ;1"
    }
  ],
  "@context": "dtmi:dtdl:context;2"
}
```

```
{
  "@id": "dtmi:com:example:ConferenceRoom;1",
  "@type": "Interface",
  "extends": "dtmi:com:example:Room;1",
  "contents": [
    {
      "@type": "Property",
      "name": "capacity",
      "schema": "integer"
    },
    {
      "@type": "Relationship",
      "name": "bookingResponsible",
      "target": "dtmi:com:example:Person;1"
    }
  ],
  "@context": "dtmi:dtdl:context;2"
}
```

② local field

① node - edge - node

Digital Twin Definition Language -- DTDL



Open Challenges

Are these anecdotes about language design needs borne out in practice?

Reuse challenges

- In-depth analysis of patterns/ontologies and their usability
- Creating a sufficient corpus

Tooling

- Cost of building
- Longevity of maintenance

Fin