Design Patterns 102

John Beverley

Assistant Professor, *University at Buffalo*
Co-Director, National Center for Ontological Research
Affiliate Faculty, *Institute of Artificial Intelligence and Data Science*

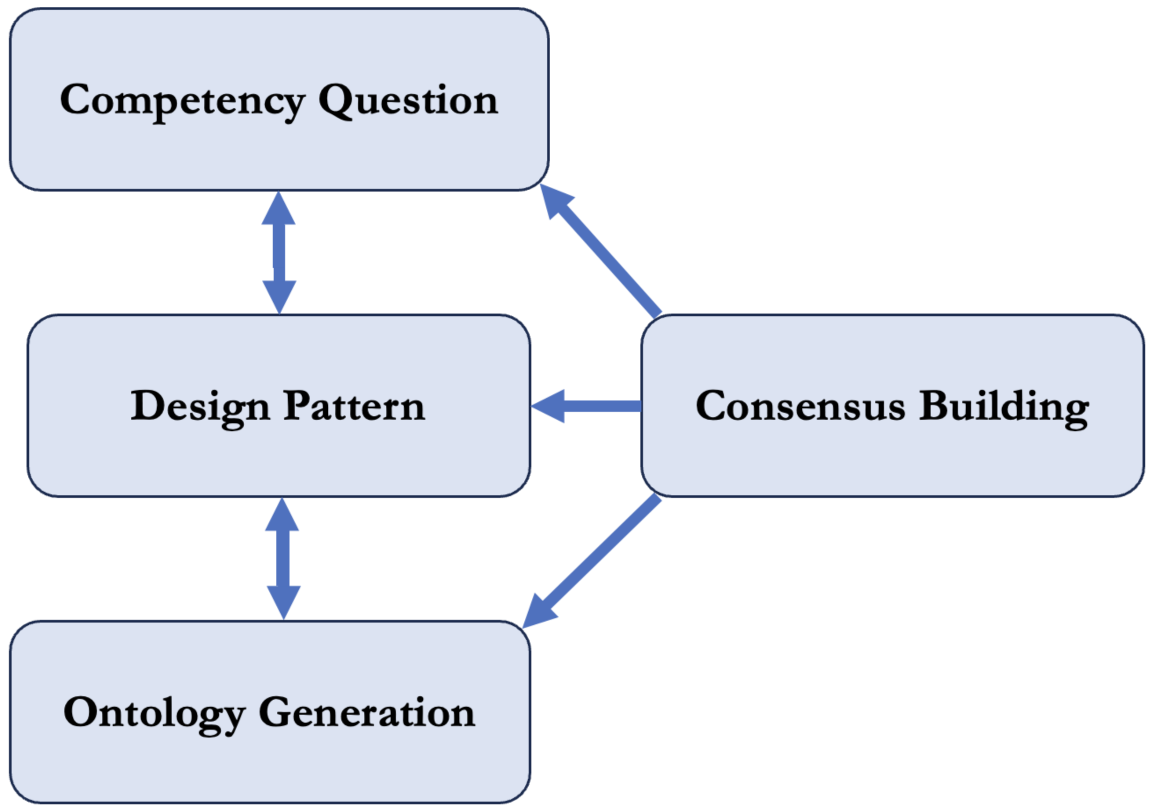
Outline

- Design Patterns Guidance
- Mermaid
- Guardrails
- Complex vs Simplified Design Patterns

Outline

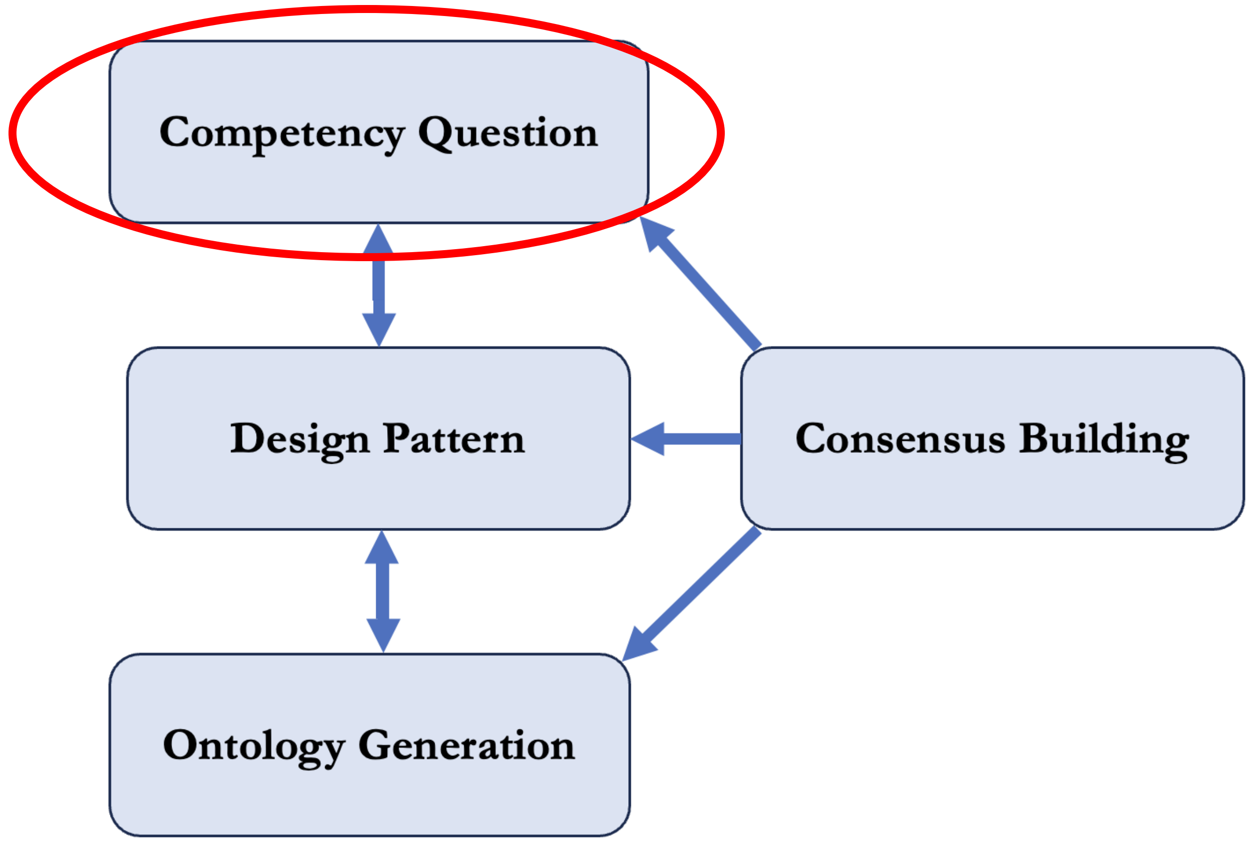
- Design Patterns Guidance
- Mermaid
- Guardrails
- Complex vs Simplified Design Patterns

Guidance


Competency Question
Design Pattern
Consensus Building
Ontology Generation

Guidance

- Identify competency questions



Competency Questions

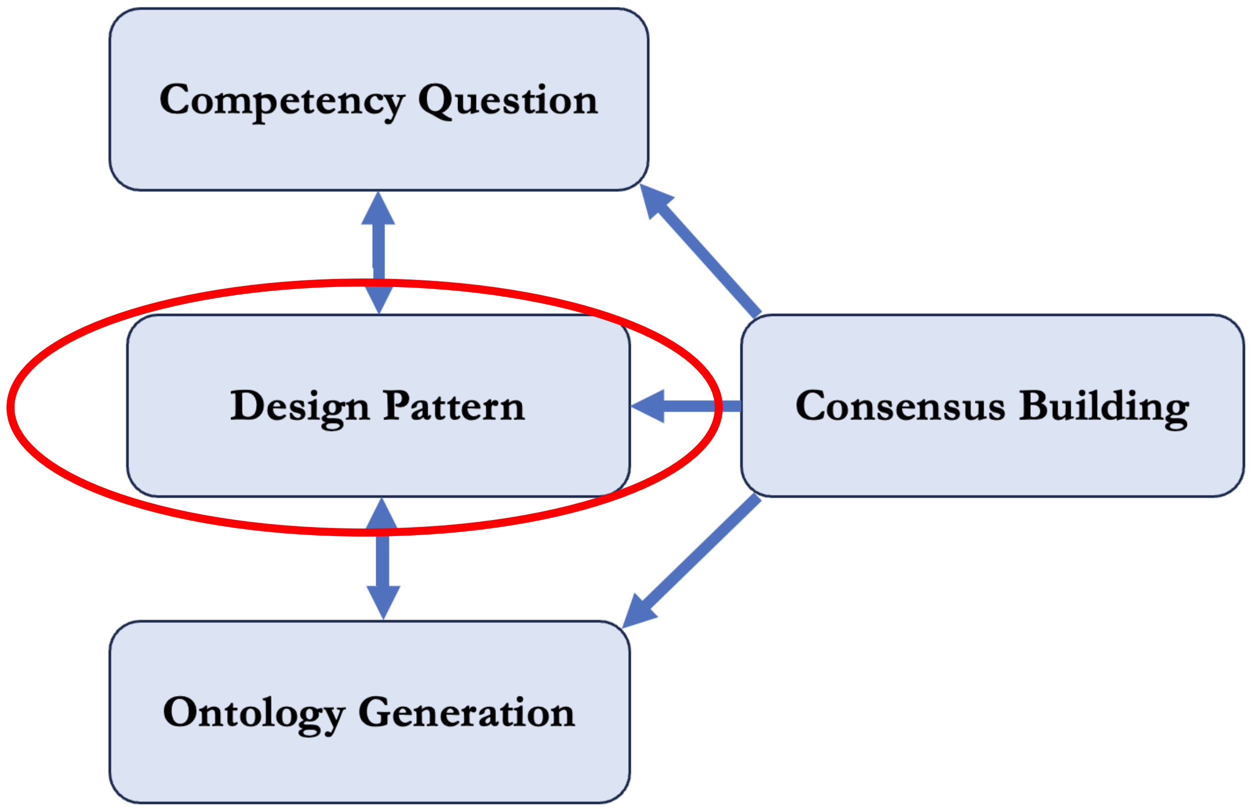
- Competency questions are **used to guide ontology development** and **generate automated checks** to ensure answers are sufficient

- In our drone tracking example we needed to represent:

border/boundary	latitude
geographic region	altitude
drone	longitude
speed	time

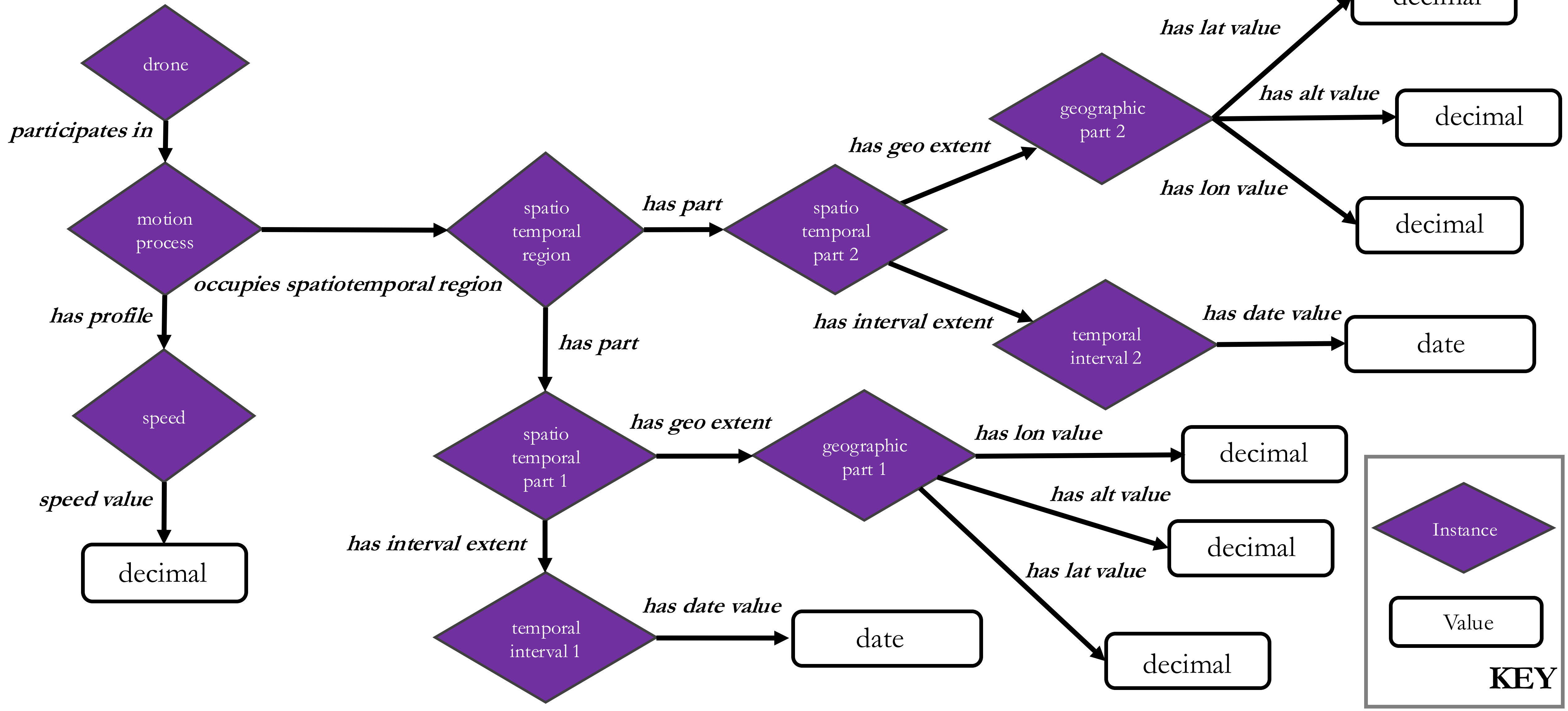
Guidance

- Identify competency questions
- Generate design patterns to address competency questions and inform ontology creation



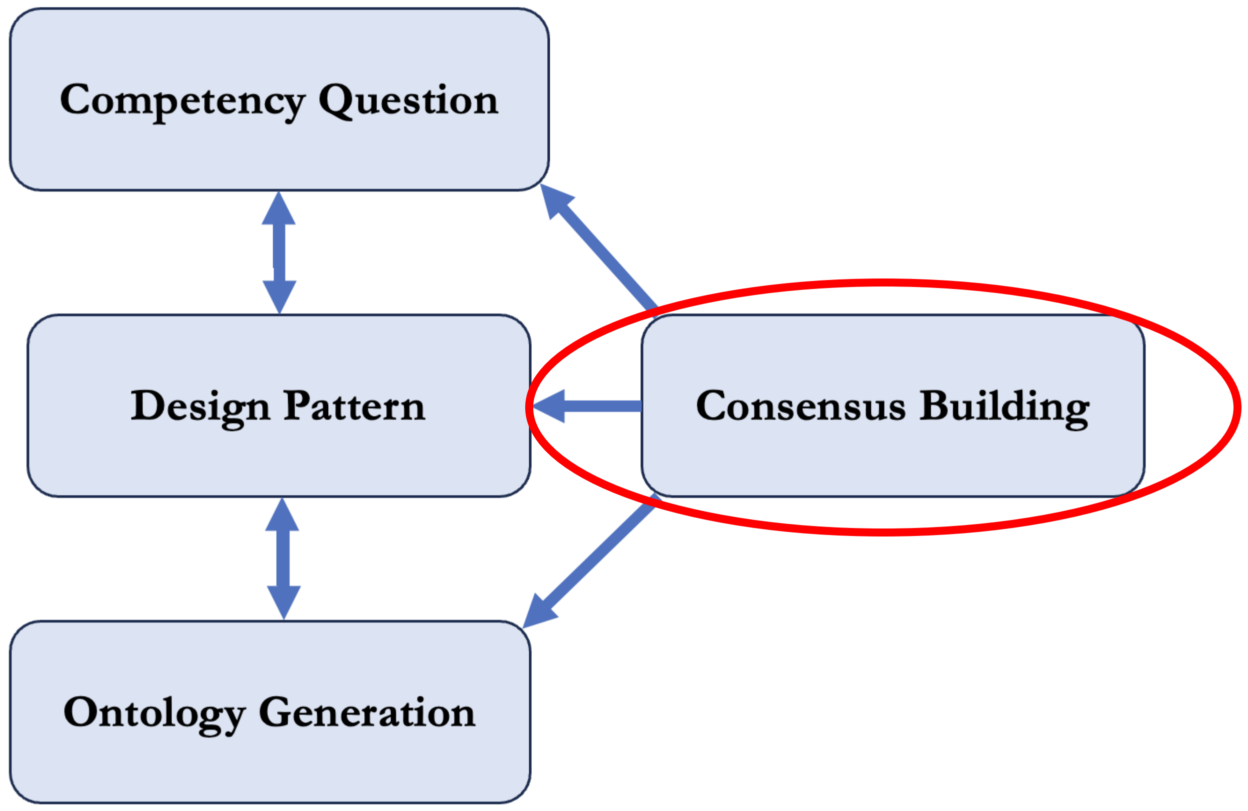
*Classes not diagramed due to space constraints

Representing Drone Motion



Guidance

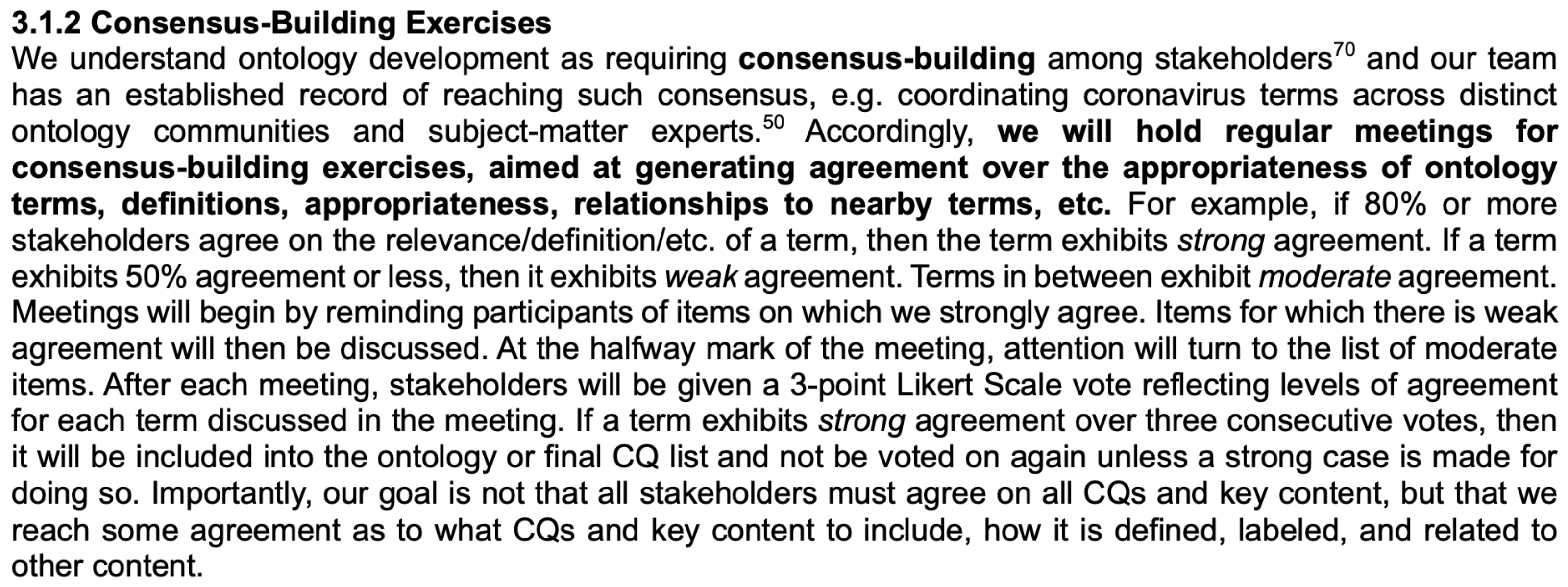
- Identify competency questions
- Generate design patterns to address competency questions and inform ontology creation
- Revise competency questions, design patterns, and ontology through consensus-building exercises



Consensus-Building Exercises

- Consensus-building exercises are where ontologists and domain experts work towards an agreed understanding of ontology terms, definitions, etc.

- Importantly, whatever agreement is reached is **meant to be added to the ontology**; domain experts can **continue speaking as they need**



3.1.2 Consensus-Building Exercises
We understand ontology development as requiring **consensus-building** among stakeholders[70] and our team has an established record of reaching such consensus, e.g. coordinating coronavirus terms across distinct ontology communities and subject-matter experts.[50] Accordingly, **we will hold regular meetings for consensus-building exercises, aimed at generating agreement over the appropriateness of ontology terms, definitions, appropriateness, relationships to nearby terms, etc.** For example, if 80% or more stakeholders agree on the relevance/definition/etc. of a term, then the term exhibits *strong* agreement. If a term exhibits 50% agreement or less, then it exhibits *weak* agreement. Terms in between exhibit *moderate* agreement. Meetings will begin by reminding participants of items on which we strongly agree. Items for which there is weak agreement will then be discussed. At the halfway mark of the meeting, attention will turn to the list of moderate items. After each meeting, stakeholders will be given a 3-point Likert Scale vote reflecting levels of agreement for each term discussed in the meeting. If a term exhibits *strong* agreement over three consecutive votes, then it will be included into the ontology or final CQ list and not be voted on again unless a strong case is made for doing so. Importantly, our goal is not that all stakeholders must agree on all CQs and key content, but that we reach some agreement as to what CQs and key content to include, how it is defined, labeled, and related to other content.

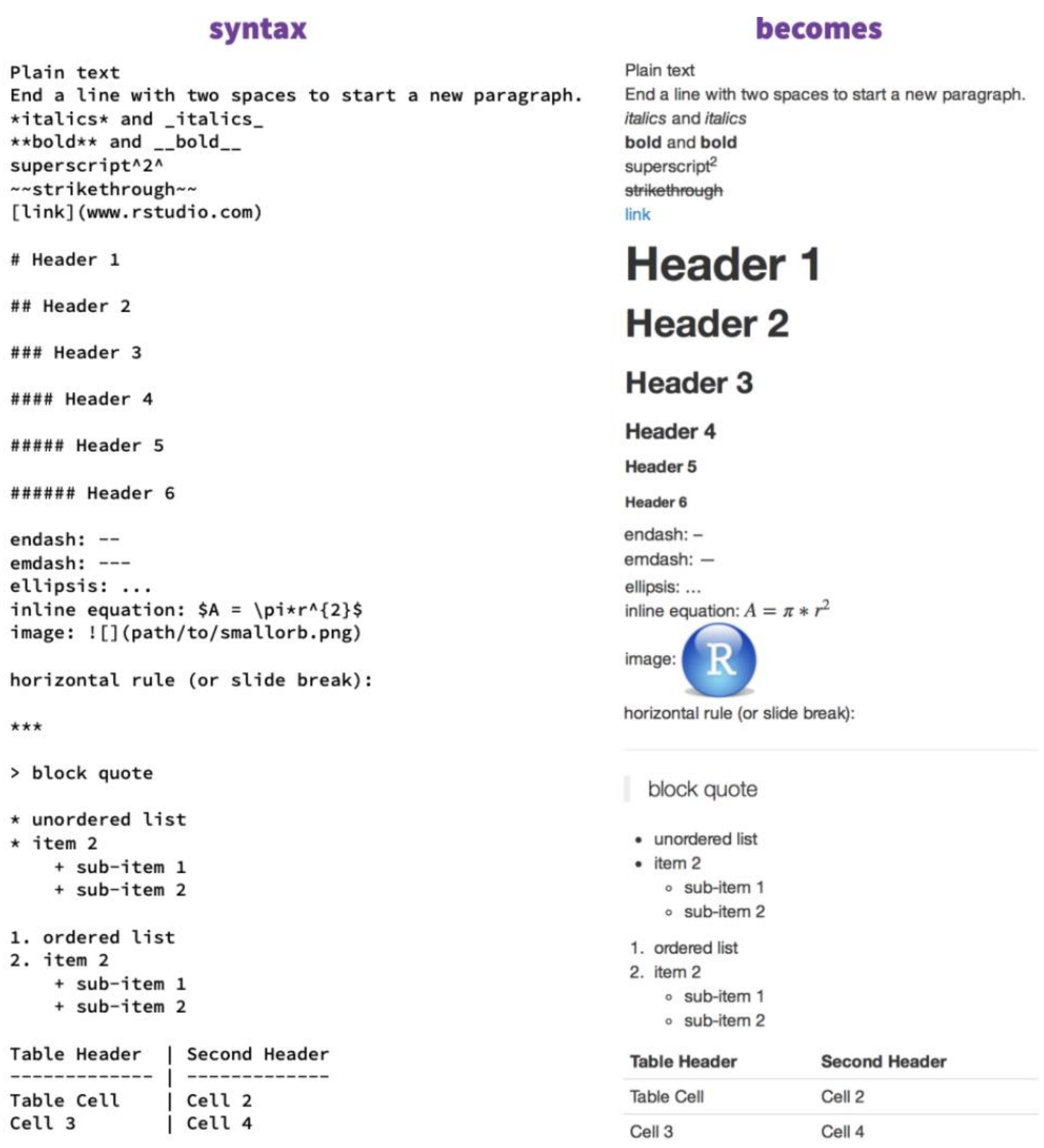
Outline

- Design Patterns Guidance
- Mermaid
- Guardrails
- Complex vs Simplified Design Patterns

Markdown

- Markdown is a popular **markup language** (that is, a text encoding system, such as HTML)

- You’ll be using markdown in GitHub to format issues, posts, responses, etc.



syntax

```
Plain text
End a line with two spaces to start a new paragraph.
*italics* and _italics_
**bold** and __bold__
superscript^2^
~~strikethrough~~
[link](www.rstudio.com)

# Header 1

## Header 2

### Header 3

#### Header 4

##### Header 5

###### Header 6

endash: --
emdash: ---
ellipsis: ...
inline equation: $A = \pi*r^{2}$
image: ![](path/to/smallorb.png)

horizontal rule (or slide break):

***

> block quote

* unordered list
* item 2
    + sub-item 1
    + sub-item 2

1. ordered list
2. item 2
    + sub-item 1
    + sub-item 2

Table Header  | Second Header
------------- | -------------
Table Cell    | Cell 2
Cell 3        | Cell 4
```

becomes

Plain text
End a line with two spaces to start a new paragraph.
italics and *italics*
bold and **bold**
superscript^2
~~strikethrough~~
link

Header 1

Header 2

Header 3

Header 4

Header 5

Header 6

endash: –
emdash: —
ellipsis: …
inline equation: $A = \pi * r^2$
image: R

horizontal rule (or slide break):

> block quote

- unordered list
- item 2
    - sub-item 1
    - sub-item 2

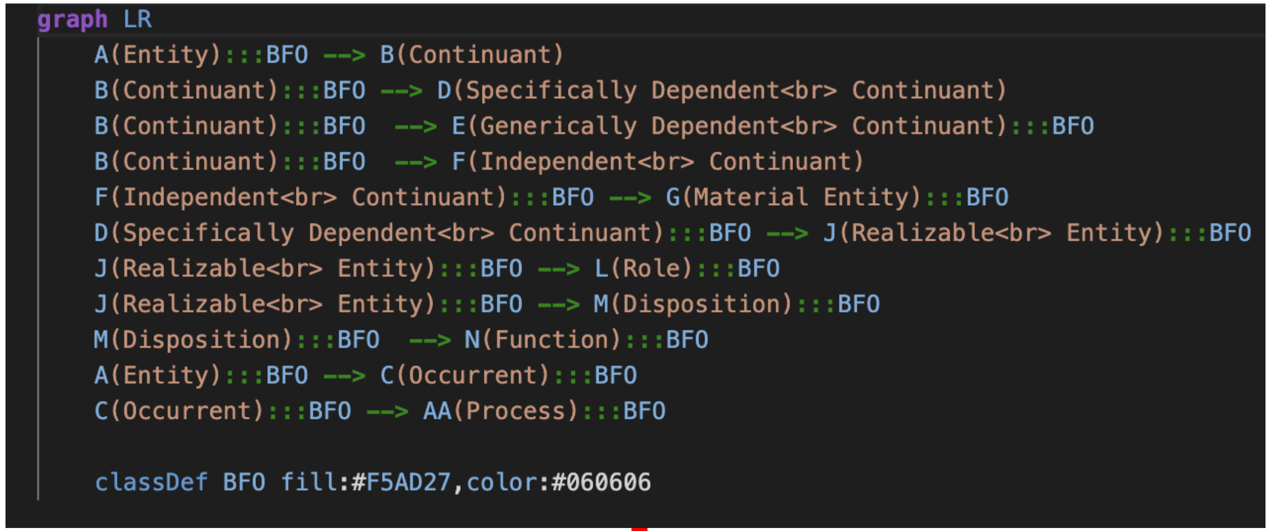
1. ordered list
2. item 2
    - sub-item 1
    - sub-item 2

Table Header	Second Header
Table Cell	Cell 2
Cell 3	Cell 4

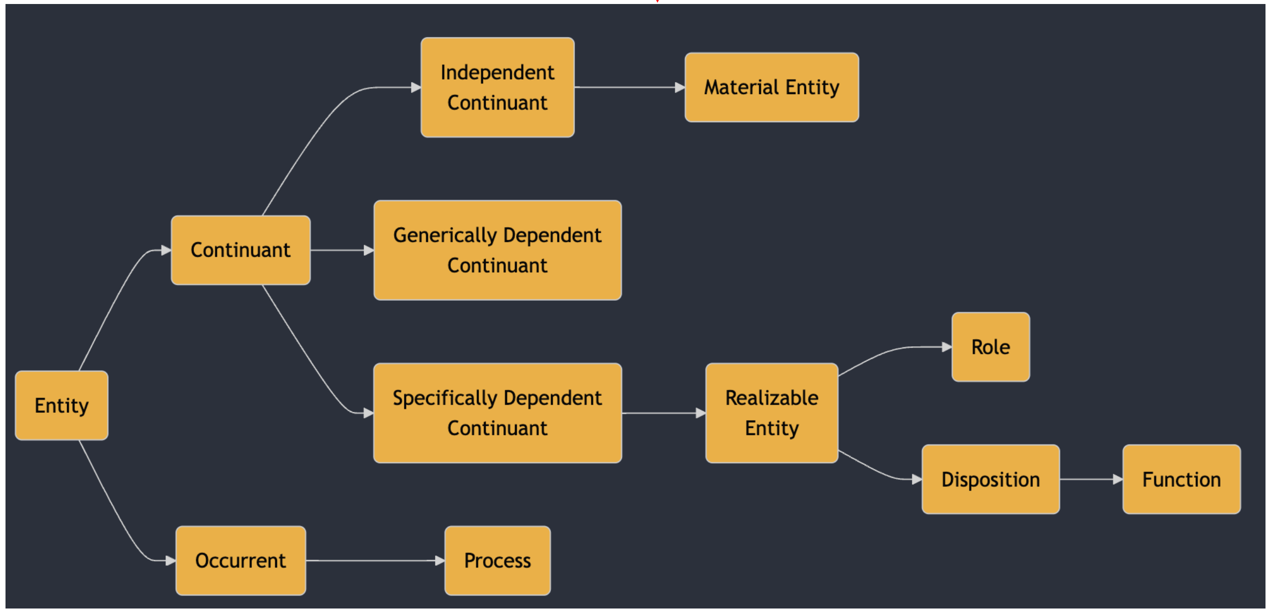
Mermaid

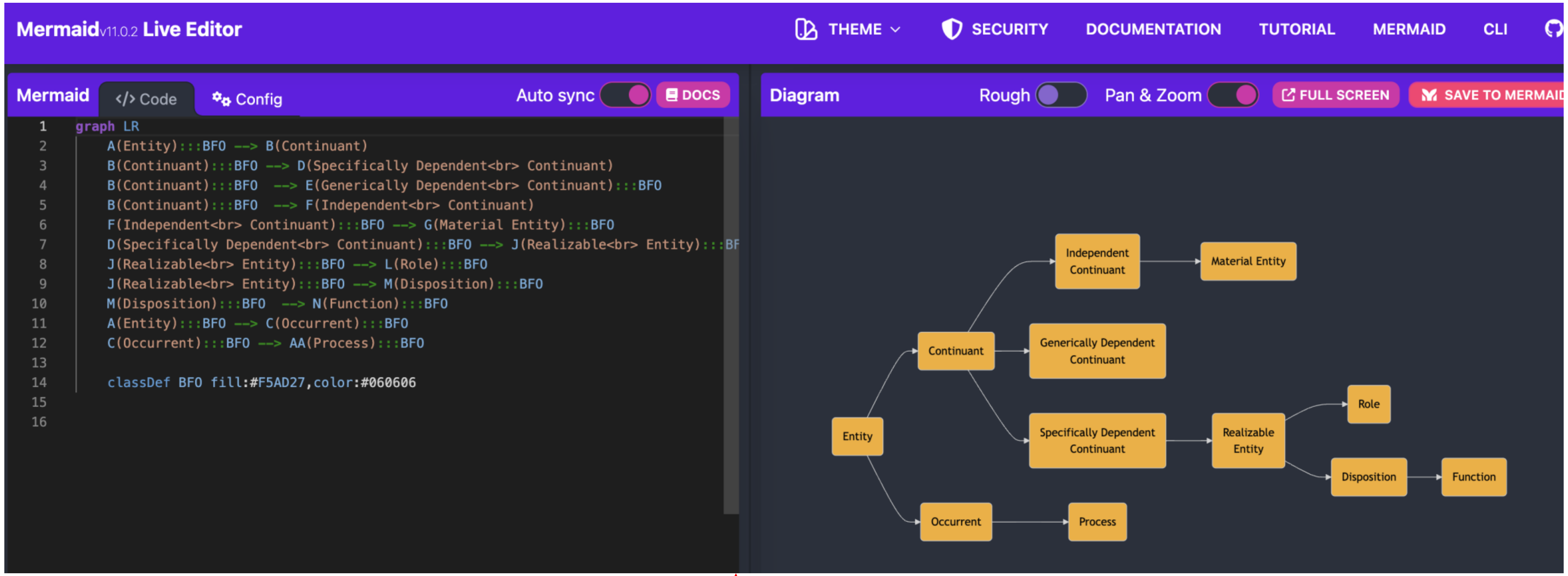
- Mermaid is a flavor of Markdown specialized for representing diagrams

- You'll be using mermaid to represent design patterns in this course

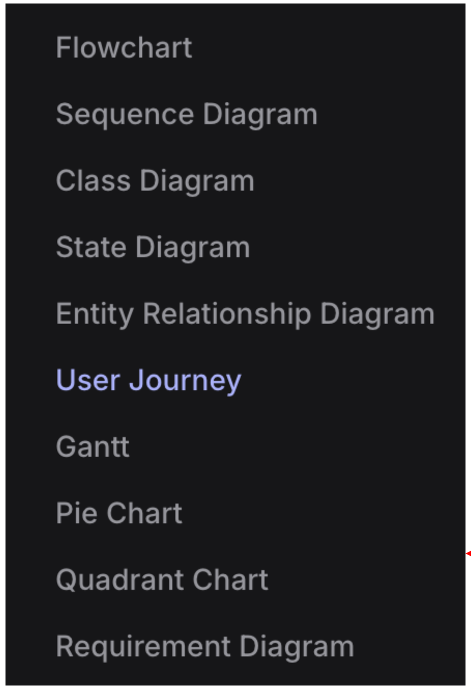


```
graph LR
    A(Entity):::BFO --> B(Continuant)
    B(Continuant):::BFO --> D(Specifically Dependent<br> Continuant)
    B(Continuant):::BFO  --> E(Generically Dependent<br> Continuant):::BFO
    B(Continuant):::BFO  --> F(Independent<br> Continuant)
    F(Independent<br> Continuant):::BFO --> G(Material Entity):::BFO
    D(Specifically Dependent<br> Continuant):::BFO --> J(Realizable<br> Entity):::BFO
    J(Realizable<br> Entity):::BFO --> L(Role):::BFO
    J(Realizable<br> Entity):::BFO --> M(Disposition):::BFO
    M(Disposition):::BFO  --> N(Function):::BFO
    A(Entity):::BFO --> C(Occurrent):::BFO
    C(Occurrent):::BFO --> AA(Process):::BFO

    classDef BFO fill:#F5AD27,color:#060606
```





```
graph LR
    A(Entity):::BFO --> B(Continuant)
    B(Continuant):::BFO --> D(Specifically Dependent<br> Continuant)
    B(Continuant):::BFO  --> E(Generically Dependent<br> Continuant):::BFO
    B(Continuant):::BFO  --> F(Independent<br> Continuant)
    F(Independent<br> Continuant):::BFO --> G(Material Entity):::BFO
    D(Specifically Dependent<br> Continuant):::BFO --> J(Realizable<br> Entity):::BF
    J(Realizable<br> Entity):::BFO --> L(Role):::BFO
    J(Realizable<br> Entity):::BFO --> M(Disposition):::BFO
    M(Disposition):::BFO  --> N(Function):::BFO
    A(Entity):::BFO --> C(Occurrent):::BFO
    C(Occurrent):::BFO --> AA(Process):::BFO

    classDef BFO fill:#F5AD27,color:#060606
```

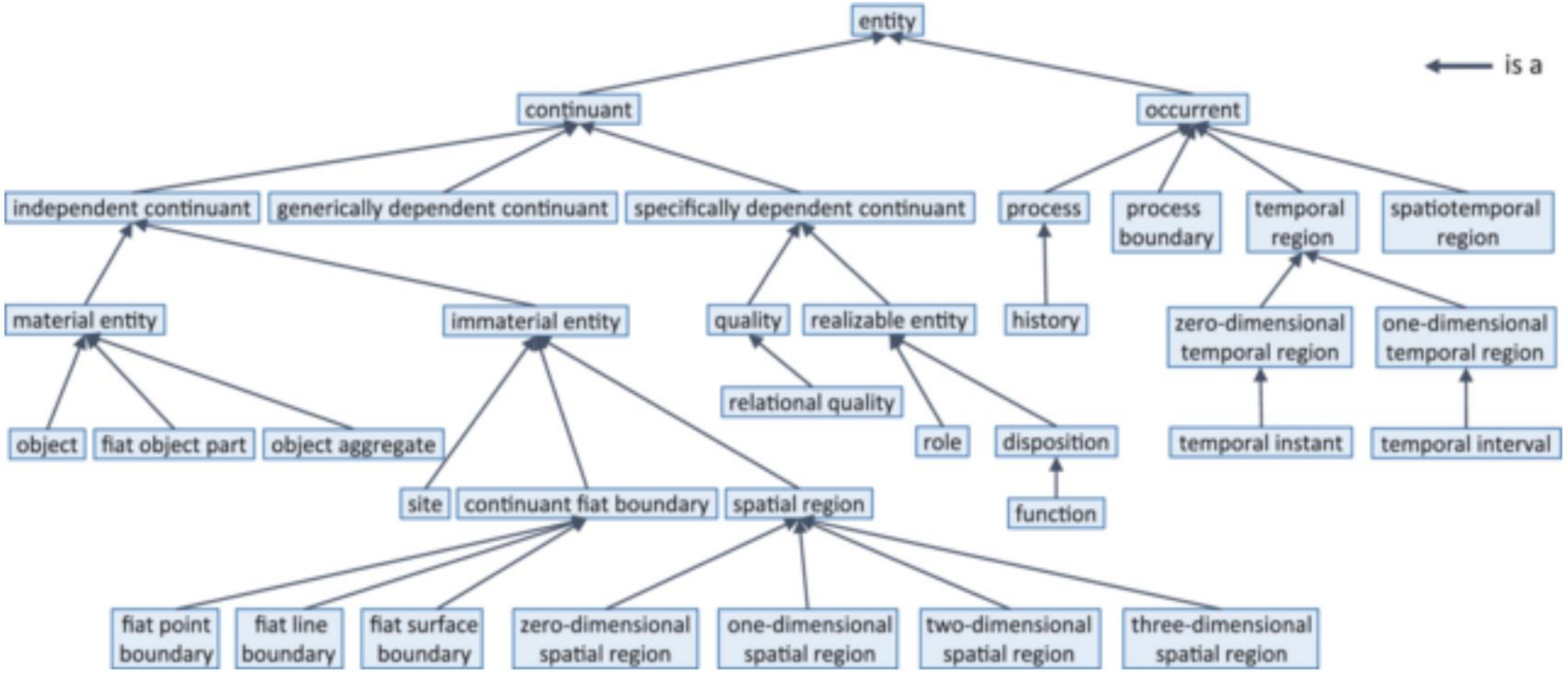

Flowchart
Sequence Diagram
Class Diagram
State Diagram
Entity Relationship Diagram
User Journey
Gantt
Pie Chart
Quadrant Chart
Requirement Diagram

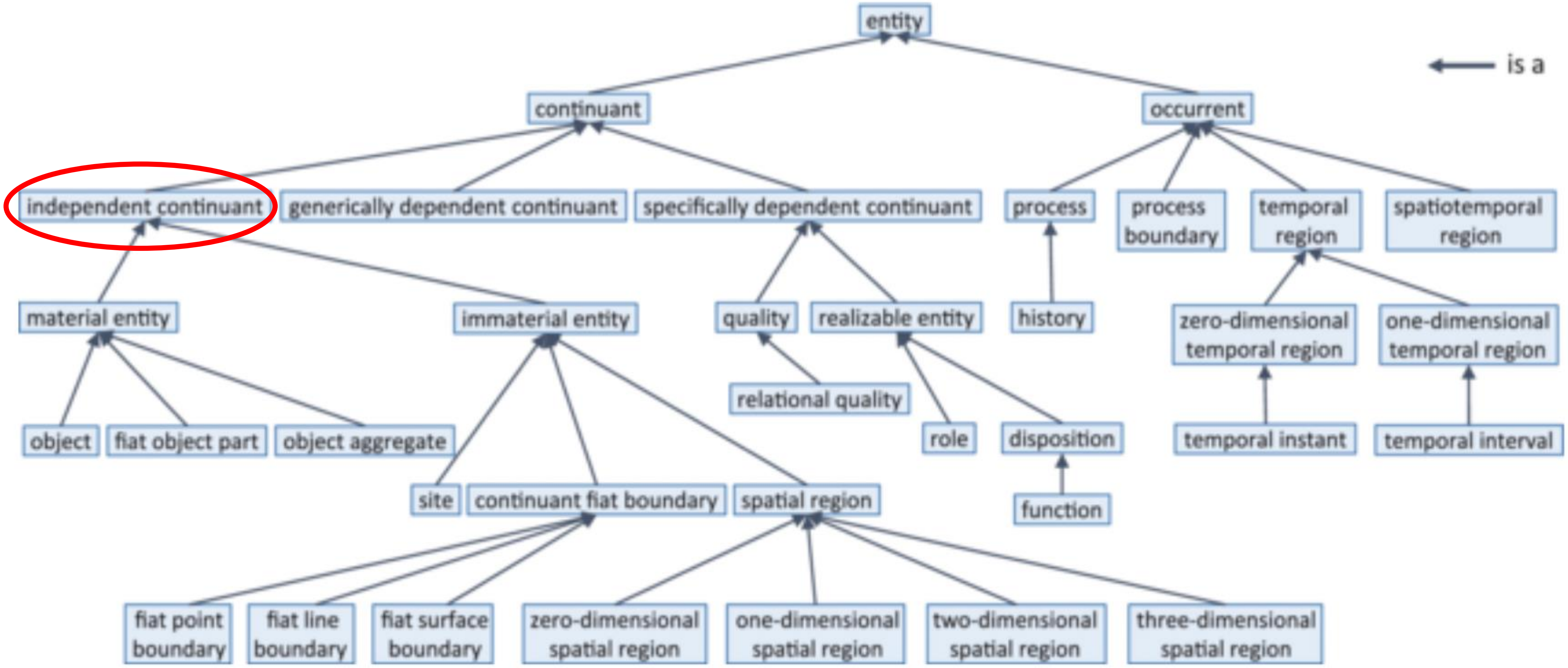
Play around in the live editor:
https://mermaid.live/edit

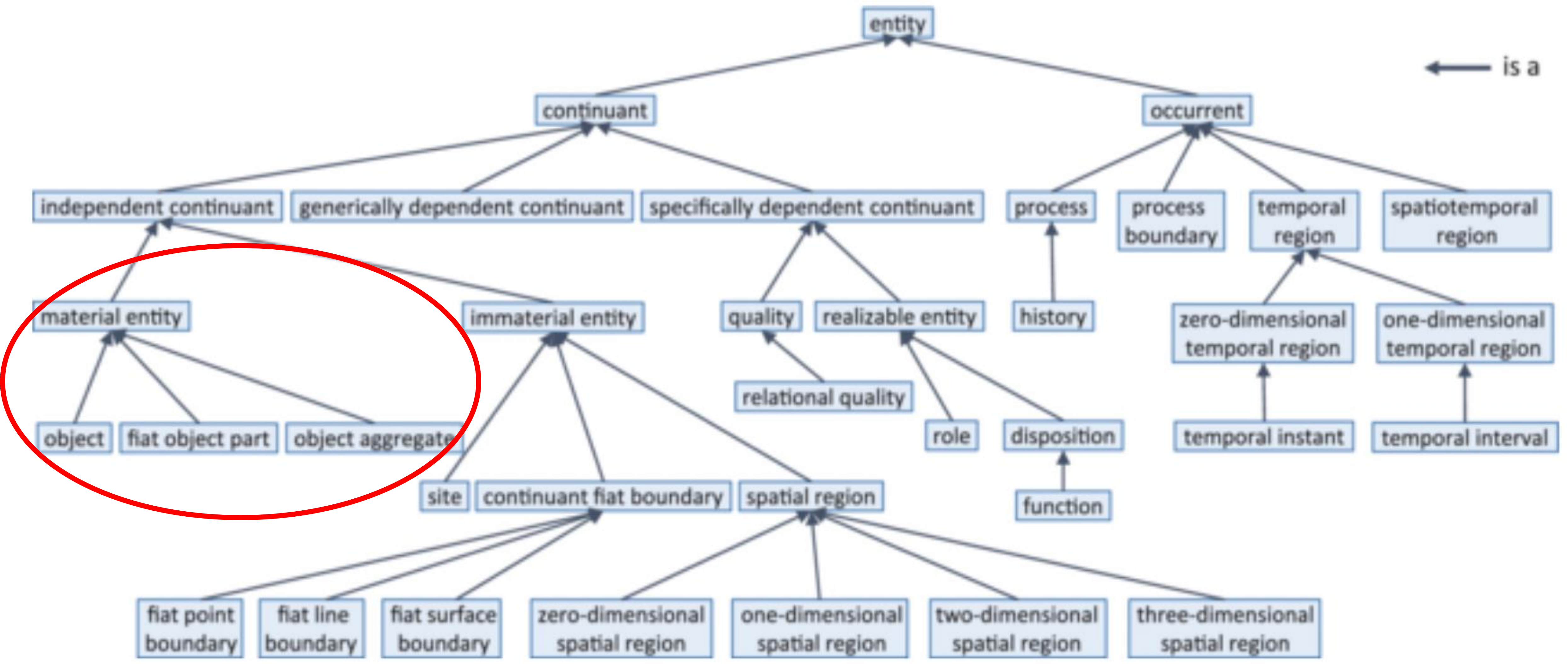
Navigate to the helpful tutorials:
https://mermaid.js.org/ecosystem/tutorials.html

Outline

- Design Patterns Guidance
- Mermaid
- Guardrails
- Complex vs Simplified Design Patterns

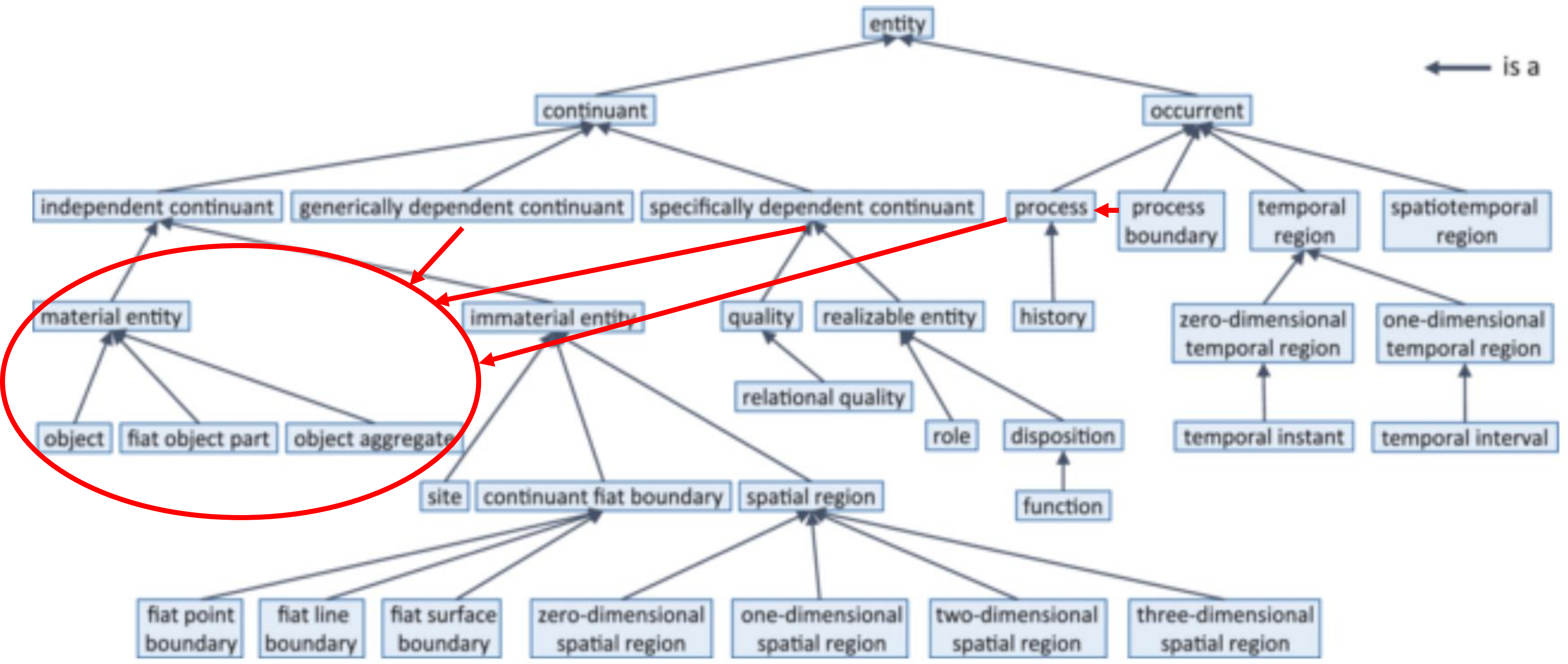

entity
is a
continuant
occurrent
independent continuant
generically dependent continuant
specifically dependent continuant
process
process boundary
temporal region
spatiotemporal region
material entity
immaterial entity
quality
realizable entity
history
zero-dimensional temporal region
one-dimensional temporal region
relational quality
object
fiat object part
object aggregate
role
disposition
temporal instant
temporal interval
site
continuant fiat boundary
spatial region
function
fiat point boundary
fiat line boundary
fiat surface boundary
zero-dimensional spatial region
one-dimensional spatial region
two-dimensional spatial region
three-dimensional spatial region


entity
is a
continuant
occurrent
independent continuant
generically dependent continuant
specifically dependent continuant
process
process boundary
temporal region
spatiotemporal region
material entity
immaterial entity
quality
realizable entity
history
zero-dimensional temporal region
one-dimensional temporal region
relational quality
object
fiat object part
object aggregate
role
disposition
temporal instant
temporal interval
site
continuant fiat boundary
spatial region
function
fiat point boundary
fiat line boundary
fiat surface boundary
zero-dimensional spatial region
one-dimensional spatial region
two-dimensional spatial region
three-dimensional spatial region


entity
is a
continuant
occurrent
independent continuant
generically dependent continuant
specifically dependent continuant
process
process boundary
temporal region
spatiotemporal region
material entity
immaterial entity
quality
realizable entity
history
zero-dimensional temporal region
one-dimensional temporal region
relational quality
object
fiat object part
object aggregate
role
disposition
temporal instant
temporal interval
site
continuant fiat boundary
spatial region
function
fiat point boundary
fiat line boundary
fiat surface boundary
zero-dimensional spatial region
one-dimensional spatial region
two-dimensional spatial region
three-dimensional spatial region

Eleatic Principle

- The scope of BFO is space, time, and anything in space and time
- As a litmus test, anything in space and time must have some causal impact on other entities in space and time
- Material entities are paradigmatically in space
- Nearly everything in BFO traces its dependence back to material entities


entity
is a
continuant
occurrent
independent continuant
generically dependent continuant
specifically dependent continuant
process
process boundary
temporal region
spatiotemporal region
material entity
immaterial entity
quality
realizable entity
history
zero-dimensional temporal region
one-dimensional temporal region
relational quality
object
fiat object part
object aggregate
role
disposition
temporal instant
temporal interval
site
continuant fiat boundary
spatial region
function
fiat point boundary
fiat line boundary
fiat surface boundary
zero-dimensional spatial region
one-dimensional spatial region
two-dimensional spatial region
three-dimensional spatial region

Eleatic Principle

- Generically dependent continuants are **concretized in** specifically dependent continuants

- Processes realize certain specifically dependent continuants

- Specifically dependent continuants are **borne by** independent continuants, such as material entities

Rules of Thumb

- When building a design pattern, describe:

1. Material entities within scope, i.e. **Material Entity**
2. Qualities these material entities have, i.e. **Quality**
3. What these material entities can do, i.e. **Process**
4. What properties underwrite what they can do, i.e. **Realizable Entity**
5. Information we use to talk about 1-4, i.e. **Generically Depedent Continuant**

Exercise

- Suppose you've been given the following competency question to answer:

How quickly are shipping containers transported by Wal-Mart's truck fleet over a given fiscal quarter?

- Leveraging the BFO hierarchy, work in groups to provide a design pattern that aims to answer this question

Outline

- Design Patterns Guidance
- Mermaid
- Guardrails
- Complex vs Simplified Design Patterns

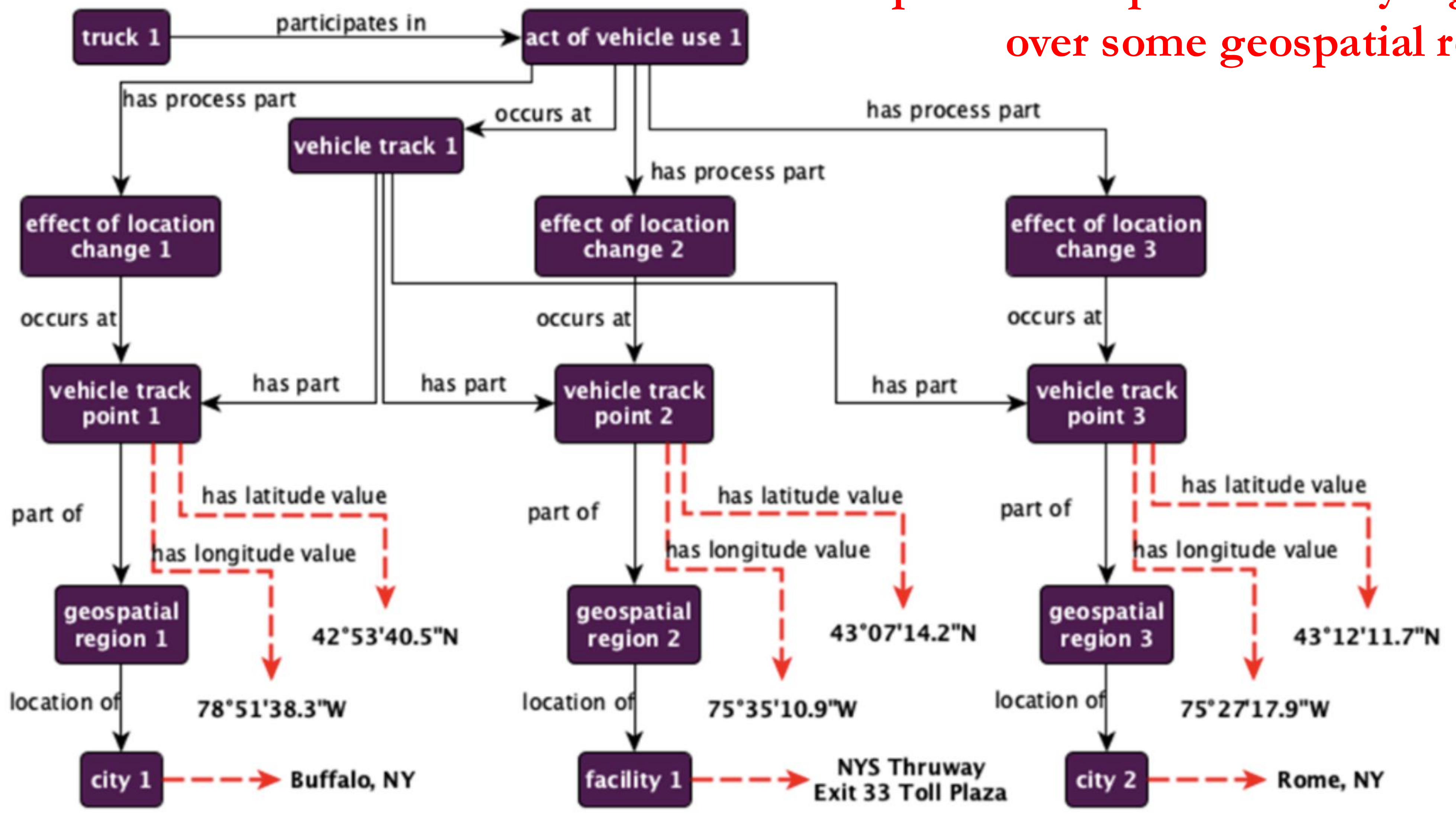
Complex vs Simplified

- Ontology engineers must often work with developers who need to deploy our work
- Developers do not always need the full ontology design patterns or ontologies that we create
- We of course create full patterns where we can as a way to future-proof our projects
- In the short-term however, we must be able to give developers what they need

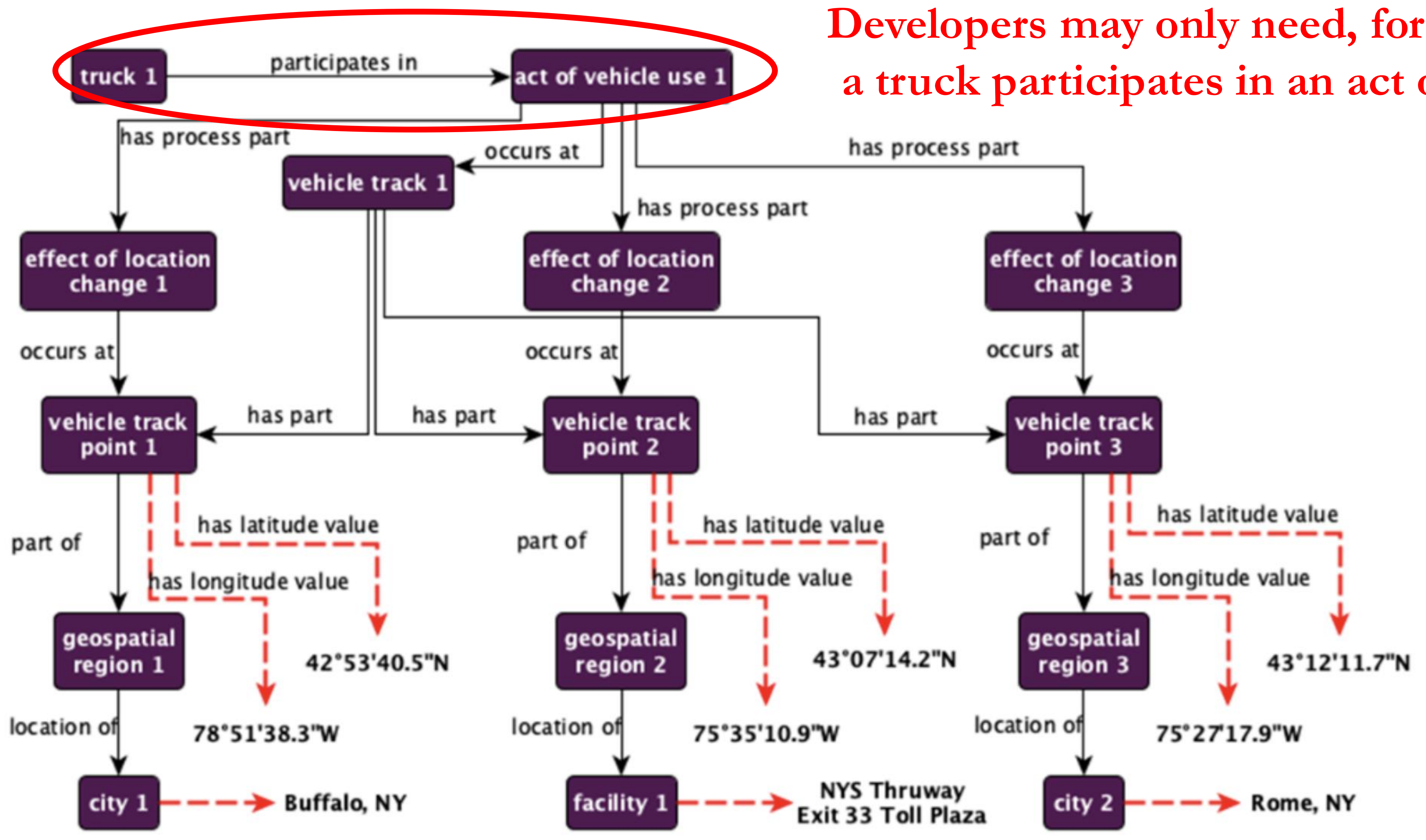
Complex vs Simplified

- Call the full pattern ontology engineers are trained to generate a **Complex Design Pattern (CDPs)**
- Call sub-graphs of Complex Design patterns that aim to addressing specific user needs **Simplified Design Patterns (SDPs)**
- Our task is to show how to construct SDPs from CDPs, so we maintain a connection from what developer’s need now to what they might need in the future

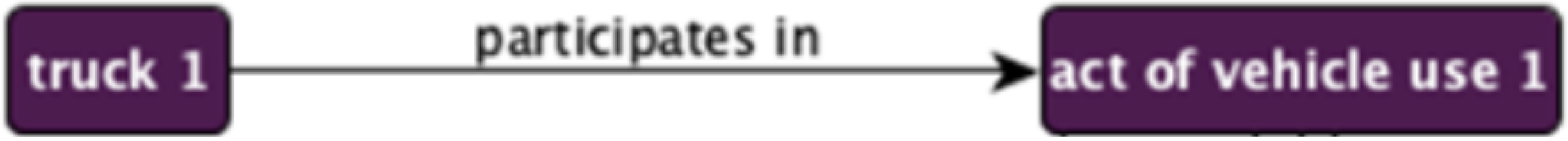
Represent the path taken by a ground vehicle over some geospatial region.



Developers may only need, for example, that a truck participates in an act of vehicle use



If so, then that is what you give them…



This is acceptable because – in the actual ontology artifact on a given computing system – we can cut this part of the graph from the larger graph represented in the preceding design pattern

The rest is simply adopting configuration management strategies to ensure you SPCs remain aligned with CDPs

Exercise

- Suppose you’ve been given the following competency question to answer:

Are shipping containers transported by Wal-Mart’s truck fleet?

- Leveraging the design pattern emerging from your previous exercise, extract a simplified design pattern that addresses this question