Debugging Enterprise Ontologies

by Michael Uschold, Semantic Arts

Presented at CoDeS 2016

International Workshop on
Completing and Debugging the Semantic Web

Crete, Greece

Monday AM May 30, 2016

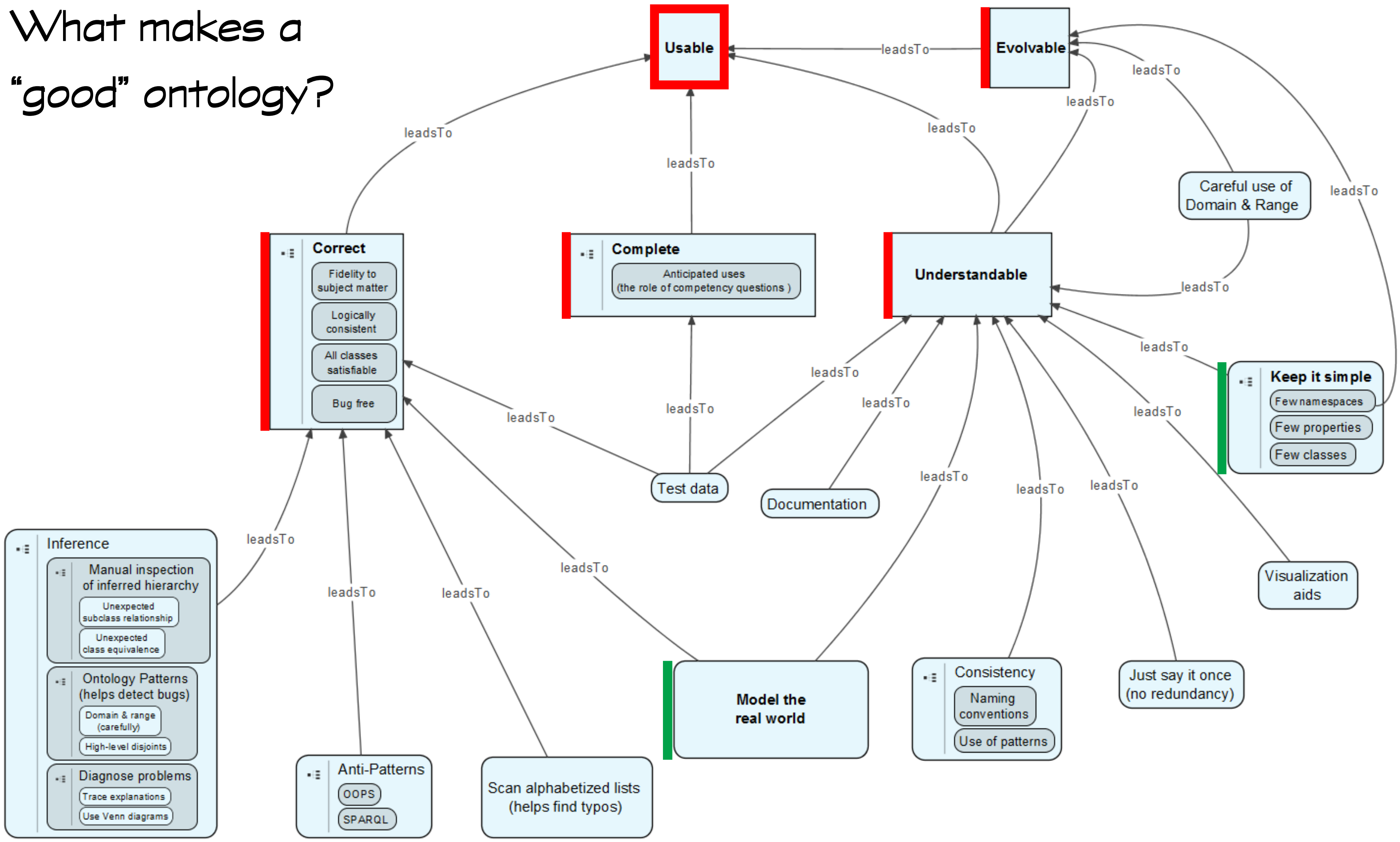
semantic arts

Overview

- The paper on a single slide
- Enterprise ontology
- Model the real world
- Keep it simple
- Finding and preventing bugs

The Paper on a Single Slide

What makes a "good" ontology?

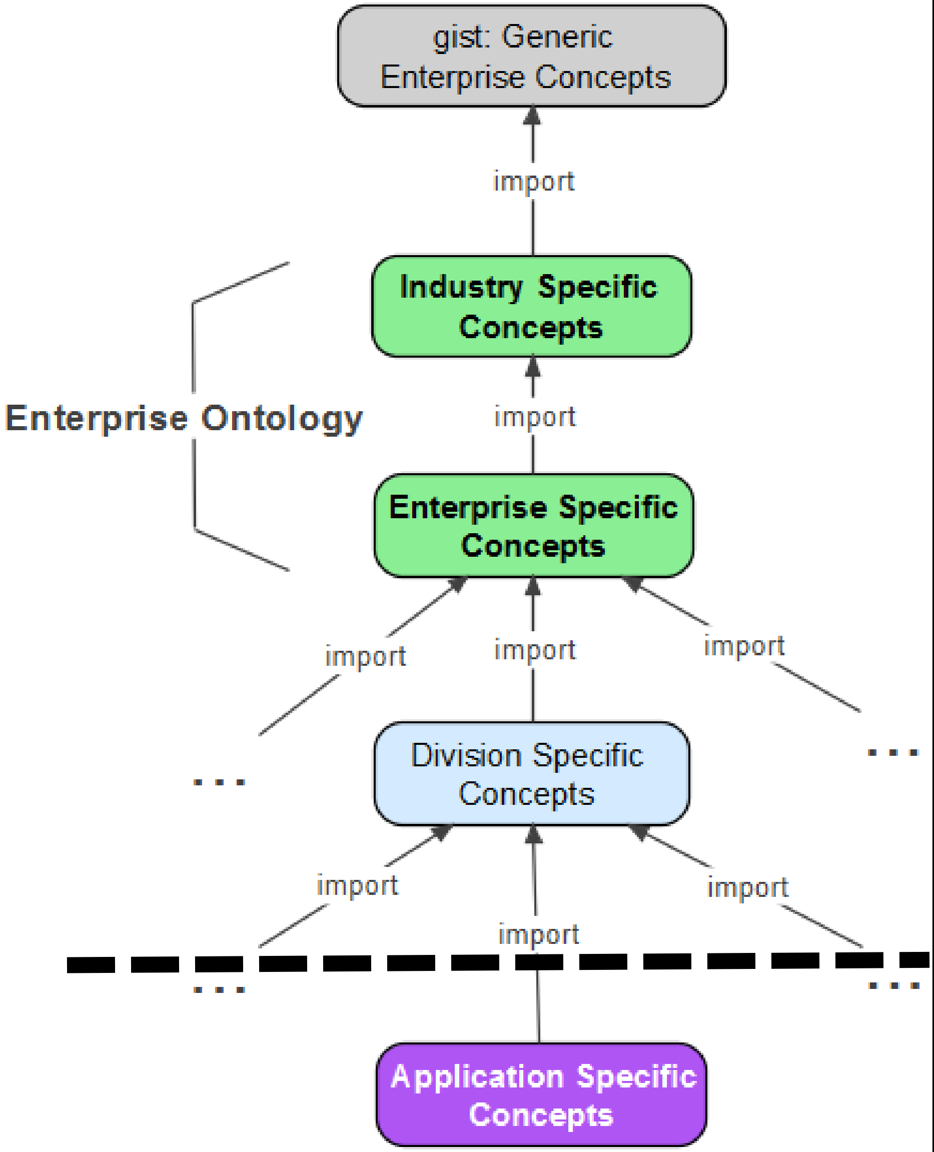


Enterprise Ontology

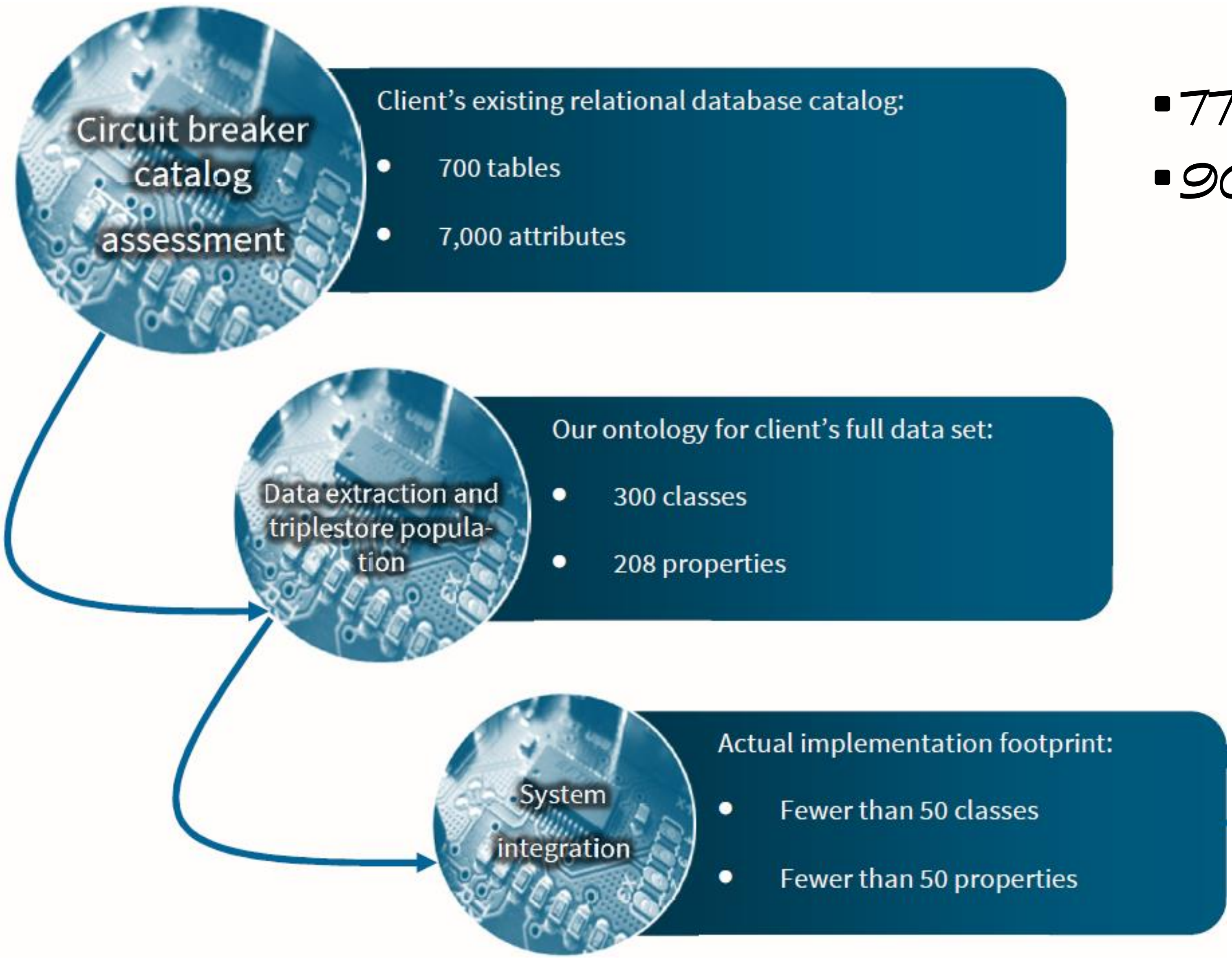
Scope: all/only concepts that are:

- Central to the enterprise
- Stable for a long time
- Substantially different from one another

Separate real world concepts from application concepts.


gist: Generic Enterprise Concepts
import
Industry Specific Concepts
Enterprise Ontology
import
Enterprise Specific Concepts
import
import
import
...
Division Specific Concepts
...
import
import
import
...
...
Application Specific Concepts

Model the Real World


Circuit breaker catalog assessment

Client's existing relational database catalog:

- 700 tables
- 7,000 attributes

Data extraction and triplestore population

Our ontology for client's full data set:

- 300 classes
- 208 properties

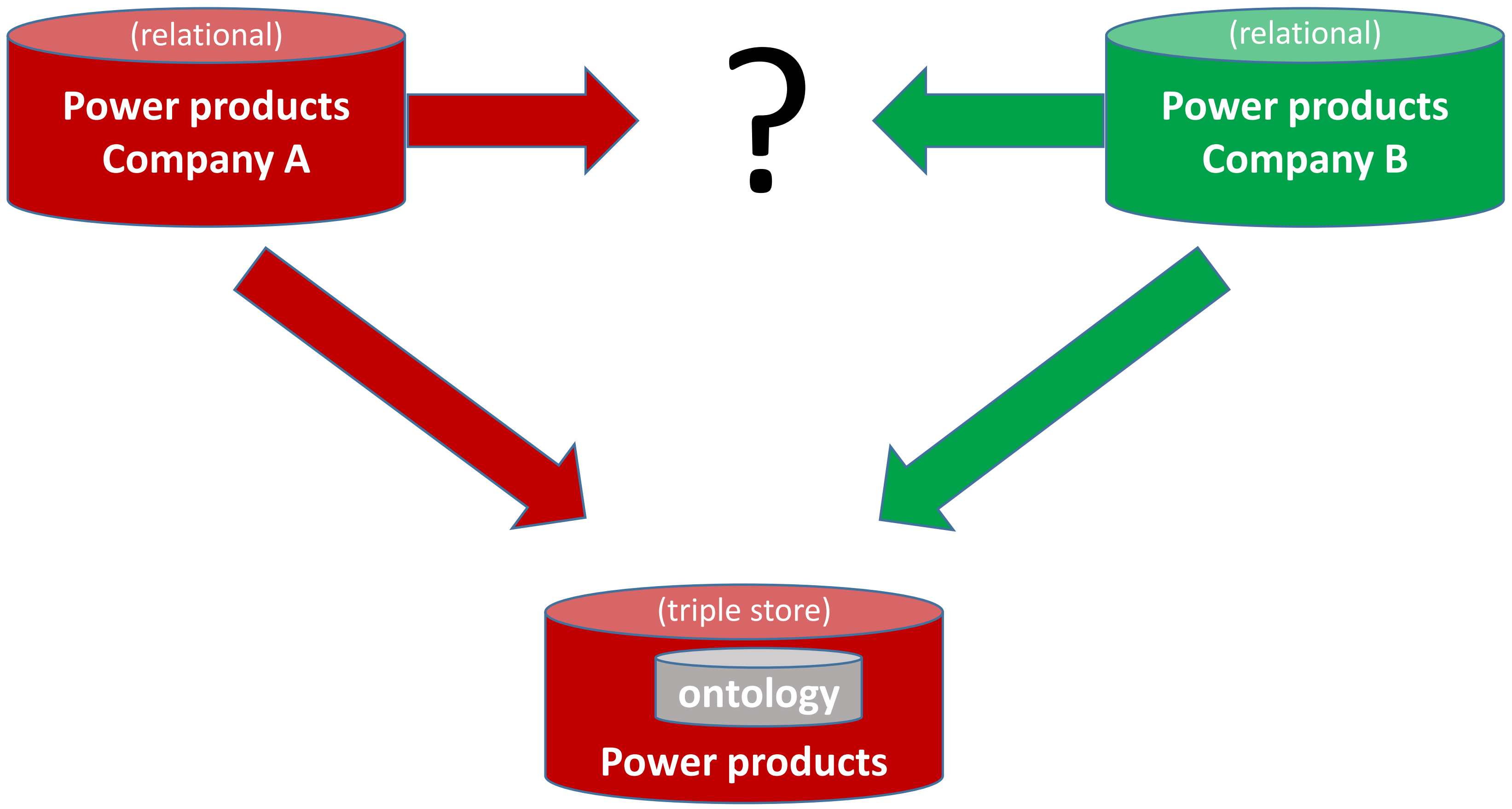
System integration

Actual implementation footprint:

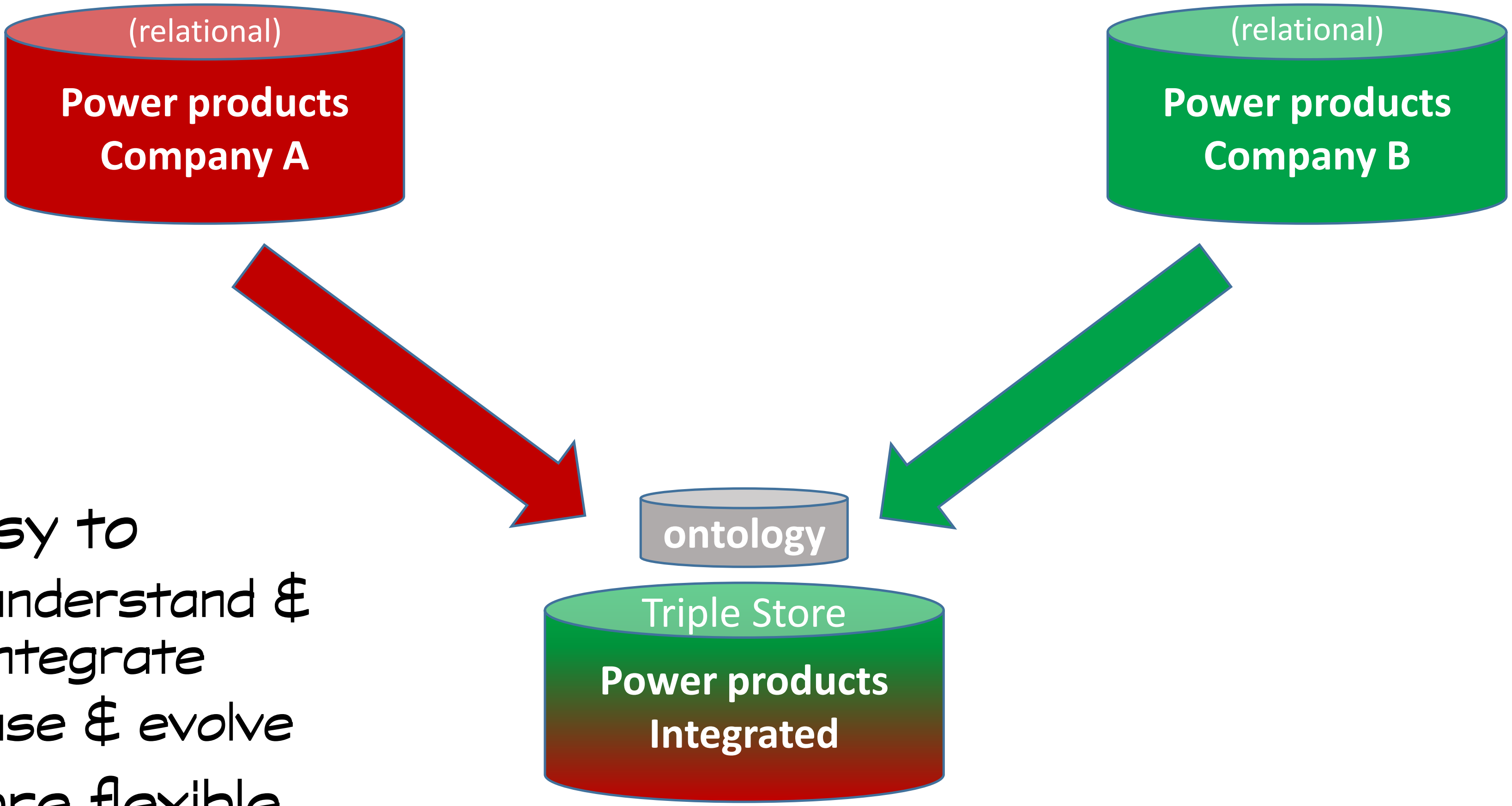
- Fewer than 50 classes
- Fewer than 50 properties

- 7700 vs. 708
- 90% is bloat

The Importance of Keeping it Simple



The Importance of Keeping it Simple

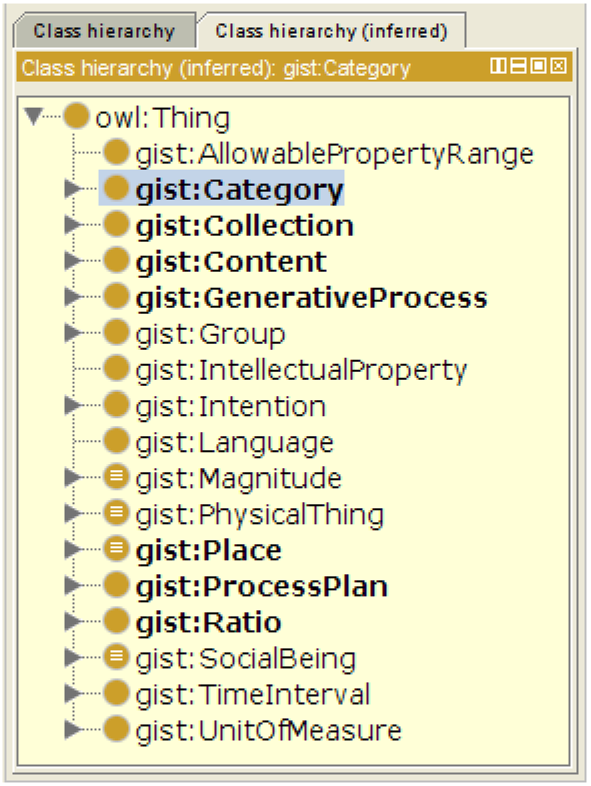


- Easy to
  - understand & integrate
  - use & evolve
- More flexible
- Prevents bugs

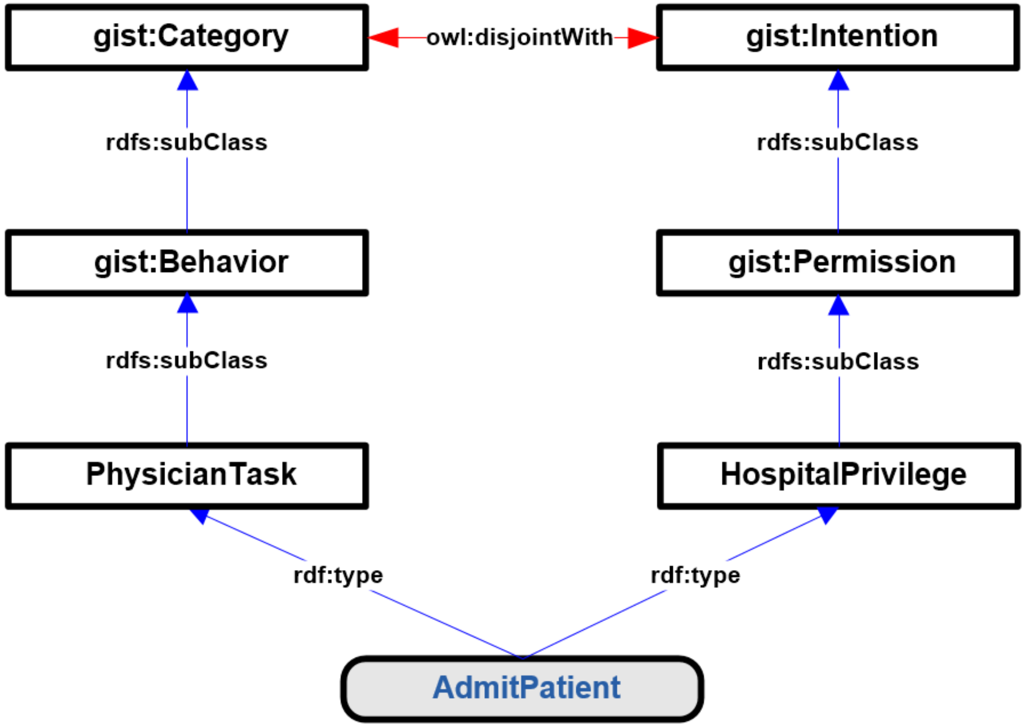
Ways to Catch & Prevent Bugs

- Keep it simple
- Use the inference engine to catch 20-30% of bugs. Help it along:
  - High level disjoints
  - Careful use of domain & range

High Level Disjoints


Class hierarchy
Class hierarchy (inferred)
Class hierarchy (inferred): gist:Category

- owl:Thing
  - gist:AllowablePropertyRange
  - gist:Category
  - gist:Collection
  - gist:Content
  - gist:GenerativeProcess
  - gist:Group
  - gist:IntellectualProperty
  - gist:Intention
  - gist:Language
  - gist:Magnitude
  - gist:PhysicalThing
  - gist:Place
  - gist:ProcessPlan
  - gist:Ratio
  - gist:SocialBeing
  - gist:TimeInterval
  - gist:UnitOfMeasure



- Have relatively few high level classes that are mostly disjoint.
- Combine with domain & range to catch many errors

Domain & Range: Use with Caution!

- Important way to catch bugs, BUT…
- It is common practice to over-constrain domain and range

Examples:

- W3C Media Ontology
  - hasPolicy & hasLanguage can only be used with MediaResource

- Financial Ontology
  - hasCount can only be used with Schedule.

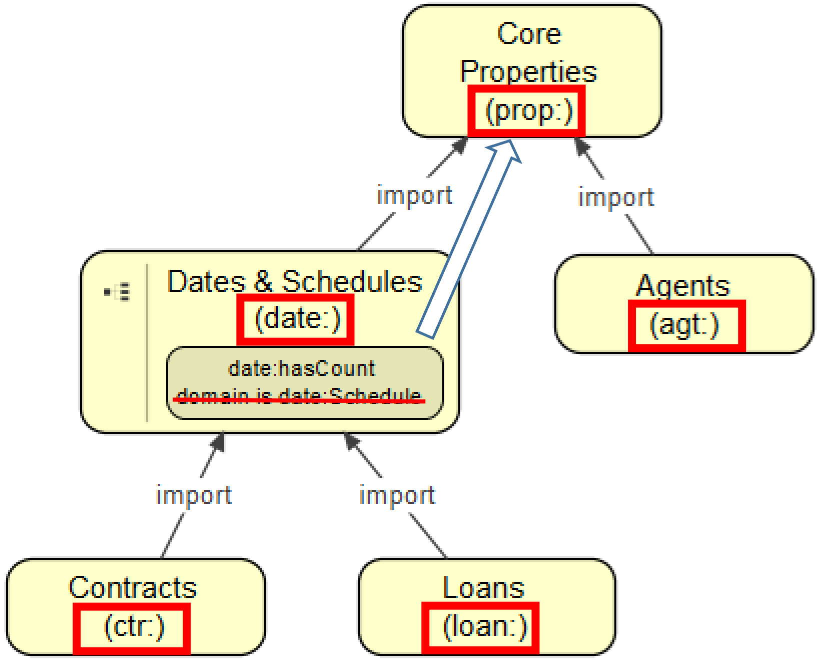
- Results in
  - difficulty in reuse
  - unnecessary proliferation of properties

Ways to Catch & Prevent Bugs

- Manually examine inferred hierarchy for oddities
- Anti-patterns:
  - identify
  - programmatically find examples
- Keep it simple, avoid proliferation of:
  - namespaces
  - properties
  - classes

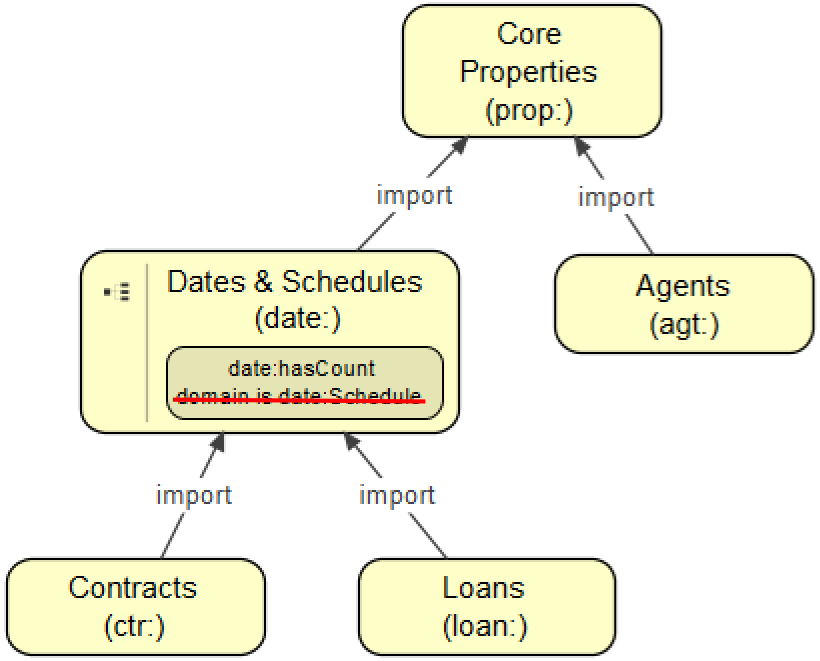
Namespaces

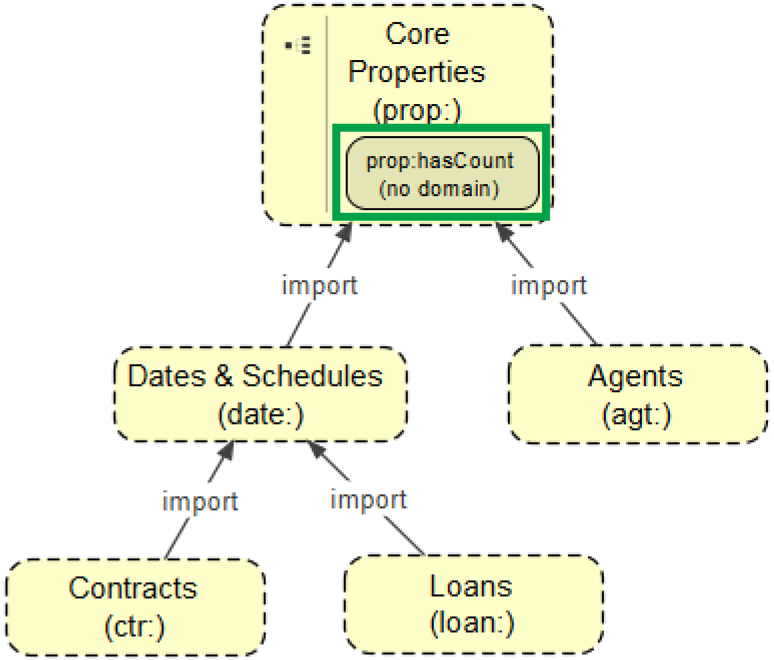
- It is common practice to have a different namespace for every ontology. I have seen 100+.



Namespaces

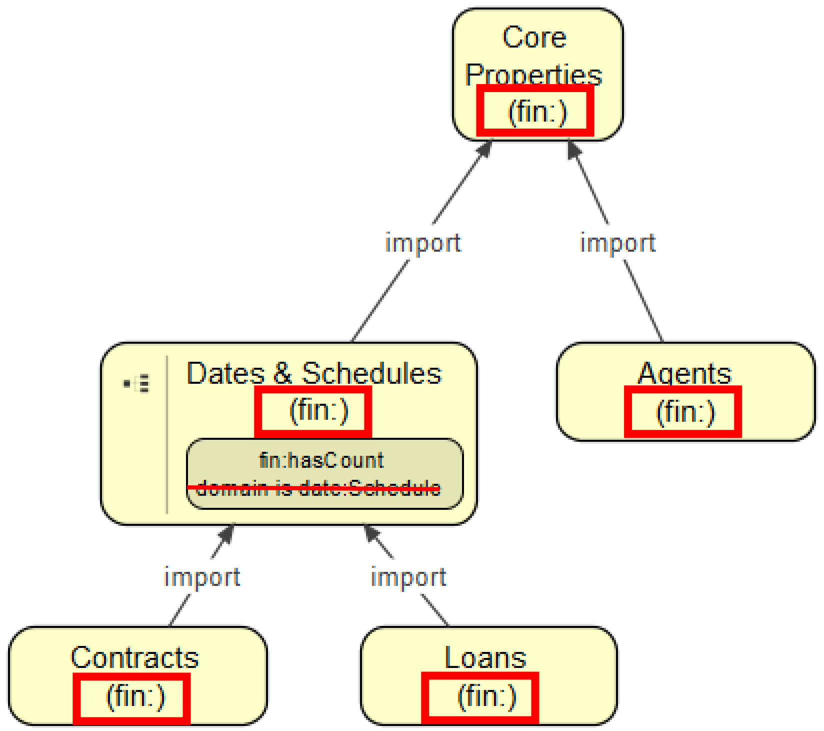
- It is common practice to have a different namespace for every ontology. I have seen 100+.
- Not needed for namespace collisions, all on same topic.
- Error prone for use, extending and refactoring.
- Have to check for use everywhere.

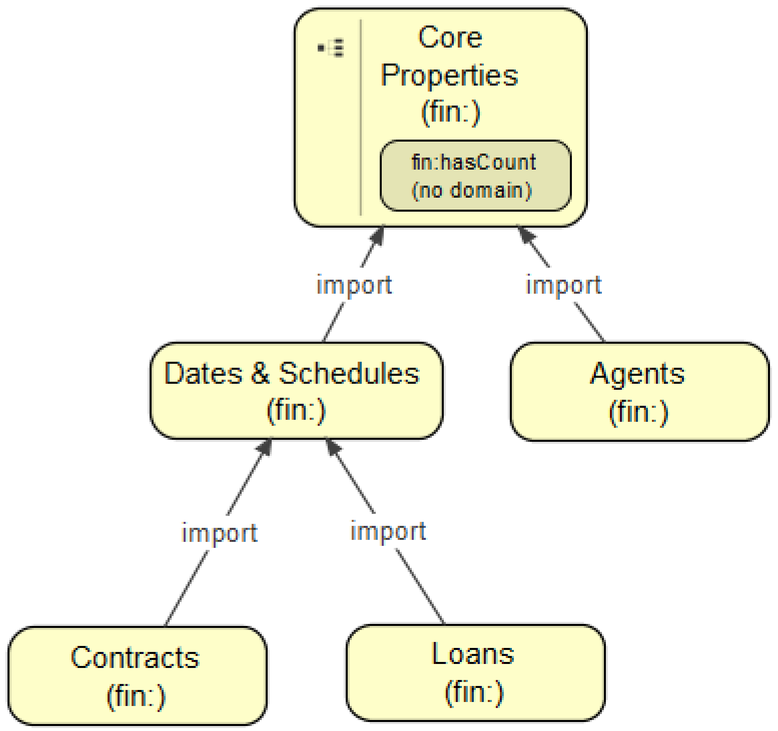




What We Do

- Use single namespace for ontologies on same topic
- No need to check for where used.
- Use different namespaces if under different governance





Namespaces: No Free Lunch

- There is a tradeoff
- This breaks the ‘follow your nose’ principle which makes it easy to find things on the web.
- This has not been a significant issue for us
- If this is important to you, consider adding more namespaces and be careful.
    - re-factoring is error-prone
    - don’t end up having the problem semantic technology solves: rigidity

Test Data

- Create a suite of test data
  - for ongoing unit testing
  - to illustrate how to use the ontology

- Correctness: ensuring things are the way you think they are.
- Completeness: can you represent the data you need to?
- Understandability: a suite of test examples to show users can be a really fast way for users to get started.

Competency Questions

- I'm a great fan of competency questions…
- … but we have not found them to play a major role.
- The enterprise ontology is relatively small and general.
- Perhaps more useful for more specific ontologies built out to support particular applications.

Conclusion

- Its all about usability
- The most usable ontologies are:
  - correct,
  - complete
  - understandable.
- Test data helps with all three characteristics
- Model the real world - not application-specific concepts
- Inference is a powerful tool, but cannot catch most bugs.
- Anti-patterns are important

Conclusion

- Keep it simple: if in doubt, leave it out
  - few classes: introduce only if different properties are important
  - few properties: introduce only if semantics genuinely different
  - few namespaces: new namespace if under different governance
- You can do things that help prevent bugs in the first place
  - keep it simple
  - make the ontology easy to maintain
  - use high-level disjoints & domain and range
  - avoid over-constraining domain and range
  - ease of use prevents bugs in downstream applications
- Competency questions have not featured largely for enterprise ontology development.