# Ontology Summit 2014: Big Data and Semantic Web Meet Applied Ontology Track C: Overcoming Ontology Engineering Bottlenecks – Synthesis II

Track Co-Champions:

Krzysztof Janowicz, Pascal Hitzler, Matthew West

# Track Mission

To identify bottlenecks that hinder the large-scale development and (re)usage of ontologies and identify ways to overcome them.

# Ontology Engineering Bottlenecks – Session II

**Oscar Corcho** (Universidad Politecnica de Madrid)

10 basic rules to overcome ontology engineering deadlocks in collaborative ontology engineering tasks

**Dhaval Thakker** (University of Leeds)

Modeling Cultural Variations in Interpersonal Communication for Augmenting User Generated Content

**Peter Haase** (Fluid Operations)

Developing Semantic Applications with the Information Workbench – Aspects of Ontology Engineering

# Some Key Problem Areas

- Ontologies are perceived as costly
- There is confusion over the level of expressiveness needed
- Who will develop the shared ontologies?
- How do we do quality control?
- What level of semantics is needed?
- What tools to use?
- How to reuse successfully?
- Why are ontologies in English?

# Ontologies are perceived as costly

- Ontologies may be hard to develop, but taken as a proportion of the overall project (at a business improvement level) they are a part of, they are generally a relatively small proportion of the total cost
- Timely and appropriate ontology development will reduce overall project costs, whereas no (implicit) ontology or late development of ontologies will lead to higher overall project costs
- You need to be able to make this case

# There is confusion over the level of expressiveness needed

- Different applications will require different levels of expressiveness:
  - Expressiveness is important for descriptive ontologies, where the queries are not known at design time
  - When you do know the queries to be answered, it is often possible to construct a more restrictive ontology that will answer those queries with improved performance
  - You may need multiple ontologies (or a master and subsets) to meet all needs in a domain

# Who will develop the shared ontologies?

- Many ontologies will be private – some ontologies will be public
  - Accounts – private
  - Product Catalogue – public
  - Public administration data, and standards - public
- The authoritative source should develop the shared ontology
  - Avoids replication
  - Sources need to be aware of obligation
- Examples
  - BIPM should develop UOM ontology
  - Governments should develop ontologies related to their laws
  - Standards bodies should develop ontologies common to multiple governments

# How do we do quality control?

- Test, test, test
- It is just the same as software development quality control and data quality control
- Inferencing tools can help with logical consistency, but there are many more errors that can be made beyond logical consistency

# What level of semantics is needed?

- The first priority is identity (same name – same thing) not semantics
- Level of semantics required varies from application to application and domain to domain.
- Examples
  - Engineering : quite a lot.
  - Life sciences : medium.
  - Publishing : very little as they focus on vocabularies

# What tools to use?

- It is sensible to start with lightweight tools like Excel
- You cannot manage large/complex ontologies with lightweight tools

# How to reuse successfully?

- Select for re-use
  - Reuse is generally not an objective in itself
  - Determine requirements before reviewing candidate ontologies to reuse
  - Reuse is successful when it reduces costs and increases quality
- Design for re-use
  - Reuse is unlikely to be achieved unless it is a design objective
  - Common errors are:
    - The ontology is over-constrained
    - Range and domain are set at too low a level of abstraction – the highest level in a particular domain, rather than across domains
    - Local constraints are implemented that are not generally valid

# Why are ontologies in English?

- Many ontologies intended for reuse are designed in English and it is assumed all users will use English – this is not valid
- It is pragmatic that IDs should be in the language of the developer, since this helps the development and debugging process
- IDs should be hidden from end users, who should be able to choose the language for the labels they see