Representing USDL in RDF

Service Module

Dr. Carlos Pedrinaci
(The Open University)

Dependency

- Dependencies are modelled between items and targets and include different types
- Should be modelled as a hierarchy of Properties
  - dependsOn
    - requires, includes, enhances, mirrors, canSubstitute, canConflictWith
- e3Service identifies other relationships like

Dependency

- e3Service identifies other relationships like coreEnhancing, etc.
- Good relations as well:
  - gr:addOn, gr:isSimilarTo, gr:isVariantOf, gr:isConsumableFor, gr:isAccessoryOrSparePartFor
- These relationships should be better explorer and materialised

Service

- Relevant external concepts are:
  - e3service:Service (equivalent)
  - msm:Service (not the same since this one is actually a description)
  - gr:ProductOrService
  - gr:ActualProductOrServiceInstance

Service

- Services include ServiceNature
- This could be modelled as classifications (see previous module on how to approach it)

Abstract Service

- See concept gr:ProductOrServiceModel
- Abstract Services are classified as well. (See previous times how it was approached)

Service Bundle

- See ServiceBundle in e3Service
- Can be modelled in GoodRelations using the gr:isAccessoryOrSparePartFor
- Bundles have bundling constraints which would best be modelled using SPARQL (very simple ones) or RIF for more advanced ones (see WSMO-Lite approach for axioms)

Composite Service

- I'm not clear what the difference between ServiceBundle and CompositeService is
- Grouping constraint would again be best modelled using SPARQL (very simple ones) or RIF for more advanced ones

Service Variants

- Waiting for the discussion
- Relevant notions in GoodRelations
  - ProductOrServiceModel vs ActualProductOrServiceInstance
  - gr:successorOf

Parts

- ServiceBundles and CompositeService have parts
  - http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/
  - http://ontologydesignpatterns.org/wiki/Submissions:PartOf
- Typically though, it is beneficial to distinguish between directPart and indirectPart (only 1st does)

Parts

- Parts could be optional
- The optionality of parts could be done with subproperties of partOf (e.g., optionalPartOf)

Network Provisioned Entity

- Probably a new concept
- Reuse terms like rdf:label, dc:version, etc