A Comparison of XML and OWL based Approaches to Representing Cognitive Radio Functions

YANJI CHEN

MAR 17, 2016

Problem Statement

- Dynamic spectrum access is a way to reduce underutilization of spectrum
- Radios may request permission to transmit, and be allowed or disallowed to do so, depending on the applicable policies
- Q: What languages are good for describing the requests, policies and radio capabilities?

Possible Solutions

Overview of the W3C Standards currently adopted in the XML and SW Worlds [1]

	Schema	Data	Query
XML World	XML Schema	XML	XQuery/XPath
Semantic World	RDF Schema /OWL	RDF	SPARQL

[1] Nikos Bikakis, **Chrisa Tsinaraki**, **Nektarios Gioldasis**, **Ioannis Stavrakantonakis** and **Stavros Christodoulakis, "The XML and Semantic Web Worlds: Technologies, Interoperability And Integration. A Survey of the State of the Art" in "Semantic Hyper/Multi-media Adaptation: Schemes and Applications", Springer 2013.**

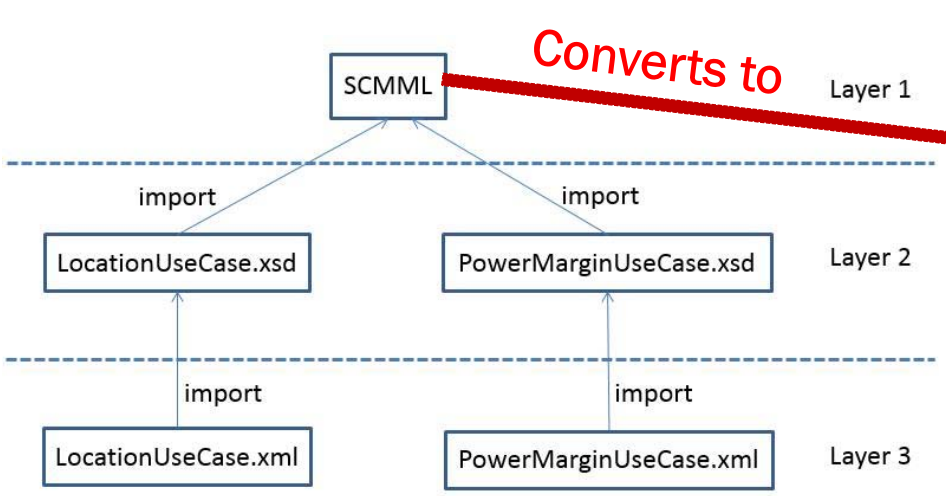
How to Make a Reasonably Fair Comparison?

- Use same data sets
  - Reuse two use cases published earlier
  - First represent the data in SCMML – the schema originally proposed for representing Spectrum Consumption Models (SCMs), proposed by John Stine and Sam Schmitz [1]
  - Represent the SCMML data in both XML and RDF
- Use the same queries
  - Use the same queries (expressed in English)
  - Translate the queries into XQuery and SPARQL

[1] J. Stine and S. Schmitz, "Model based spectrum management," The MITRE Corporation, Tech. Rep., 2013.

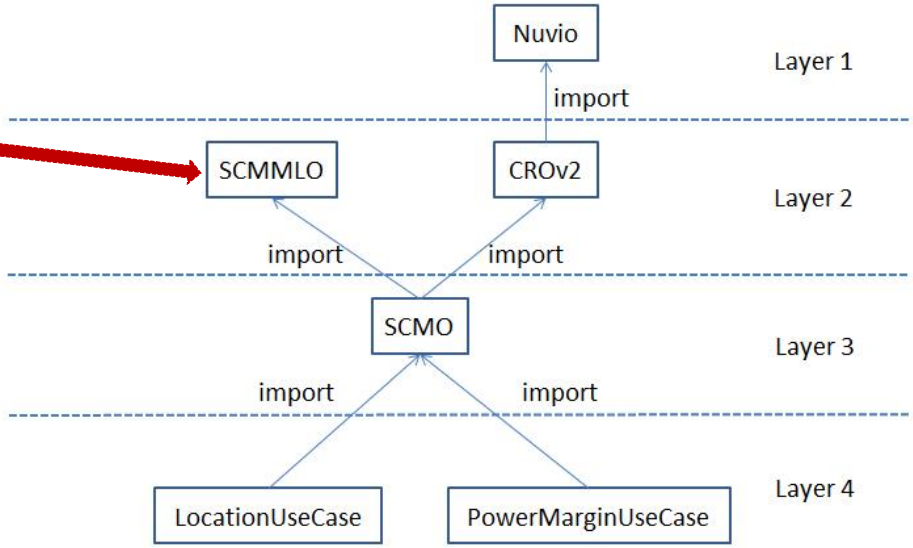
Modeling Structure

XML Approach



Converts to

OWL Approach



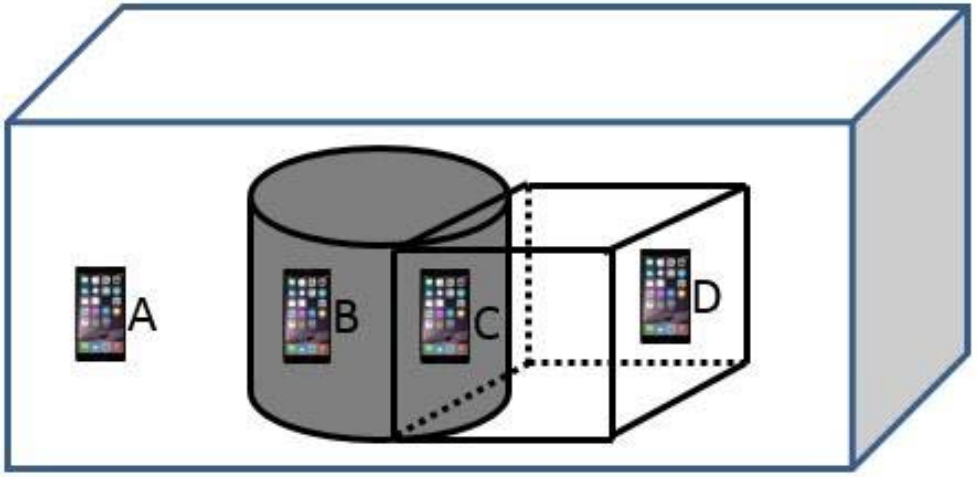
Use Case Review

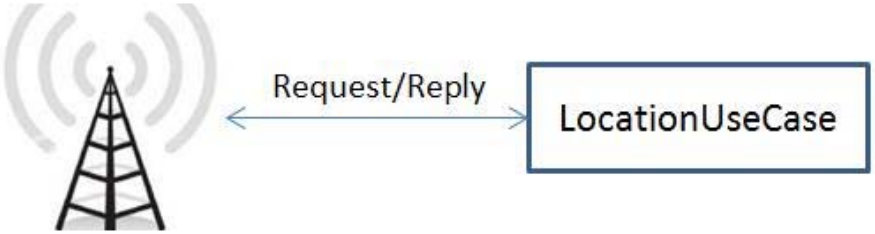
- **Location use case:** Determine whether transmitters of different stakeholders can transmit from specific locations
- **Power margin use case:** Determine whether signals from specific transmitters might be harmful to receivers that are not target receivers, before actual transmission occurs

Location Use Case

Locations:

- Large cuboid
- Cuboid
- Cylinder





Scenario Description

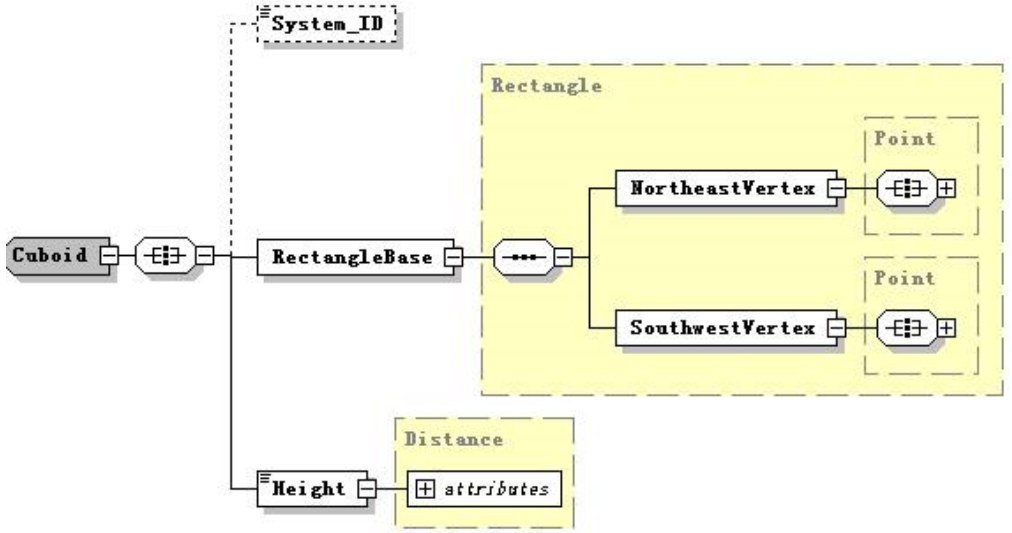
Transmitter Info and Policies for Particular Locations

	Stakeholder	Requested Spectrum
TransmitterA	**AT&T**	**600-606MHz**
TransmitterB	**T-Mobile**	**300-306MHz**
TransmitterC	**Verizon**	**600-606MHz**
TransmitterD	**Sprint**	**312-318MHz**

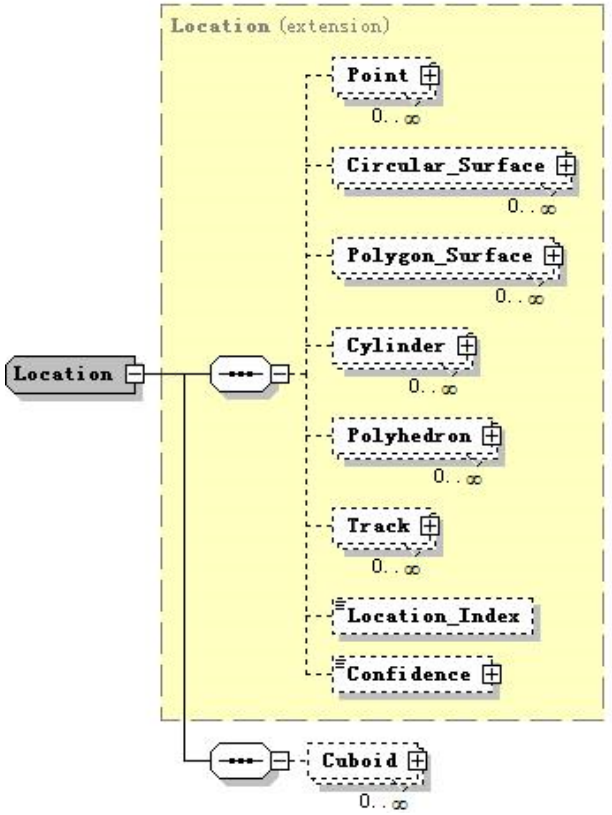
Stakeholder	Location	Spectrum	Allow to Use
AT&T	**Cuboid**	**560-610MHz**	**N**
AT&T	**Cylinder**	**560-610MHz**	**N**
T-Mobile	**Cylinder**	**280-320MHz**	**N**
Verizon	**Cuboid**	**590-620MHz**	**Y**
Sprint	**Cylinder**	**300-320MHz**	**N**
⋮	⋮	⋮	⋮

Modeling with SCMML

- Define new types



- Extend the types from SCMML

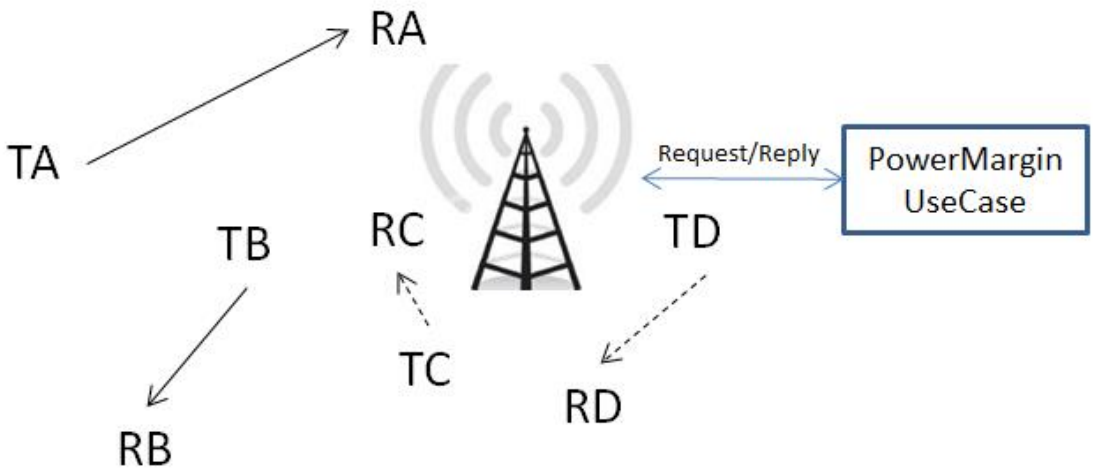


Representing and Processing Queries in XQuery

- Represent policies in XSD plus XQuery
- Annotate the data in XML
- Execute a query: Querying whether a mobile station can use the spectrum in the place where it is currently located
- XQuery FLOWR expressions

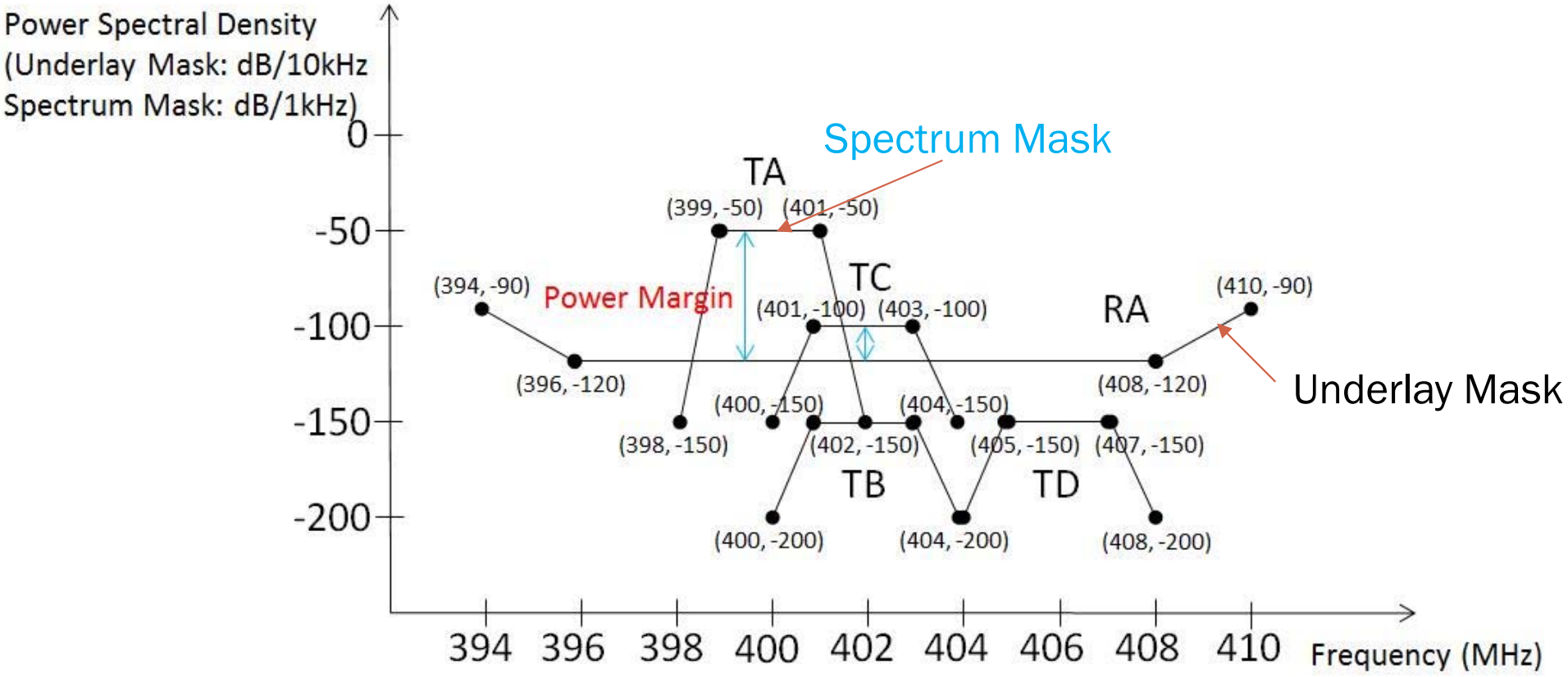
```
CHECK-POLICIES(P, T)
1  for each policy p ∈ P
2      for each transmitter t ∈ T
3          if p.allowedStakeholder == t.stakeholder
   and t.location ∈ p.region and
   t.allocatedSpectrum ∈ p.spectrumRange
4              PRINT-ALLOWED(t, p.region)
```

Power Margin Use Case



Scenario Description

Masks


Power Spectral Density
(Underlay Mask: dB/10kHz
Spectrum Mask: dB/1kHz)
0
-50
-100
-150
-200
Spectrum Mask
TA
(399, -50)
(401, -50)
TC
(394, -90)
Power Margin
(401, -100)
(403, -100)
RA
(410, -90)
(396, -120)
(408, -120)
Underlay Mask
(400, -150)
(404, -150)
(398, -150)
(402, -150)
(405, -150)
(407, -150)
TB
TD
(400, -200)
(404, -200)
(408, -200)
394
396
398
400
402
404
406
408
410
Frequency (MHz)

Representing Queries in XQuery

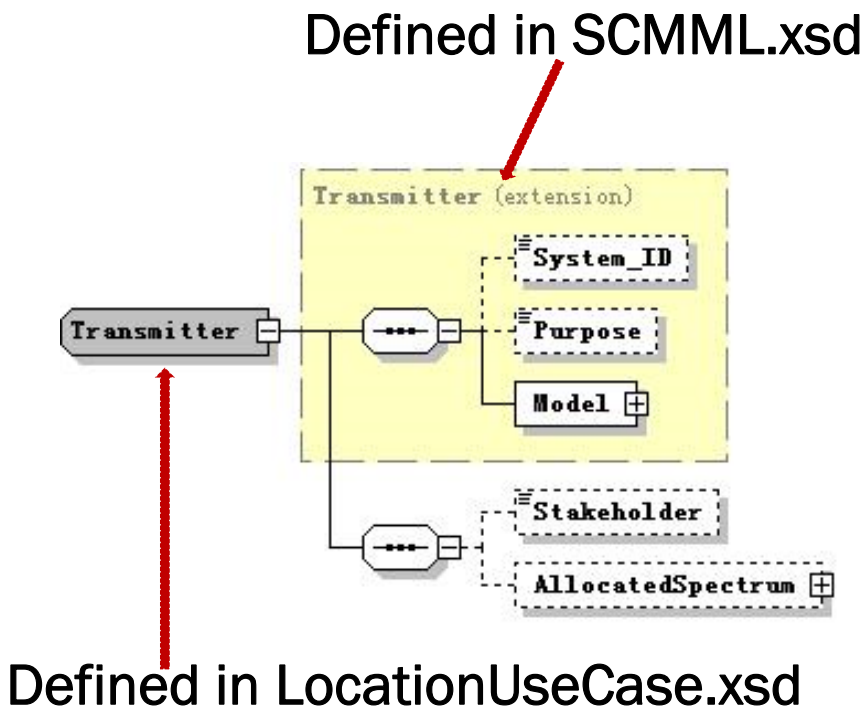
- Execute a query: Querying whether sufficient power margins exist for the receiver-transmitter pairs
- Use same data sets
  - Find all the transmitter-receiver pairs
  - Convert undelay mask and interfering signal's spectrum mask to the same resolution bandwidth
  - Detect whether power margin for each pair exists

Using OWL Approach vs. XML Approach

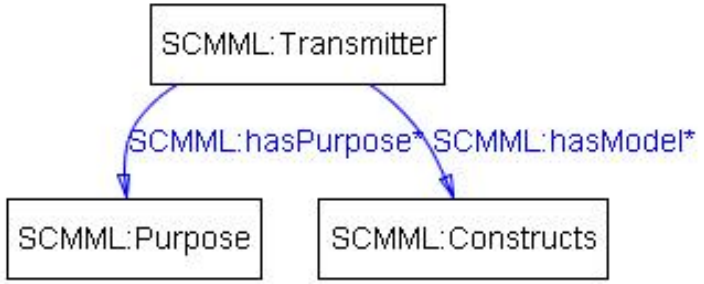
Features		XML	OWL
Modeling	Amount of Hard-coding	**High**	**Low**
	Expressiveness	**Tree**	**Graph**
	Multiple Inheritance	**No**	**Yes**
	Polymorphism	**No**	**Yes**
	Automatic Inference	**No**	**Yes**
Querying	Intermediate Variables	**Succinct**	**Redundant**
	Function Library	**Strong**	**Weak**

Modeling: Amount of Hard-coding Required

XML Approach



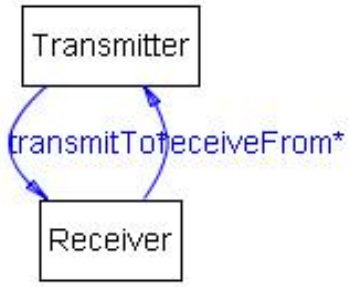
OWL Approach



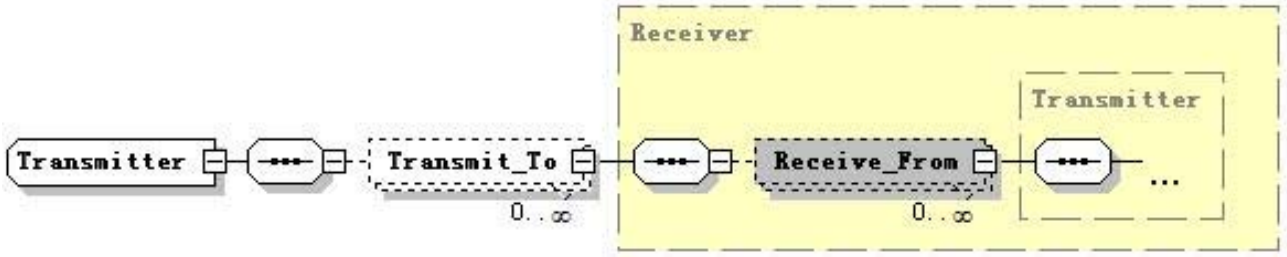
Modeling: Expressiveness (1/2)

- XML: Elements organized in a tree structure
- OWL: Classes organized via properties, forming a graph
- Example: Modeling “transmitters transmit signals to receivers, and the receivers receive signals from the transmitters”

OWL Approach



XML Approach

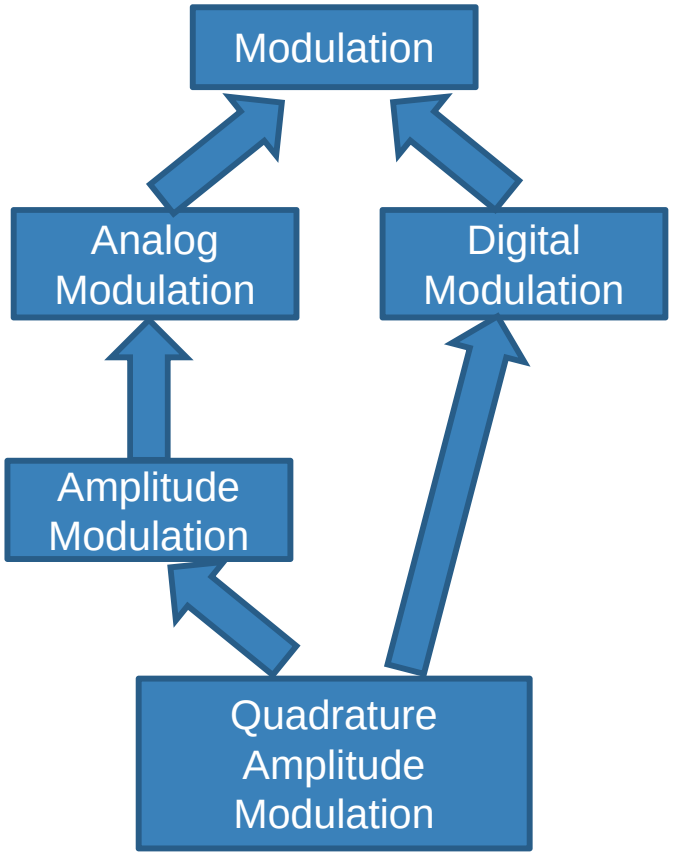


Modeling: Support of "Inheritance" (1/2)

- Inheritance in OOP: when a class extends another class (super class) and uses the super class implementation to inherit both its structure and **behavior**
- XML doesn't inherit behavior, it only inherits syntax
- XML approach: <extension> tag
- OWL approach: rdf:subClassOf

Modeling: Support of "Inheritance" (2/2)

- A class in OWL can be defined as a subclass of multiple ancestors (multiple inheritance)
- Example:



Modeling: Support of “Polymorphism” (1/2)

- In programming languages and type theory, **polymorphism** is the provision of a single interface to entities of different types.
- A polymorphic type is one whose operations can also be applied to values of some other type, or types.
- Q: Is an element of Cuboid complex type a Polyhedron type?
  - It should be yes, but XSD doesn’t know…

Modeling: Support of “Polymorphism” (2/2)

- Modeling: Policy complex type

```
<xs:complexType name="Policy">
    <xs:all>
        <xs:element name="Stakeholder" type="xs:string"/>
        <xs:element name="AppliedRegion" type="Polyhedron"/>
        <xs:element name="SpectrumRange" type="Frequency_Band"/>
        <xs:element name="AllowtoUse" type="xs:boolean"/>
    </xs:all>
</xs:complexType>
<xs:complexType name="Cuboid">
    <xs:complexContent>
        <xs:extension base="Polyhedron"/>
    </xs:complexContent>
</xs:complexType>
```

- Data: XML data of Policy complex type

```
<Policy>
    <Stakeholder>ATT</Stakeholder>
    <AppliedRegion>
        <Cuboid>
            <System_ID>TestCuboid</System_ID>
            <RectangleBase>
                <NortheastVertex>
                    <Longitude>55</Longitude>
                    <Latitude>44</Latitude>
                    <Altitude>0</Altitude>
                </NortheastVertex>
                <SouthwestVertex>
                    <Longitude>49</Longitude>
                    <Latitude>33</Latitude>
                    <Altitude>0</Altitude>
                </SouthwestVertex>
            </RectangleBase>
            <Height>100</Height>
        </Cuboid>
    </AppliedRegion>
    <SpectrumRange>
        <Start_Frequency units="MHz">600</Start_Frequency>
        <End_Frequency units="MHz">606</End_Frequency>
    </SpectrumRange>
    <AllowtoUse>false</AllowtoUse>
</Policy>
```

Modeling: Automatic Inference

- XML approach
  - All relations among elements have to be hard-coded
  - All the inference steps need to be hard-coded into the XQuery expressions
- OWL approach
  - Properties with various characteristics model relationships among components
  - OWL inference engine generates more implicit facts from such data

Querying: Expression on Intermediate Variables

- XQuery makes it easier to store intermediate variables
- Example:

```
$altitudeForCircleCenter :=
   $cylinder/scmml:Base/scmml:Center/scmml:Altitude
```

XQuery

```
<Individual variable="AShape" rdf:type="scmml:Cylinder
 <scmml:hasBase variable="BaseForACylinder">
   <scmml:hasCenter variable="CircleCenter">
    <scmml:hasAltitude variable=
    "AltitudeForCircleCenter">
     <scmml:hasValue variable=
     "AltitudeValueForCircleCenter"/>
    </scmml:hasAltitude>
   </scmml:hasCenter>
 </scmml:hasBase>
</Individual>
```

BaseVISor Query

Querying: Support of Function Library

- XQuery has a better support for function library
- Example:
  - fn: index-of($seqParam, $srchParam)
  - Get the index of location associated with spectrum mask

```
let $indexOfLocation := index-of($transmitter/Model/Location,
$transmitter/Model/Location[System_ID = $receiver/Model/Location/System_ID])
```

Conclusions

- It is possible to use pure XML and XQuery as a non-semantic approach to derive desirable results.
- OWL approach is better than XML approach in the aspect of succinct modeling.
- XQuery is better at querying data due to its succinct expressions and strong support of function libraries.
- XQuery does not provide any semantic inference rules, and thus all inferences must be explicitly hard-coded into the queries.

Future Work

- Compare two approaches in a quantitative way.
- Use metrics – precision/recall, bandwidth usage, the size of encoding of instance data, query, encoding size, inference and query processing times, query complexity.
- Dynamic behaviors – supporting new devices and functionalities.

THANKS!

Any questions?

You can find me at

chen.yanj@husky.neu.edu