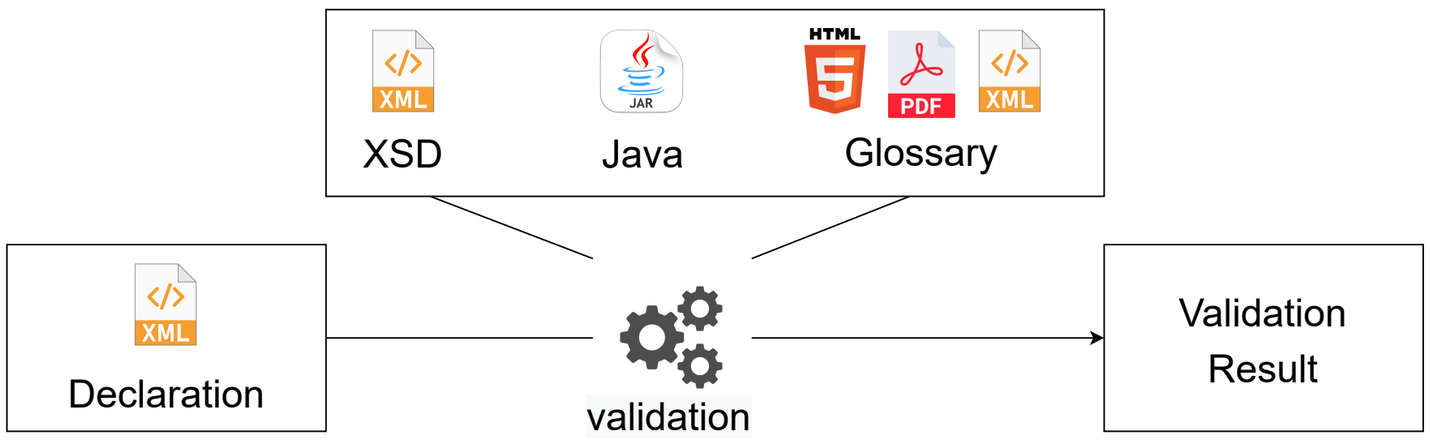
Using Knowledge Graph Technologies to Contextualize and Validate Declarations in the Social Security Domain

Davan Chiem Dao, Montefiore Institute, University of Liège
Supervised by Christophe Debruyne, Montefiore Institute, University of Liège

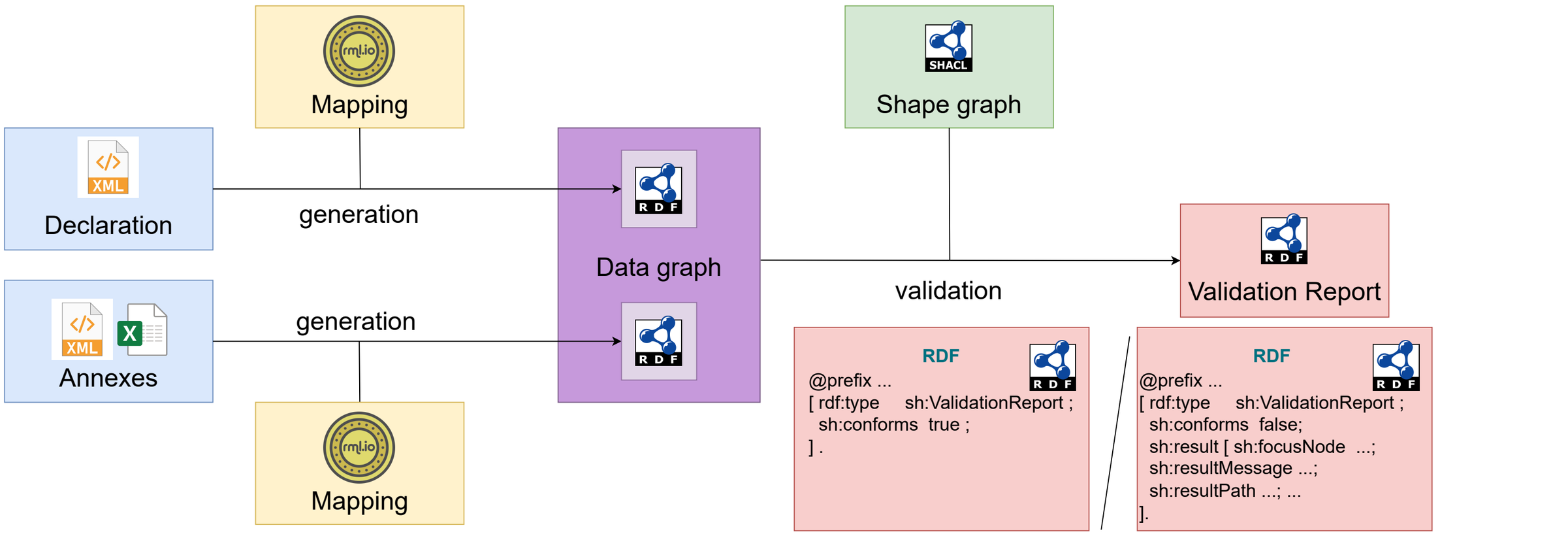
Current situation



Problems

- XSD: Not expressive enough for all rules
- Java: Expressive enough for all rules but not in an interoperable format
- Glossary: Specifies rules in natural language which cannot be processed by a computer agent

Graph Technologies



DmfA.xml

```
<DmfAOriginal> ...
 <Form> ...
  <Identification>DMFA</Identification> ...
  <EmployerDeclaration> ...
   <Quarter>20223</Quarter> ...
   <NaturalPerson> ...
     <NaturalPersonSequenceNbr>1</NaturalPersonSequenceNbr> ...
   </NaturalPerson>
   <NaturalPerson> ...
     <NaturalPersonSequenceNbr>2</NaturalPersonSequenceNbr> ...
   </NaturalPerson>
   <NaturalPerson> ...
     <NaturalPersonSequenceNbr>3</NaturalPersonSequenceNbr> ...
   </NaturalPerson>
...
```

RML

```
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
...
<#Form-Mapping> a rr:TriplesMap ;
  rml:logicalSource [ rml:source "DmfA.xml" ; ... ] ;
  rr:subjectMap [ ... ] ;
  rr:predicateObjectMap [
    rr:predicate ont:Identification;
    rr:objectMap [
      rml:reference "Identification";
      rr:datatype xs:string ; ] ; ] ;
  rr:predicateObjectMap [
    rr:predicate ont:R_90059_90007 ;
    rr:objectMap [
      rr:parentTriplesMap <#EmployerDeclaration-Mapping> ;
      rr:joinCondition [ ... ] ; ] ; ] ;
...
```

RDF

```
@prefix rdfs: ...

_:DmfAOriginal0 rdfs:type ont:DmfAOriginal .
_:Form0 rdfs:type ont:Form .
_:Form0 ont:Identification "DMFA".
_:Form0 ont:R_90059_90007 _:EmployerDeclaration0.
_:EmployerDeclaration0 rdfs:type
      ont:EmployerDeclaration.
_:EmployerDeclaration0 ont:Quarter
      "20222"^^xsd:integer.
_:EmployerDeclaration0 ont:R_90007_90017
      _:NaturalPerson0.
ont:NaturalPerson0 rdfs:type ont:NaturalPerson.
ont:NaturalPerson0 ont:NaturalPersonSequenceNbr
      "1"^^xsd:integer.
```

SHACL

```
kg:FormShape a sh:NodeShape ;
   sh:targetClass ont:Form ;
   sh:property [
      sh:path ont:Identification;
      sh:datatype xs:string;
      sh:minCount 1 ;
      sh:maxCount 1 ;
      sh:maxLength 7 ;
      sh:sparql _:codeIsValid;
      sh:sparql _:codeInQuarterRange;
   ]
.
```

Constraint value to belong to a set defined by an annex

```
_:codeIsValid a sh:SPARQLConstraint;
   sh:prefixes <> ;
   sh:select """
      SELECT $this ?value WHERE {
        { SELECT $this ?value (SUM (?match) as ?nbrMatch) WHERE {
          $this $PATH ?value.
          ?idtype a an11:IdentificationType ; rdfs:label ?allowedvalue.
          BIND (IF (?value = ?allowedvalue, 1, 0 ) AS ?match)
        } GROUP BY ?value $this}
        FILTER(?nbrMatch = 0)
      }""" ;
.
```

Constraint value to be temporally valid as defined by an annex

```
_:codeInQuarterRange a sh:SPARQLConstraint;
   sh:prefixes <> ;
   sh:select """
   SELECT $this ?value WHERE {
      {SELECT $this ?value (SUM (?match) as ?nbrMatch) WHERE {
         $this $PATH ?value.
         ?wc a an11:IdentificationType; rdfs:label ?value;
            an11:validFromQuarter ?sQuarter;
            an11:validToQuarter ?eQuarter;
         .
         $this ont:R_90059_90007/ont:Quarter ?quarter.
         BIND( IF(?sQuarter < ?quarter && ?quarter < ?eQuarter, 1, 0) AS ?match)
      } GROUP BY ?value $this}
      FILTER(?nbrMatch = 0)
   }""" ;
.
```

Constraint value to be unique

```
_:valueIsUnique a sh:SPARQLConstraint;
   sh:prefixes <> ;
   sh:select """
      SELECT ?this
      WHERE {
        {
          SELECT ?this (COUNT(?s) as ?sOcc)
          WHERE {
            $this ont:R_90007_90017 ?np
            ?np ont:NaturalPersonSequenceNbr ?s.
          }
          GROUP BY ?s $this
        }
        FILTER(?seqNbrOcc > 1)
      }""" ;
.
```

Advantages

- Integration of data from various sources in multiple formats into a single data model (RDF)
- SHACL is a W3C standard hence the rules are defined in an interoperable format
- SHACL is more expressive than XSD and more complex rules can be acheived.
  - See examples on the left
  - Checksums
  - Simple arithmetic
  - And many more ...