

Shapes applications and tools

Jose Emilio Labra Gayo
WESO Research group
University of Oviedo, Spain



Contents

Shapes applications and use cases

- Data portals

- Wikidata and wikibase

- Other use cases

Tools: challenges and perspectives

- Validating with shapes

 - Validation usability

 - Continuous integration

- Other applications of shapes

 - UIs

 - Generating code

 - Inference and rules

 - Transforming data

- Obtaining shapes

- Shapes ecosystems

Data portals

In 2013, at WESO, we were hired to develop some data portals

Examples: WebIndex (Web Foundation)

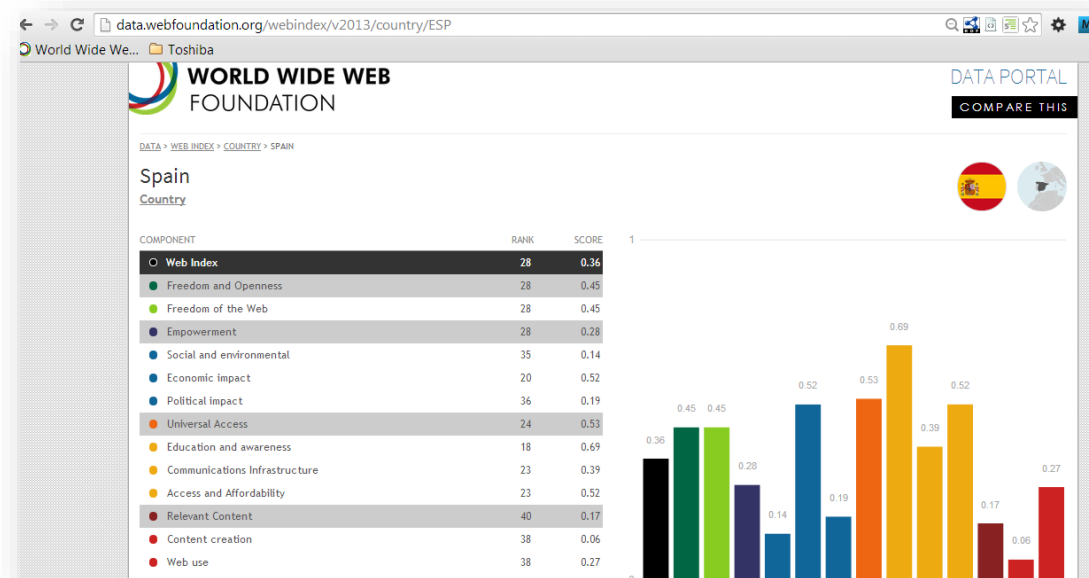
One of the first applications of ShEx

Measure WWW's contribution to development and human rights by country

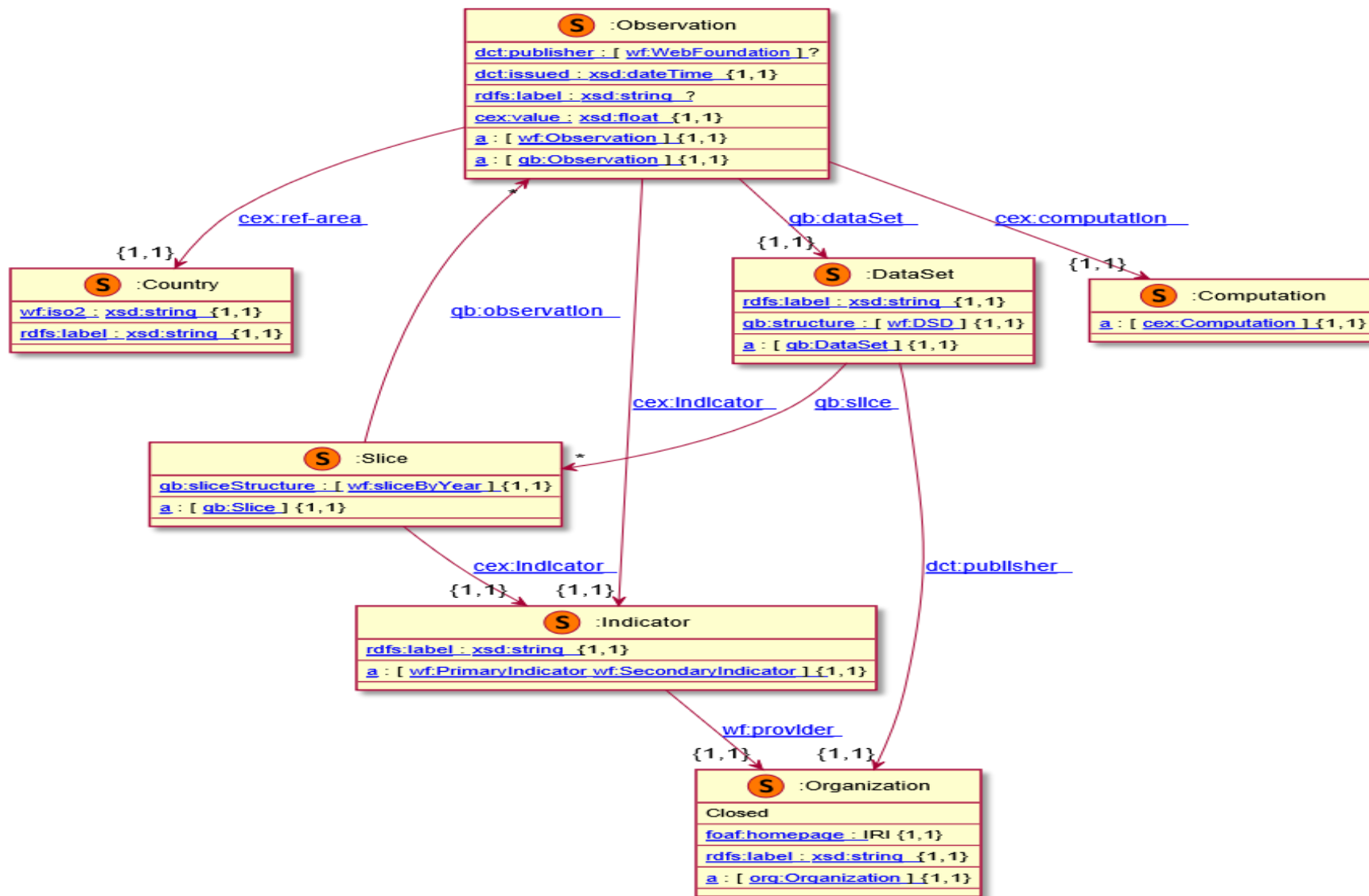
Developed by the Web Foundation

Content of data portal = statistical observations

We employed RDF Data Cube vocabulary (qb:Observation)



Simplified WebIndex data model



Lessons learnt from ShEx usage at WebIndex

1. Documentation of linked data portal

Human-readable, machine processable

<http://weso.github.io/wiDoc>

2. Team communication

Communicate the developers which shapes they had to generate

3. Validation

For example: check if a value of type `qb:Observation` had shape `<Observation>`

4. Reuse

Another data portal was later developed for <http://landportal.org> base on observations

Easy to reuse and adapt the data model

Same types (`qb:Observation`) but different structure

<http://weso.github.io/landportalDoc/data/>

Wikidata and wikibase



In May, 2019, Wikidata announced ShEx adoption

New namespace for schemas

Example:

<https://www.wikidata.org/wiki/EntitySchema:E2>

Wikibase also contains entity schemas

Online demo: [wikishape](#)

EntitySchema Discussion Read View history Search devwiki

Disease (E2)

language code	label	description	aliases	edit
en	Disease	Shape Expression for diseases coming from Disease Ontology in Wikidata (Source: https://github.com/andrawaag/Genewiki-ShEx/blob/master/diseases/Wikidata-Disease-Ontology.shex)	Sickness Illness	edit

Solid project

SOLID (SOcial Linked Data): Promoted by Tim Berners-Lee

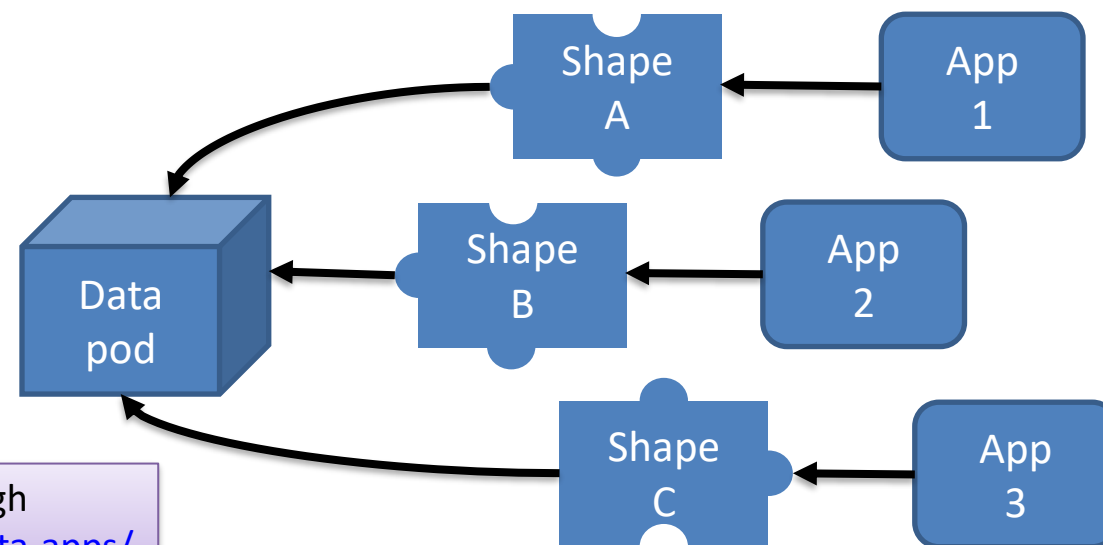
Goal: Re-decentralize the Web

- Separate data from apps

- Give users more control about their data

- Internally using linked data & RDF

Shapes needed for interoperability



"...I just can't stop thinking about shapes.", Ruben Verborgh

<https://ruben.verborgh.org/blog/2019/06/17/shaping-linked-data-apps/>

Other use cases

HL7 FHIR.

Example: <https://www.hl7.org/fhir/observation.html>

ELI validator

SHACL shapes obtained from Excel sheets:

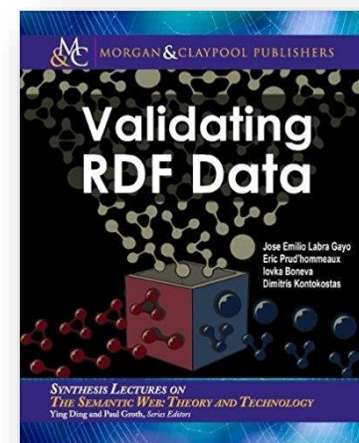
<https://webgate.ec.europa.eu/eli-validator/home>

SHACL adoption supported by Top Quadrant

See: <https://www.topquadrant.com/technology/shacl/>

More info:

Chapter 6 of Validating RDF data: <http://book.validatingrdf.com/bookHtml012.html>



Tools: challenges and perspectives

Validating with shapes

Obtaining shapes

Other applications of shapes

Shapes ecosystems

Validating with shapes

Libraries and command line validators

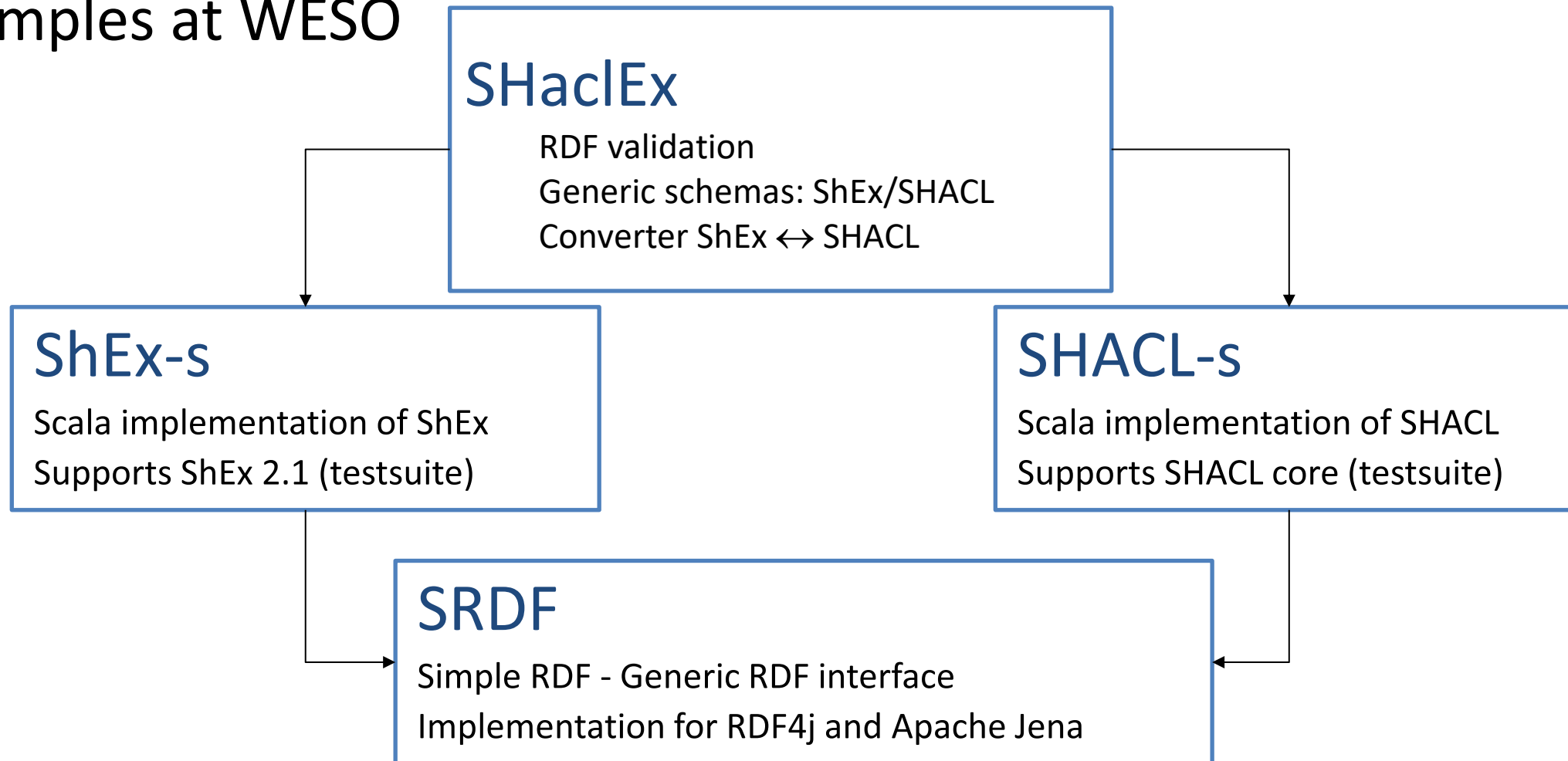
Online demos

Integrated in ontology editors

Continuous integration with Shapes

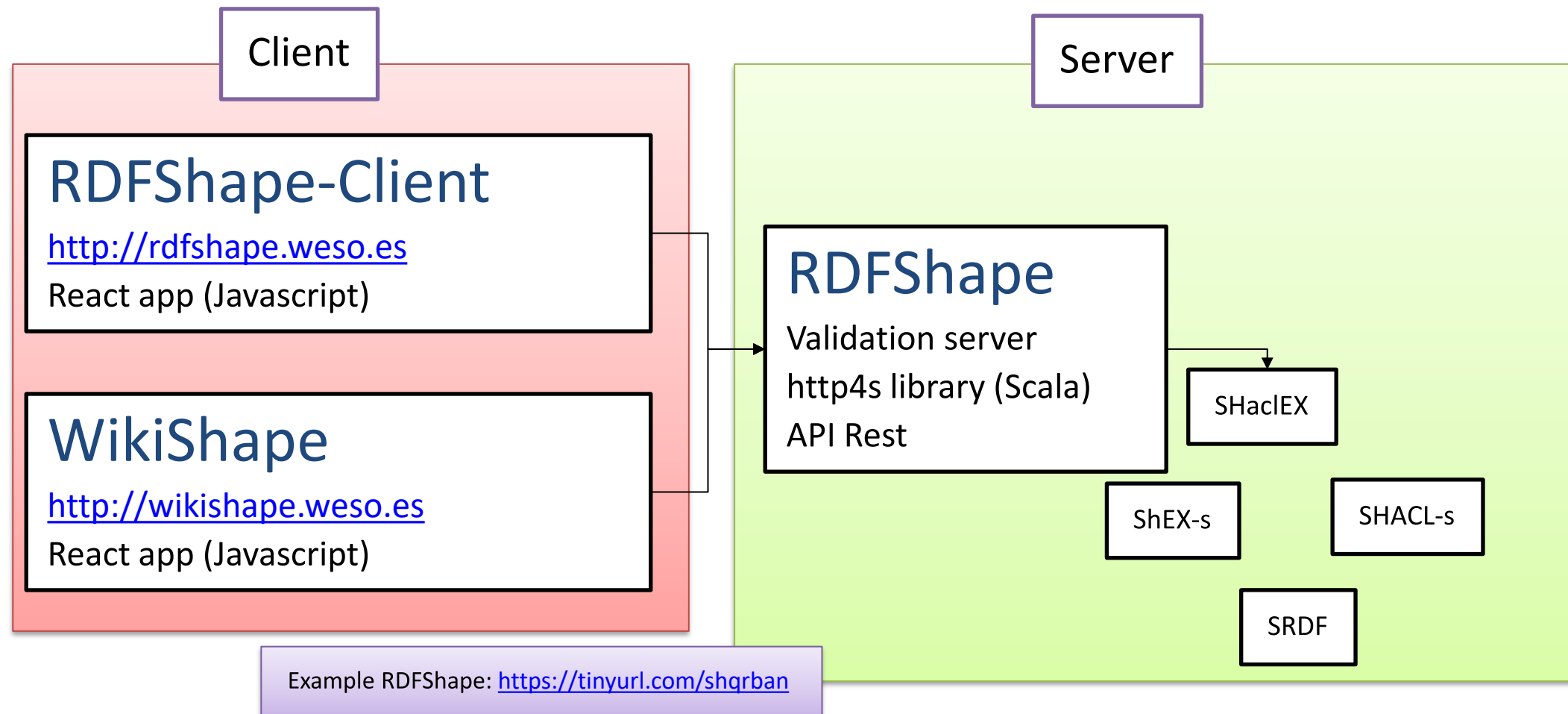
Libraries and command line validators

Examples at WESO



Online demos

Web Demos and playgrounds



Integrating shapes with other tools

TopBraid Composer

<https://www.topquadrant.com/technology/shacl/>

Ontology editors

SHACL plugin for protégé: <https://github.com/fekaputra/shacl-plugin>

ShEx plugin for protégé: <https://github.com/weso/protegeShEx>

Continuous integration with Shapes

Coexistence between ontologies/shapes

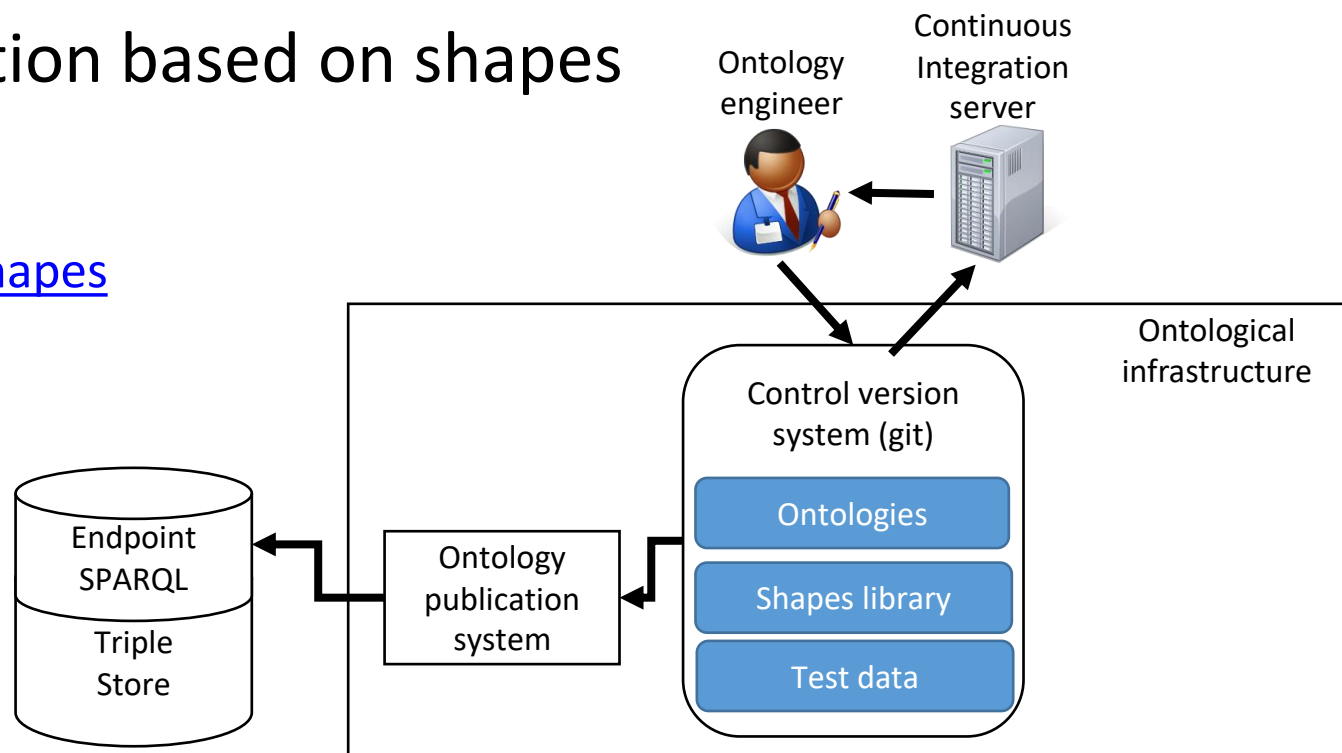
Shapes can validate the behaviour of inference systems

Shapes pre- and post- inference

TDD and continuous integration based on shapes

Gene Ontology Shapes:

<https://github.com/geneontology/go-shapes>



Continuous integration with Shapes

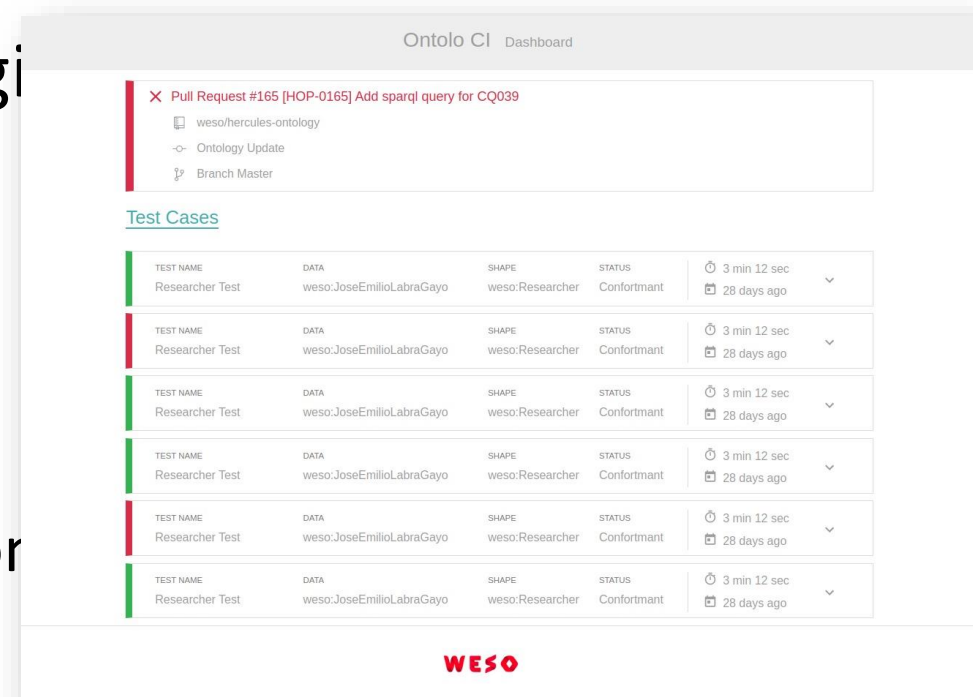
Ontolo-ci: <https://github.com/weso/ontolo-ci>

Developed as part of HERCULES-Ontology

Test-Driven-Development applied to Ontology

Input:

- Ontologies
- Shapes
- Test data
- Input shape map (SPARQL competency question)
- Expected result shape map



Creating shapes

Shapes editors

- Text-based editors

- Visual editors and visualizers

Obtaining shapes from...

- Spreadsheets

- RDF data

- Ontologies

- Other schemas (XML Schema)



Text-based editors

YaSHE: Forked from YASGUI: <http://www.weso.es/YASHE/>

Syntax highlighting

Auto-completion



The screenshot displays the YaSHE text-based editor interface. The editor window shows a SPARQL query with syntax highlighting. The query is as follows:

```
1 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
2 prefix wd: <http://www.wikidata.org/entity/>
3 prefix wdt: <http://www.wikidata.org/prop/direct/>
4
5 # Example SPARQL query: select ?researcher where { ?researcher wdt:P106 wd:Q1650915 } limit 5
6
7 <Researcher> EXTRA wdt:P31 wdt:P106 {
8   wdt:P31 [ wd:Q5 ] ; # Instance of = human
9   wdt:P106 [ wd:Q1650915 ] ; # Occupation = researcher
10  wdt:P101 @<Discipline> * ; # Field of work
11  wdt:P496 xsd:string ? ; # ORCID-ID
12  wdt:P1153 xsd:string ? ; # Scopus-Author ID
13 }
14
```

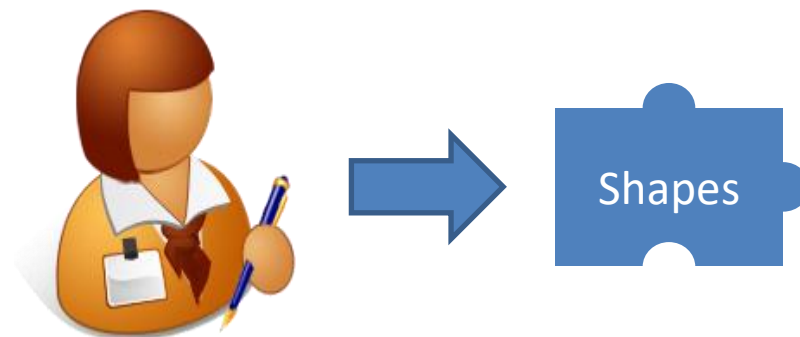
An auto-completion tooltip is visible for the property `wdt:P1153`, showing the text: "Scopus Author ID (P1153) identifier for an author assigned in Scopus bibliographic database". The editor interface includes a vertical scrollbar on the right and a toolbar with icons for undo, redo, save, and other standard editing functions.

Shapes author tools

Top Braid Composer

UnSHACLed

ShEx-Author



Shapes author tools: Top Braid Composer

Form based editor

Integrated with Top Braid product

The screenshot displays the 'Node Shape Form' for a shape named 'Person'. The form is organized into several sections:

- Annotations:** A section for adding annotations.
- Constraints:** A section for adding constraints, including `sh:property`, `sh:sparql`, and `sh:target`.
- Targets:** A section for adding targets, including `sh:targetClass`, `sh:targetObjectsOf`, `sh:targetSubjectsOf`, `sh:targetNode`, and `sh:target`.
- Other Properties:** A section for adding other properties, including `dash:abstract`, `dash:applicableToClass`, `dash:closedByTypes`, `dash:defaultViewForRole`, and `dash:resourceAction`.

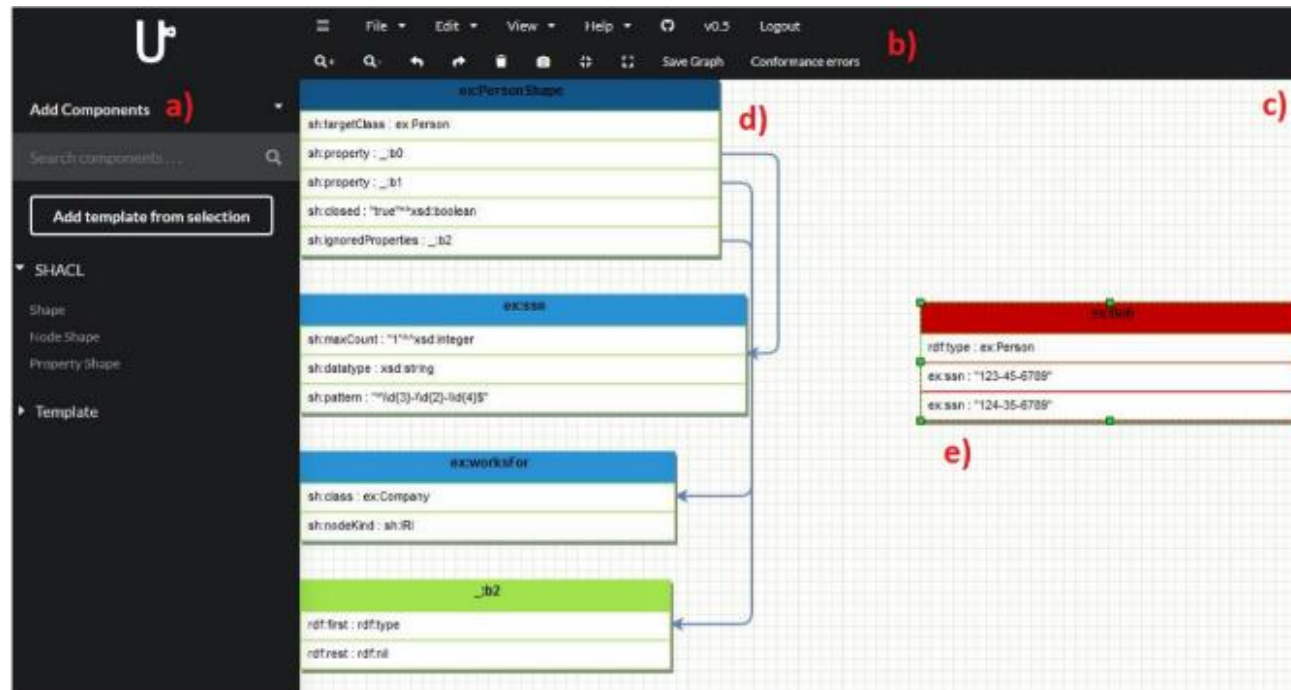
A dialog box titled 'Create sh:property for Person' is open, showing the following fields:

- Predicate:** `knows`
- Also globally declare rdf:Property:** ☐
- Display Name:** `name`
- Description:** `Name of a person`
- Count:** `Unlimited [0..*]`
- Node kind:** `Literals`
- Class:** (empty)
- Datatype:** `xsd:string`
- Value shape:** (empty)

The dialog has 'OK' and 'Cancel' buttons at the bottom right.

Shapes author tools: UnSHACLed

Visual SHACL Editor in Javascript



B. De Meester, P. Heyvaert, A. Dimou, and R. Verborgh, “Towards a Uniform User Interface for Editing Data Shapes,” in Proceedings of the 4th International Workshop on Visualization and Interaction for Ontologies and Linked Data, 2018, vol. 2187.

Shapes author tools: ShEx Author

ShEx-Author: Inspired by Wikidata Query Service

2 column: Visual one synchronized with text based

The screenshot displays the ShExAuthor web application interface. The top navigation bar includes the ShExAuthor logo, a hamburger menu, and links to YASHE and About me. A 'Fork me on GitHub' badge is visible in the top right corner of the interface.

The interface is divided into two main columns:

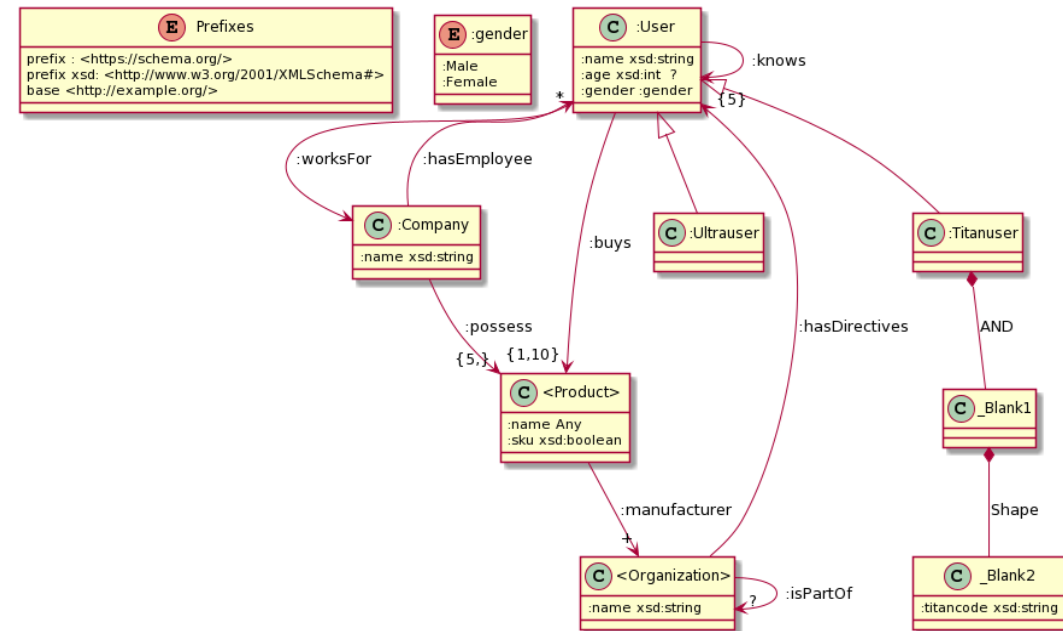
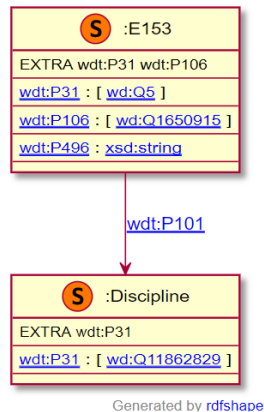
- Visual Editor (Left Column):** This section is titled 'Assistant' and contains two 'Shape' editors. The first shape is named 'Researcher' and is of type 'IriRef'. It lists five triples with the following properties and values: wdt:P31, wdt:P106, wdt:P101, wdt:P496, and wdt:P1153. The second shape is named 'Discipline' and is also of type 'IriRef'. It lists one triple with the property wdt:P31. Each shape editor includes a '+ Triple' button to add new triples.
- Text Editor (Right Column):** This section displays the ShEx text representation of the shapes defined in the visual editor. The text is as follows:

```
1 PREFIX xsd:    <http://www.w3.org/2001/XMLSchema#>
2 PREFIX wd:     <http://www.wikidata.org/entity/>
3 PREFIX wdt:    <http://www.wikidata.org/prop/direct/>
4
5 <Researcher> {
6   wdt:P31      IRI ;
7   wdt:P106     IRI ? ;
8   wdt:P101     @<Discipline>? ;
9   wdt:P496     xsd:string ? ;
10  wdt:P1153     xsd:string * ;
11 }
12
13 <Discipline> {
14   wdt:P31      IRI * ;
15 }
16
17
```

Shapes visualization

Integrated in RDFShape/Wikishape

- [UMLSHacLEX](#) UML diagrams for ShEx
- [ShUMLex](#): Conversion to UML through XMI



Shapes from spreadsheets

SKOS-Play was used at ELI to generate SHACL shapes from Excel

ShExstatements: <https://shexstatements.toolforge.org/>

ShExCSV: CSV representation of Shapes

Hermes: ShExCSV processor, <https://github.com/weso/hermes>



Generating Shapes from RDF data

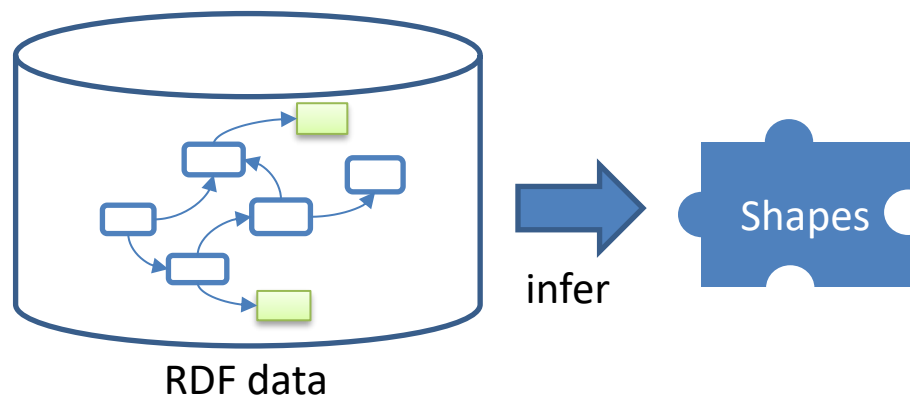
Useful use case in practice

Some prototypes

sheXer: <http://shexer.weso.es/>

RDFShape: <http://rdfshape.weso.es>

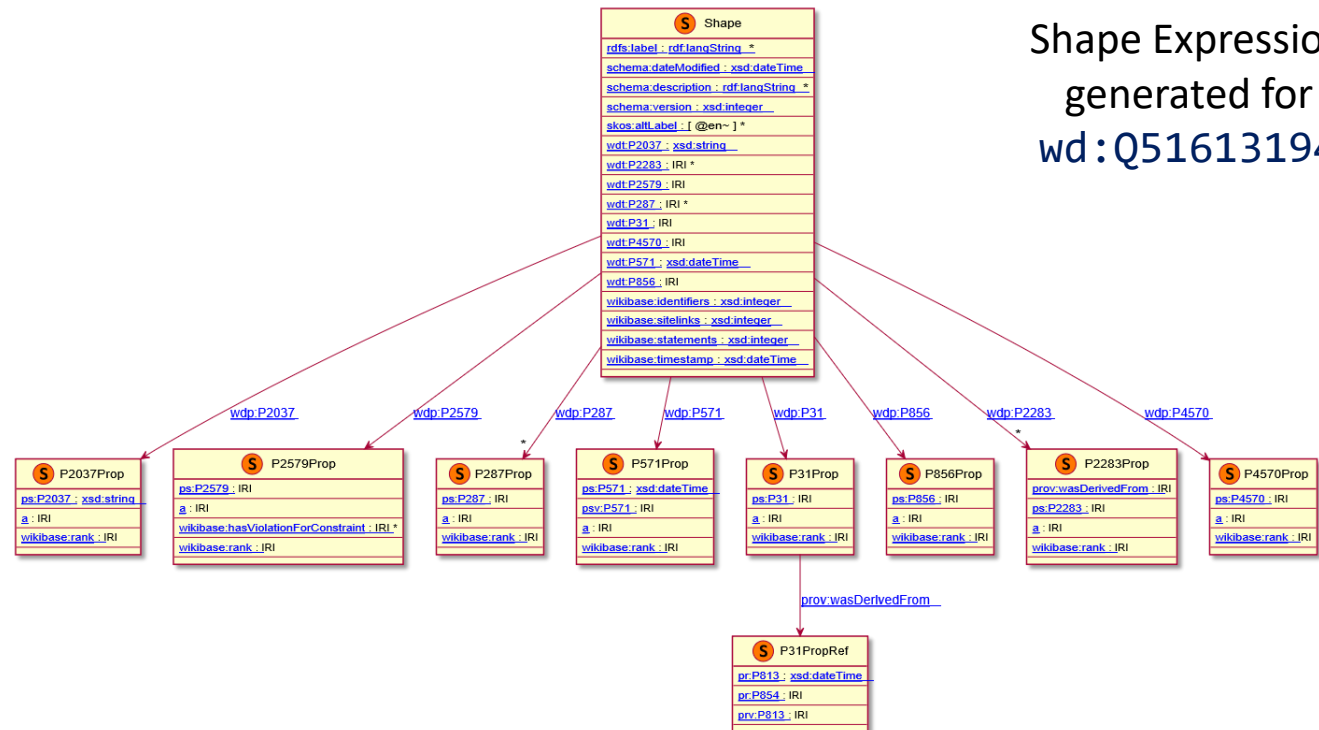
ShapeDesigner: <https://gitlab.inria.fr/jdusart/shexjapp>



Try it with RDFShape:
<https://tinyurl.com/y8pjcbfy>

Shapes from data: RDFShape

RDFShape/Wikishape implement a basic prototype to derive Shapes from RDF data



Shapes from data: sheXer

sheXer: <http://shexer.weso.es/>

Implemented in Python
Configuration options

sheXer

Welcome to an online demo of sheXer, a library to perform automatic extraction of ShEx schemata in RDF graphs

Introduction

sheXer receives an RDF graph, a bunch of configurations params and it gives back ShEx schemata associated to groupings of nodes within the graph.

This webapp is an online demo of **sheXer**. You can provide an RDF graph in several ways, select your target individuals to build shapes, tune some config values and then get the inferred shapes. This demo does not allow you to compute huge graphs nor to configure every option of **sheXer**. To reach the full potential of the tool you may prefer to install the library **sheXer**, which is available for Python. This webapp and the library itself are prototypes under development. You can follow its updates, report bugs or make any suggestion at [the sheXer github repository](#).

The way to use the code and the meaning of each config param is documented in [the repository of sheXer](#). Installation instructions are available in the repository as well.

Instructions about how to configure each field of this online demo are available at [the bottom of this page](#).

In case you want to work against the Wikidata endpoint, press the following button to fill the configuration params with some recommended values:

Wikidata mode!

Input

Raw input Remote input

Choose a way to provide a remote graph:

☐ Remote Graph

Shapes from data: ShapeDesigner

<https://gitlab.inria.fr/jdusart/shexjapp>

The screenshot displays the ShapeDesigner application interface. The top menu bar includes File, Schema, Graph, Pattern, Query, and Help. Below the menu, there are tabs for Conception and ShEx Validation. The main workspace is divided into three sections: Patterns, Queries, and Schema.

Patterns Section:

Name	Description
Wikidata direct properties	{ wdt:~ __; }
Wikidata properties and their qualifiers	{ p:P~ {ps:~ __; psn:~ __; a[__]; } };
All properties	{ p:P~ {ps:~ __; psn:~ __; a[__]; } wdt:~ __; ~ __ }
provenance for population	{ p:P1082 {ps:~ __; pq:~ __; prov:~ __ } }

Buttons: Delete, Duplicate, Add, Edit

Queries Section:

Name	Description
Select all nodes with given type	SELECT ?x WHERE { ?x wdt:P31 <replace-this-by-iri-of-...> }
Select 10 big city	SELECT ?x WHERE { ?x wdt:P31 wd:Q1549591. } LIMIT 10

Buttons: Delete, Duplicate, Add, Edit

Schema Section:

```
<?xml version="1.0" encoding="UTF-8"?>
<PREFIX wdt: <http://www.wikidata.org/entity/statement/>
3 PREFIX ontolex: <http://www.w3.org/ns/lemon/ontolex#>
4 PREFIX wdata: <http://www.wikidata.org/wiki/Special:EntityData/>
5 PREFIX p: <http://www.wikidata.org/prop/>
6 PREFIX wd: <http://www.wikidata.org/entity/>
7 PREFIX wdno: <http://www.wikidata.org/prop/novalue/>
8 PREFIX wdref: <http://www.wikidata.org/reference/>
9 PREFIX ps: <http://www.wikidata.org/prop/statement/>
10 PREFIX pq: <http://www.wikidata.org/prop/qualifier/>
11 PREFIX cc: <http://creativecommons.org/ns#>
12 PREFIX wdt: <http://www.wikidata.org/prop/direct/>
13 PREFIX wdv: <http://www.wikidata.org/value/>
14 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
15 PREFIX dct: <http://purl.org/dc/terms/>
16 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
17 PREFIX prov: <http://www.w3.org/ns/prov#>
18 PREFIX wdtn: <http://www.wikidata.org/prop/direct-normalized/>
19 PREFIX pqv: <http://www.wikidata.org/prop/qualifier/value/>
20 PREFIX prv: <http://www.wikidata.org/prop/reference/value/>
21 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
22 PREFIX psv: <http://www.wikidata.org/prop/statement/value/>
23 PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
24 PREFIX geo: <http://www.opengis.net/ont/geosparql#>
25 PREFIX bd: <http://www.bigdata.com/rdf#>
26 PREFIX pqn: <http://www.wikidata.org/prop/qualifier/value-normalized/>
27 PREFIX pr: <http://www.wikidata.org/prop/reference/>
28 PREFIX prn: <http://www.wikidata.org/prop/reference/value-normalized/>
29 PREFIX psn: <http://www.wikidata.org/prop/statement/value-normalized/>
30 PREFIX schema: <http://schema.org/>
31 PREFIX wikibase: <http://wikiba.se/ontology#>
32 PREFIX demo: <http://inria.fr/ShapeDesigner/demo>

# Result for pattern { p:P1082 {ps:~ __; pq:~ __; prov:~ __ } } and query SELECT ?x WHERE { ?x wdt:P31 wd:Q1549591. } LIMIT 10
demo:BigCityPopulation {
  # population; frequency: [10/10]
  p:P1082 {
    # population; frequency: [99/99]
    psv:P1082 [ <http://www.wikidata.org/value/> ] ;
    # population; frequency: [99/99]
    ps:P1082 xsd:decimal ;
    # point in time; frequency: [99/99]
    pqv:P585 [ <http://www.wikidata.org/value/> ] + ;
    # point in time; frequency: [99/99]
    pq:P585 xsd:dateTime + ;
    prov:wasDerivedFrom [ <http://www.wikidata.org/reference/> ] * ;
    # determination method; frequency: [28/99]
    pq:P459 [ <http://www.wikidata.org/entity/> ] ? ;
    # publisher; frequency: [1/99]
    pq:P123 [ <http://www.wikidata.org/entity/> ] ?
  } +
}
```

Shapes from RDF data

RDFShape allows to infer basic shapes automatically

The screenshot shows the WikiShape web application interface. At the top, there is a navigation bar with links for WikiShape, Entity, Schema, Property, Query, and Help. The main heading is "Extract schema from Wikidata entities". Below this, there is a search bar containing the Wikidata ID "Q63241966 (Jesualdo Tomás Fernández-Breis)" and a language dropdown set to "en". Below the search bar, the entity name "Jesualdo Tomás Fernández-Breis" and its Wikidata URL "http://www.wikidata.org/entity/Q63241966" are displayed, along with the role "researcher". A blue "Extract Schema" button is present. Below the button, the text "Shape extracted" is shown. The extracted shape is displayed in a code editor with line numbers 1 through 14. The shape is an RDF graph with various prefixes and properties. At the bottom, there is a "Details" section and a "Permalink" button.

WikiShape Entity Schema Property Query Help

Extract schema from Wikidata entities

Q63241966 (Jesualdo Tomás Fernández-Breis) × Language en

Jesualdo Tomás Fernández-Breis http://www.wikidata.org/entity/Q63241966 researcher

Extract Schema

Shape extracted

```
1 prefix sx: <http://weso.es/ns/shex/>
2 prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 prefix wdt: <http://www.wikidata.org/prop/direct-normalized/>
4 prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5 prefix wdt: <http://www.wikidata.org/prop/direct/>
6 prefix xsd: <http://www.w3.org/2001/XMLSchema#>
7 prefix wdp: <http://www.wikidata.org/prop/>
8 prefix wikibase: <http://wikiba.se/ontology#>
9 prefix schema: <http://schema.org/>
10 <Shape> {
11   rdfs:label rdf:langString* // sx:maxLength 4;
12   schema:dateModified xsd:dateTime;
13   schema:description rdf:langString* // sx:maxLength 3;
14   schema:version xsd:integer;
```

Details

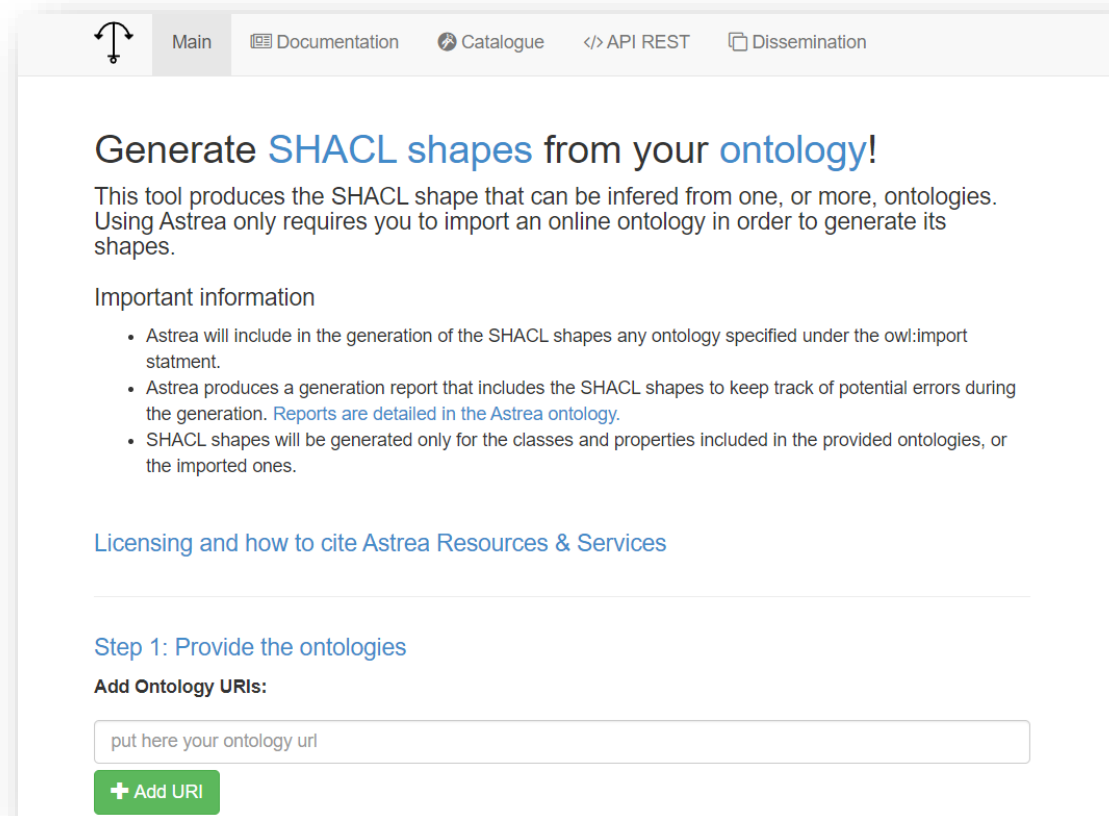
Permalink

Shapes from ontologies

Astrea*: <https://astrea.linkeddata.es/>

Generates SHACL shapes from OWL ontologies

Mappings between ontology construct patterns to SHACL



*Cimmino, A., Fernández-Izquierdo, A., & García-Castro, R. (2020). Astrea: automatic generation of SHACL shapes from ontologies. In European Semantic Web Conference

Other uses of Shapes

UIs and shapes

Generating code from Shapes

Shapes and rules

UIs and shapes

Shapes can provide hints to generate user interfaces/forms

SHACL core defines a basic vocabulary: sh:group, sh:order, ...

ShEx annotations can also be used to define UI declarations

Example: UI ontology annotations

UIs and Shapes: ShExPath and ShEx-Forms

ShEx Path can be used to point to parts of a ShEx schema

<https://shexspec.github.io/spec/ShExPath>

ShEx generated forms demo based on UI ontology:

<https://ericprud.github.io/shex-form/?manifestURL=examples/manifest.json>

UIs and shapes: TopQuadrant

Form generation from SHACL

DASH vocabulary:

<http://datashapes.org/forms.html>

Holger's Address x

Explore ▾ Modify ▾ Cancel Preview Save Changes Australian address shape ▾

Holger's Address
aussies:HolgersAddress

▼ Address

street address: 3 Teewah Close

suburb: Kewarra Beach

state: QLD ▾

postal code: 48791

▼ Contact

email: + holger@knublauch.com

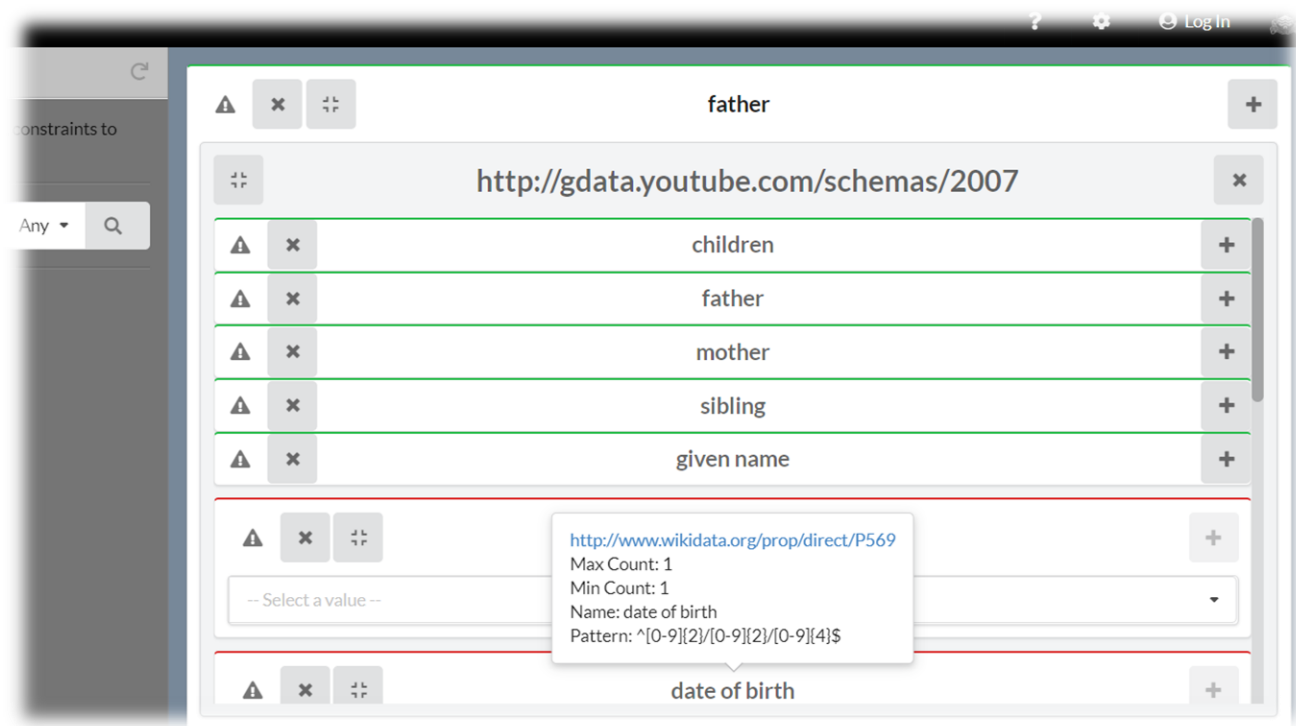
holger@topquadrant.com

phone number: +

UIs and shapes: Schímatos

<http://schimatos.org/>

It will be presented at ISWC20



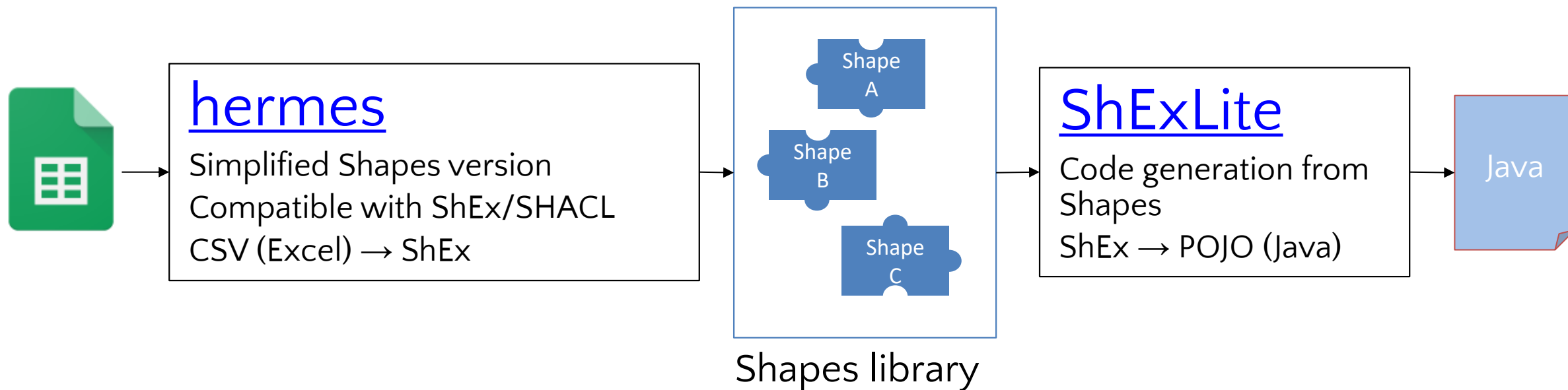
Generating code from shapes

Generate domain model from shapes

Entities (pseudo-shapes) defined with Excel (Google spreadsheets)

Shapes generation from those templates

Java code generation (POJOs) from those shapes



Generating code from shapes

Domain model based on Shapes

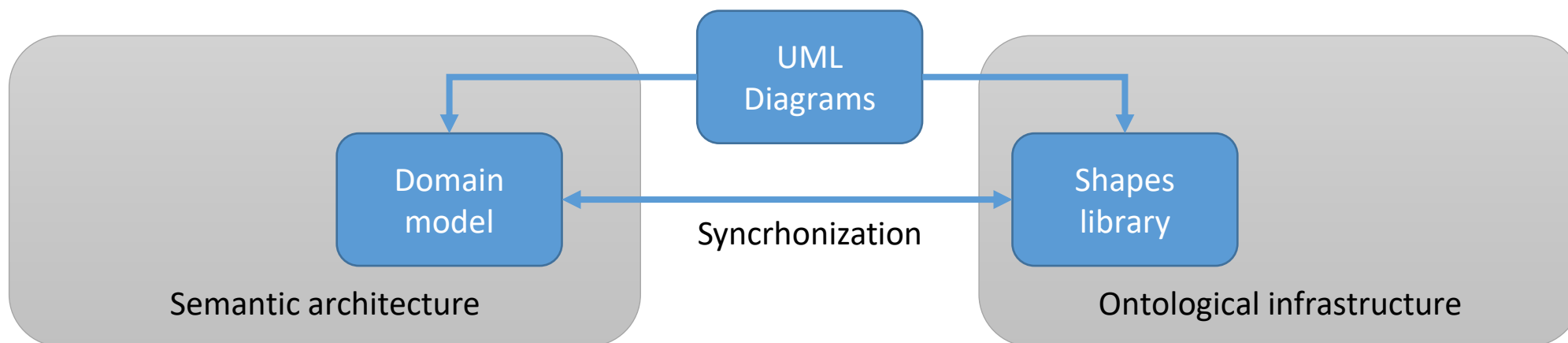
Clean architecture pattern

Domain model as central element

Simple classes (POJO): Plain Old Java Objects

Shapes synchronization

Application logic and services based on domain model



Shapes and rules

SHACL Advanced Features describes SHACL rules

```
:Rectangle a rdfs:Class, sh:NodeShape ;  
  rdfs:label "Rectangle" ;  
  sh:property [ sh:path :height ;  
    sh:datatype xsd:integer ;  
    sh:maxCount 1 ; sh:minCount 1 ;  
    sh:name "height" ] ;  
  sh:property [ sh:path :width ;  
    sh:datatype xsd:integer ;  
    sh:maxCount 1 ; sh:minCount 1 ;  
    sh:name "width" ] ;  
  sh:rule [ a sh:TripleRule ;  
    sh:subject sh:this ;  
    sh:predicate rdf:type ;  
    sh:object :Square ;  
    sh:condition :Rectangle ;  
    sh:condition [  
      sh:property [  
        sh:path :width ;  
        sh>equals :height ;  
      ] ; ] ; ] .
```

```
:I a :Rectangle .  
:N a :Rectangle ;  
  :height 2 ;  
  :width 3 .  
:S a :Rectangle ;  
  :height 4 ;  
  :width 4 .
```



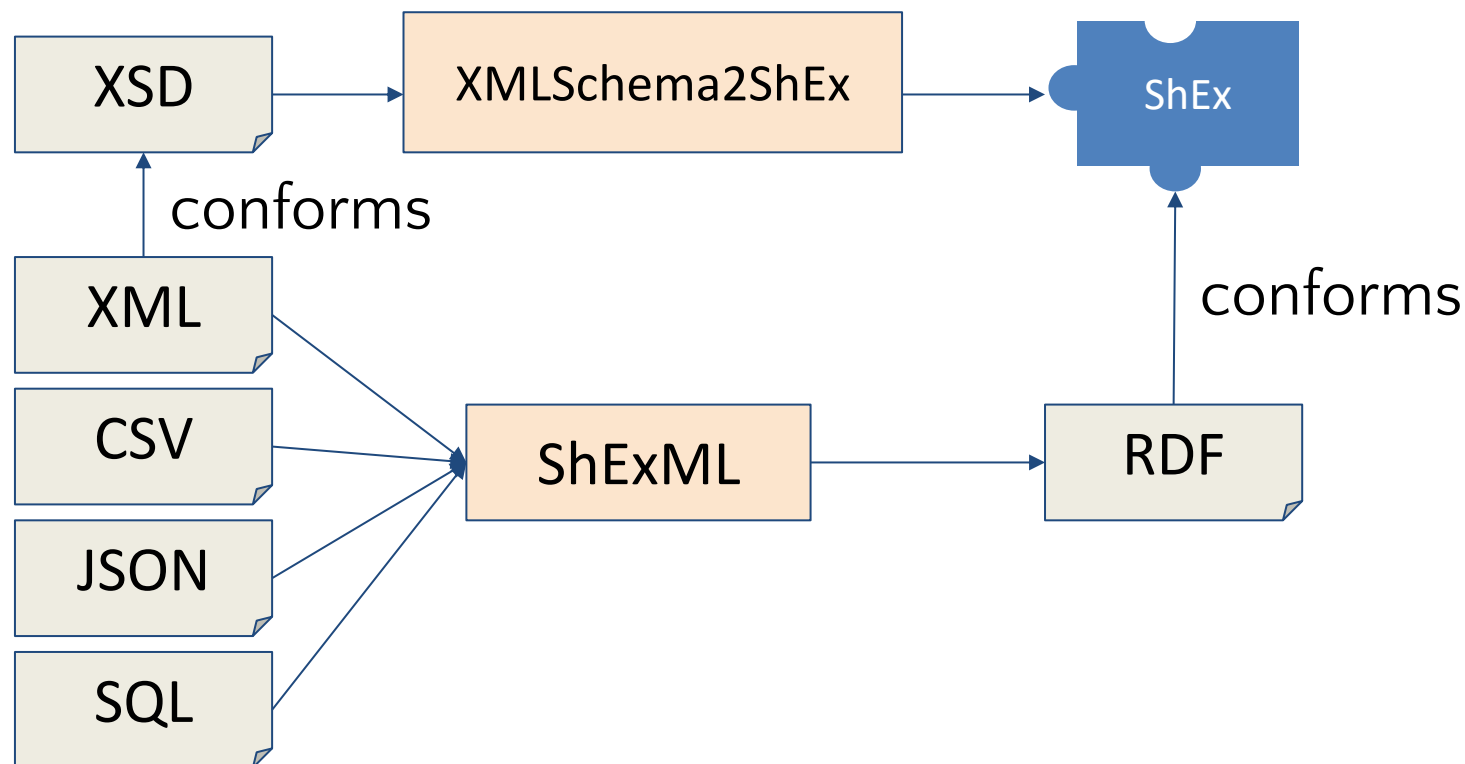
```
:S a :Square .
```

Shapes for data integration

[XMLSchema2ShEx](#): Convert XML Schemas to shapes

[ShExML](#): Domain specific language to convert data to RDF

Input formats: CSV, XML, JSON, SQL



Shapes ecosystems

Wikidata provides a whole ShEx ecosystem

Entity schemas can evolve and relate between each other

Directory: https://www.wikidata.org/wiki/Wikidata:Database_reports/EntitySchema_directory

Different schemas for the same entities?

Some schemas stress some aspects while others stress others

Evolution of schemas

Searching entity schemas

Conclusions

ShEx and SHACL have had a great level of adoption

But there are other types of Knowledge Graphs

Much more work to do

New tools and challenges



Acknowledgments

Awesome Semantic Shapes:

<https://github.com/w3c-cg/awesome-semantic-shapes>

Special thanks to Vladimir Alexiev for starting it

People from ShEx community group: Tom Baker, Kat Thornton,
Andra Waagmeester,...