Human-Friendly RDF Graph Construction: Which one do you chose?

Ana Iglesias-Molina[1], **David Chaves-Fraga[1,2,3],**
Ioannis Dasoulas[2,3,4], Anastasia Dimou[2,3,4]

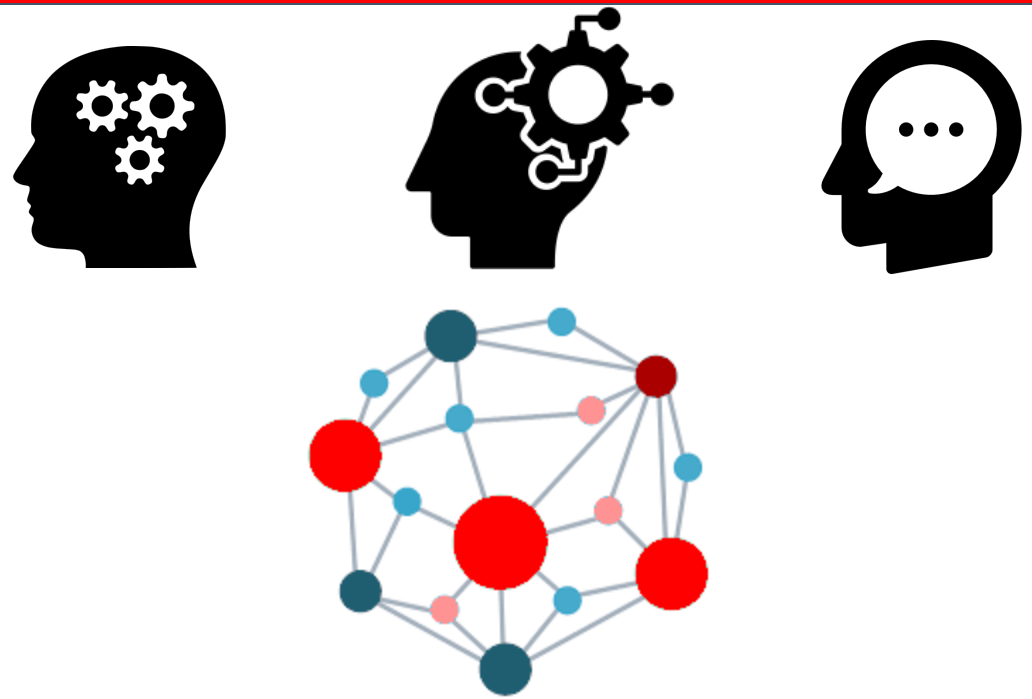
[1]Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
[2]Declarative Languages and Artificial Intelligence, KULeuven, Belgium
[3]Flanders Make, Belgium
[4]Leuven.AI, Belgium

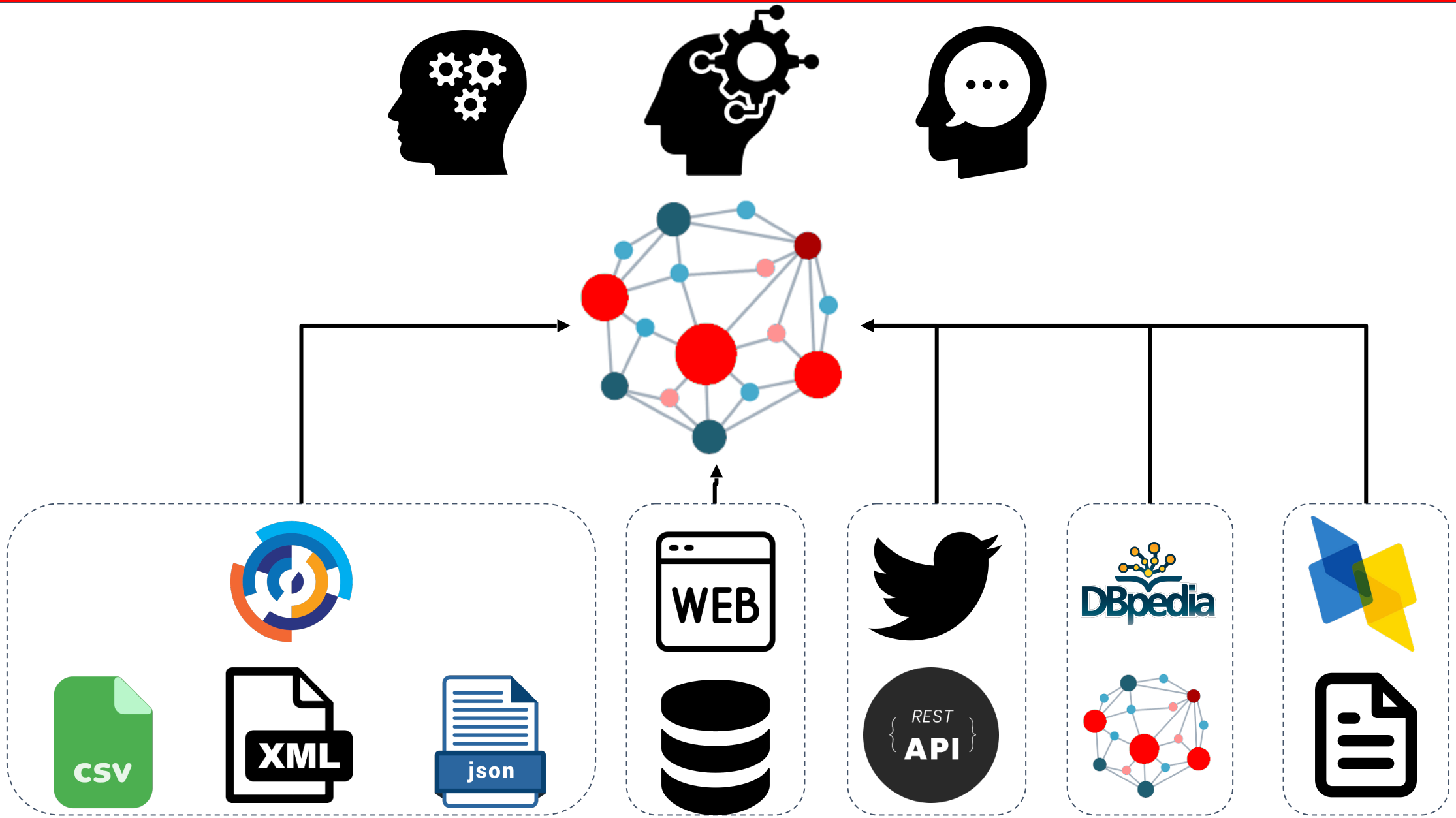
david.chaves@upm.es dchavesf

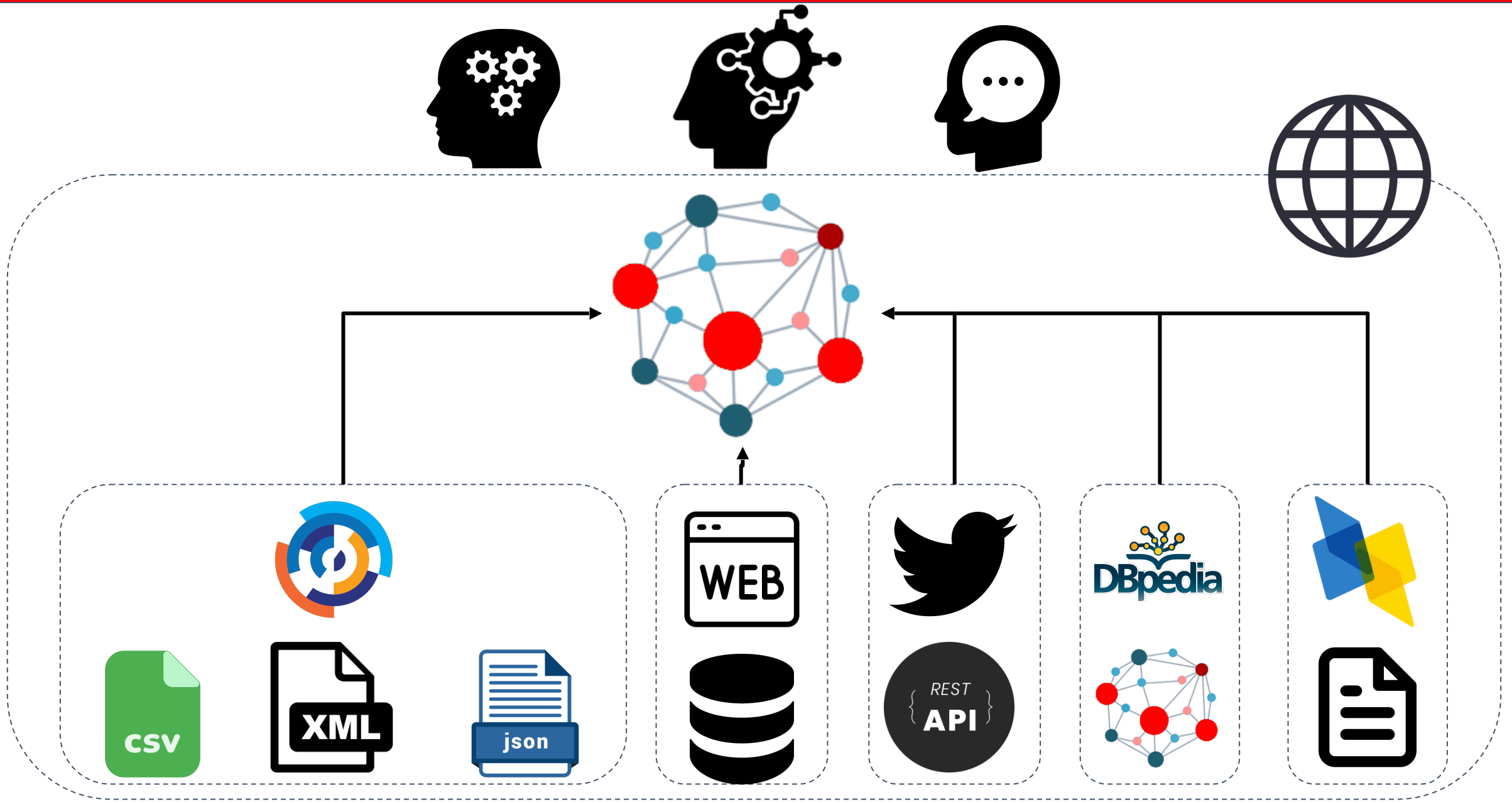
07/06/2023 ICWE

- Motivation
- Preliminaries: RML, YARRRML & RDF-star
- YARRRML-star & YATTER
- Validation & Comparison
- Conclusions and Future Work

- **Motivation**
- Preliminaries: RML, YARRRML & RDF-star
- YARRRML-star & YATTER
- Validation & Comparison
- Conclusions and Future Work

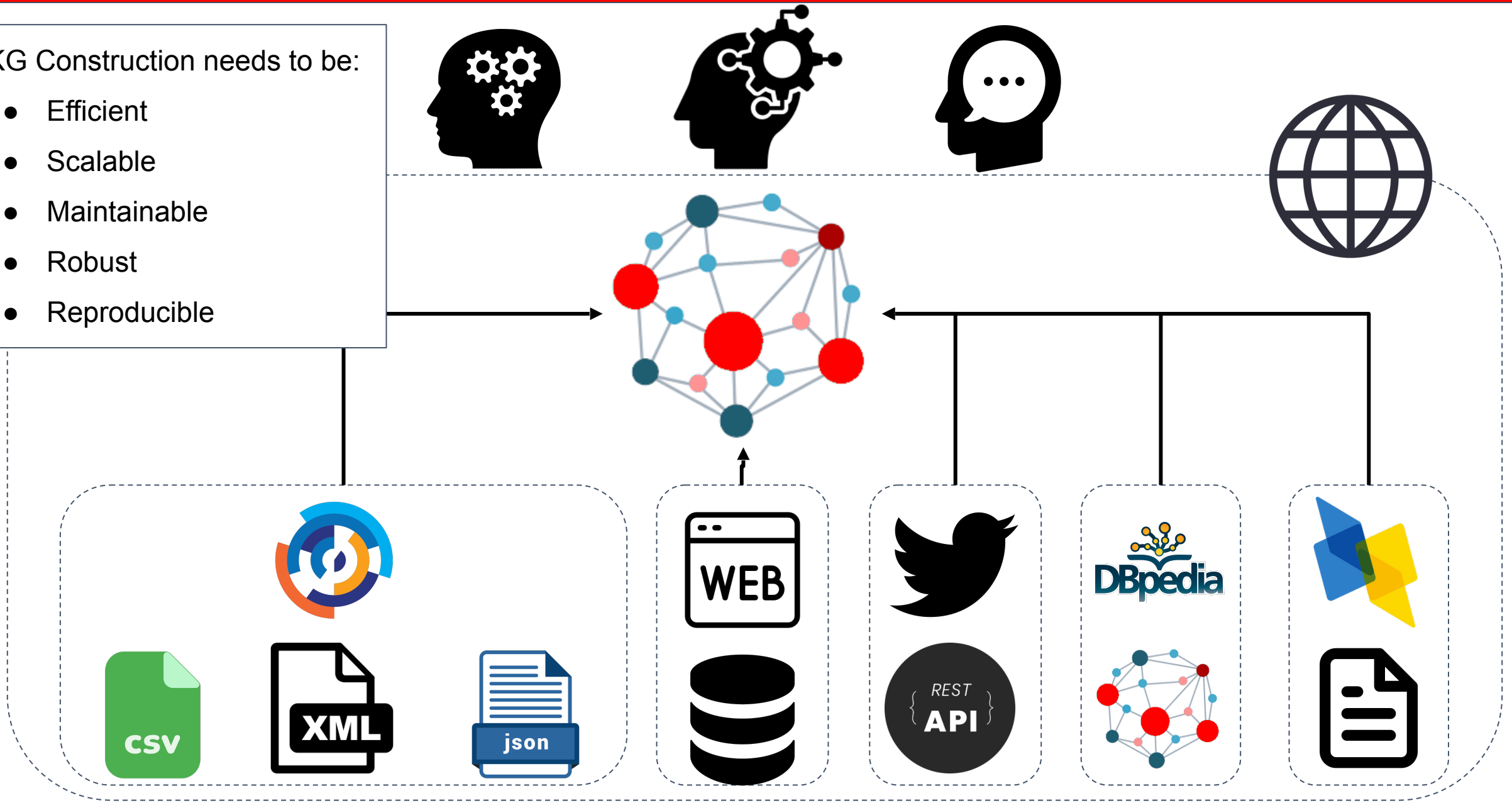



WEB
REST
API
DBpedia
csv
XML
json

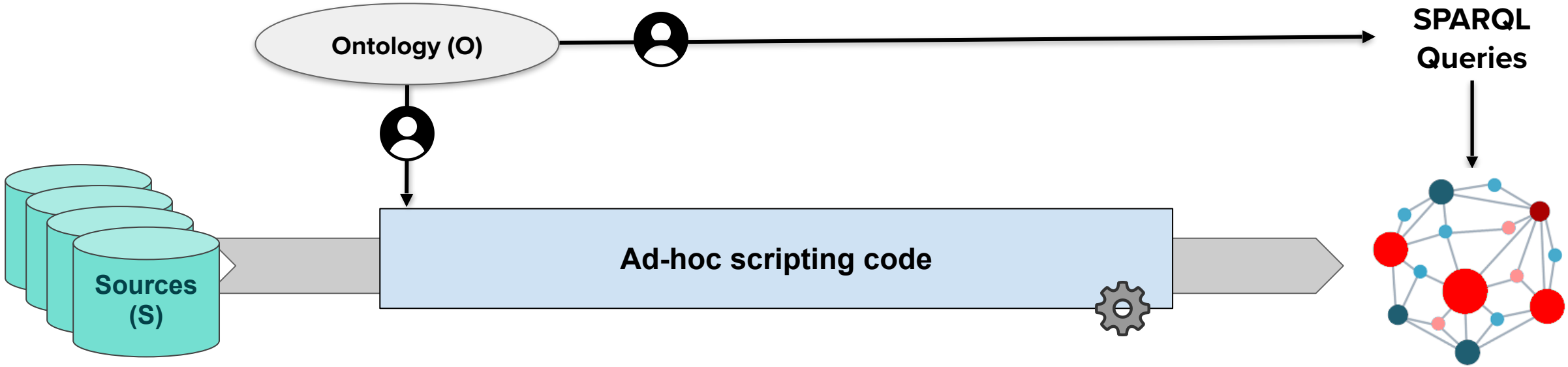

WEB
REST
API
DBpedia
CSV
XML
json

KG Construction needs to be:

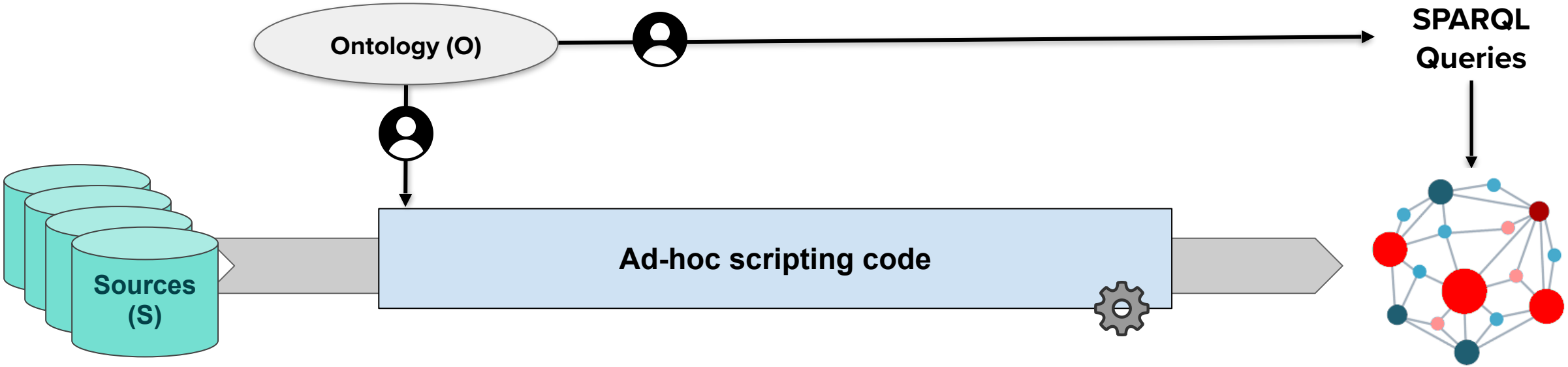
- Efficient
- Scalable
- Maintainable
- Robust
- Reproducible



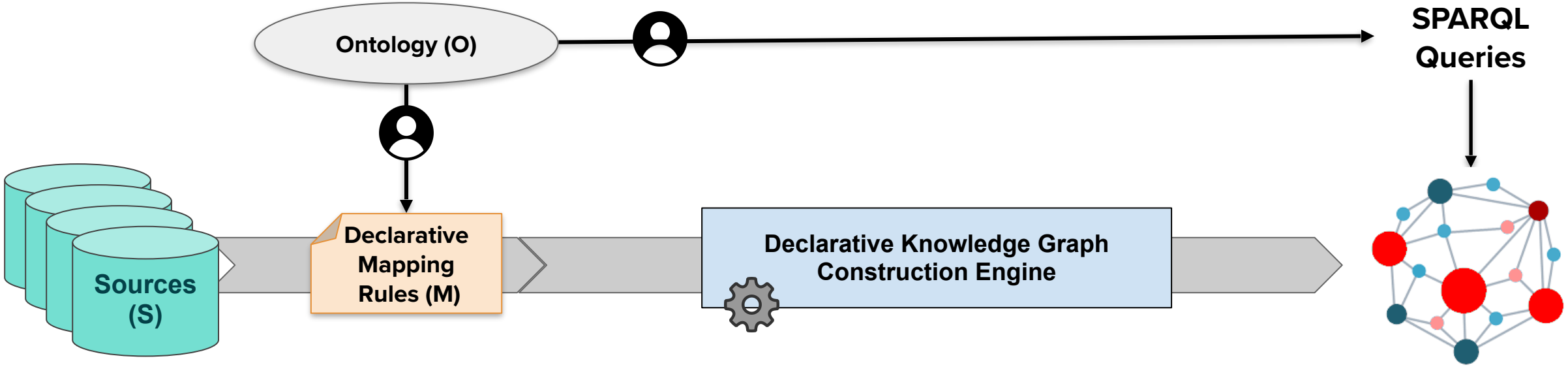
Knowledge Graph Construction: Scripting-based



Knowledge Graph Construction: Scripting-based



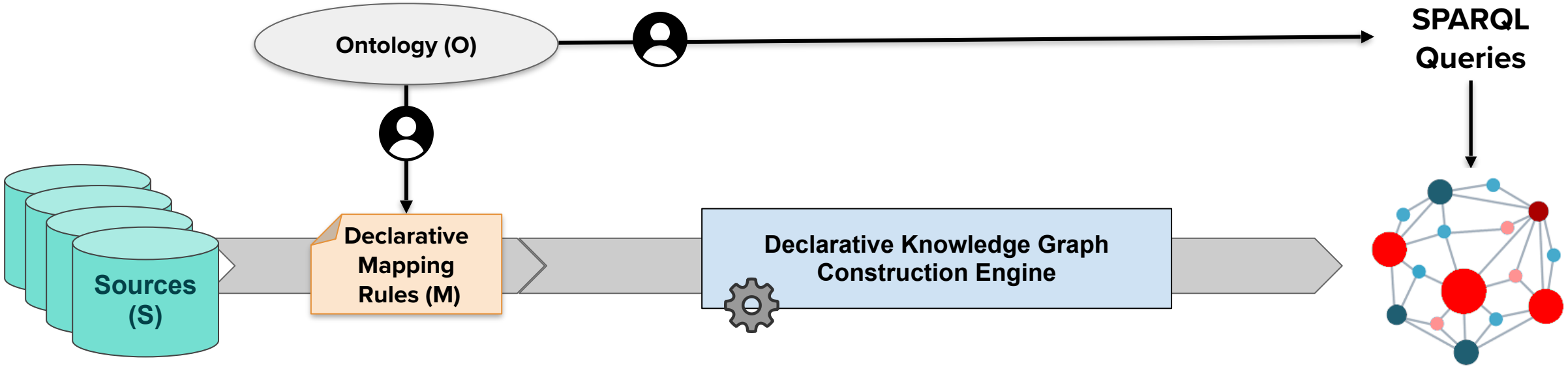
Efficient	Scalable	Maintainable	Robust	Reproducible
🤷🏽	🤷🏽	🤦	🤦	🤦



Knowledge Graph Construction = Data Integration System (DIS) = <S, M, O>

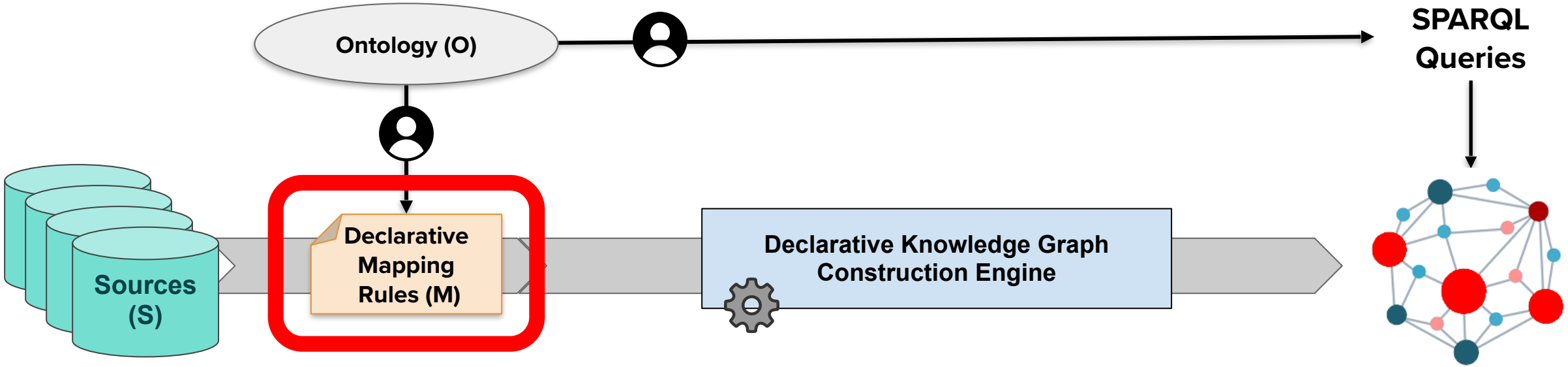
Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., & Rosati, R. (2008). Linking data to ontologies. In *Journal on data semantics X*
Lenzerini, M. Data integration: A theoretical perspective. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*

Knowledge Graph Construction = Data Integration System (DIS) = <S, M, O>



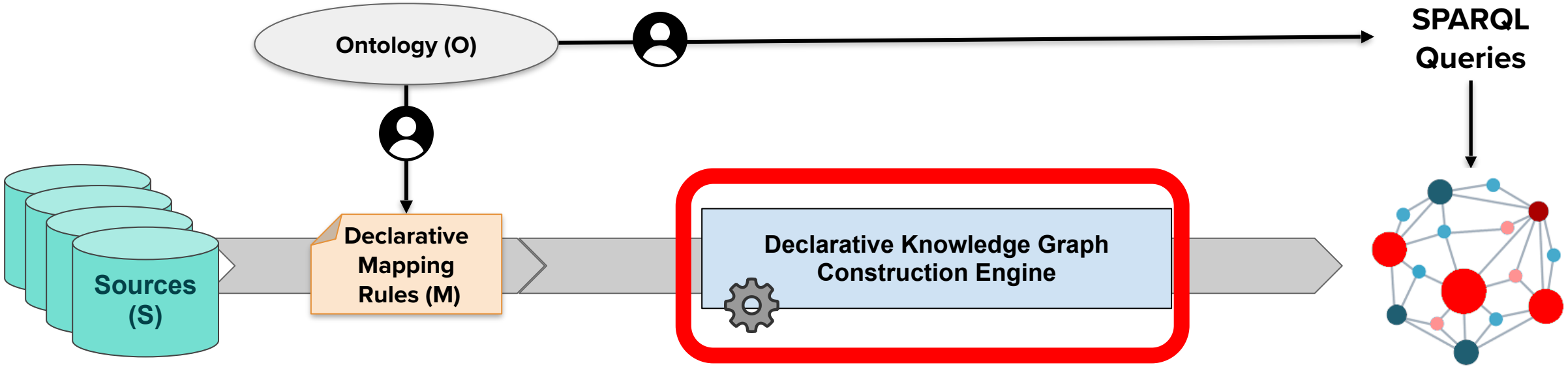
Efficient Scalable Maintainable Robust Reproducible

Knowledge Graph Construction = Data Integration System (DIS) = <S, M, O>



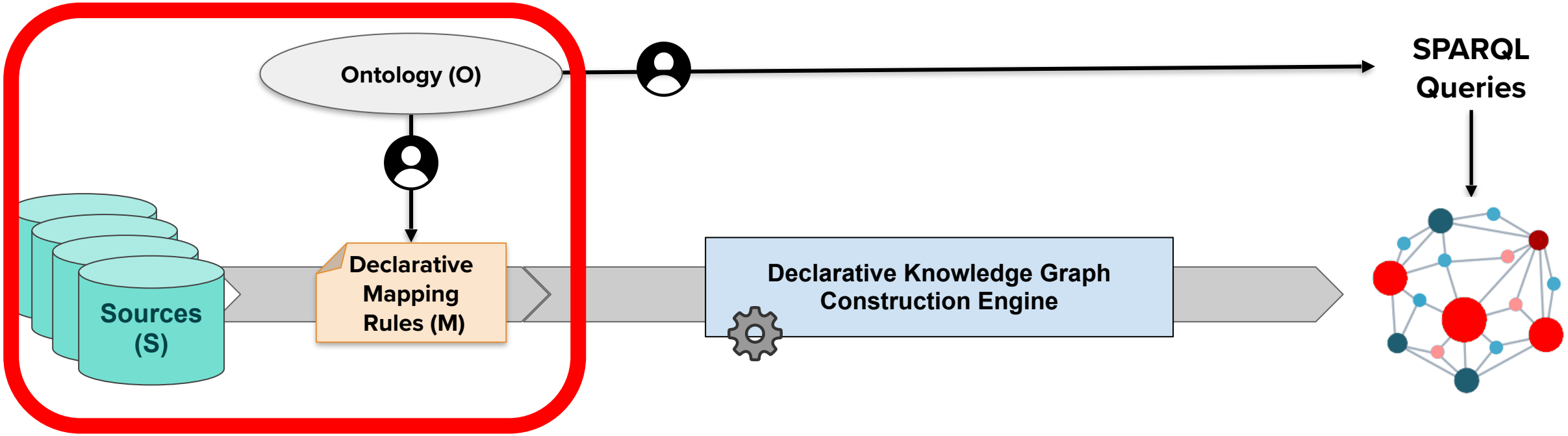
Efficient Scalable Maintainable Robust Reproducible

Knowledge Graph Construction = Data Integration System (DIS) = <S, M, O>



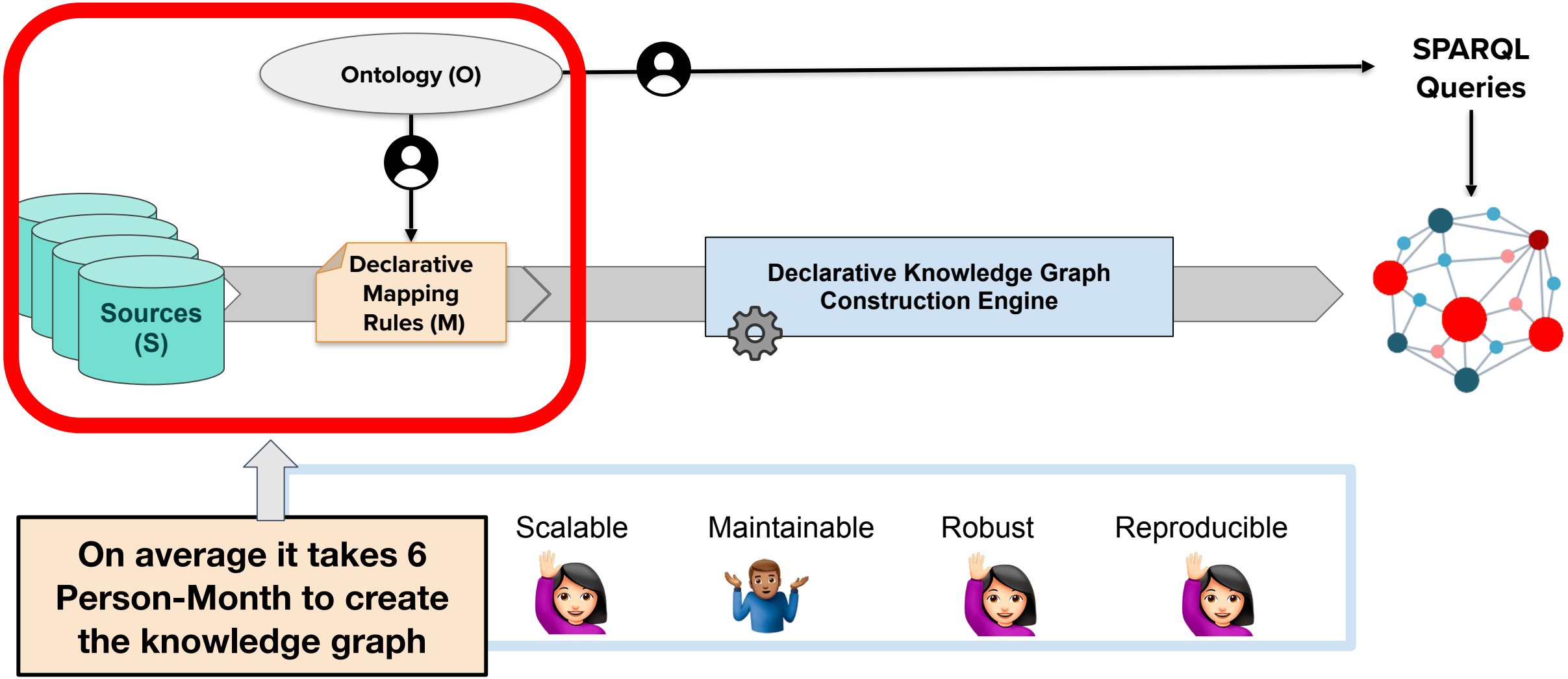
Efficient 
Scalable 
Maintainable 
Robust 
Reproducible 

Knowledge Graph Construction = Data Integration System (DIS) = <S, M, O>



Efficient Scalable Maintainable Robust Reproducible

Knowledge Graph Construction = Data Integration System (DIS) = <S, M, O>



On average it takes 6 Person-Month to create the knowledge graph

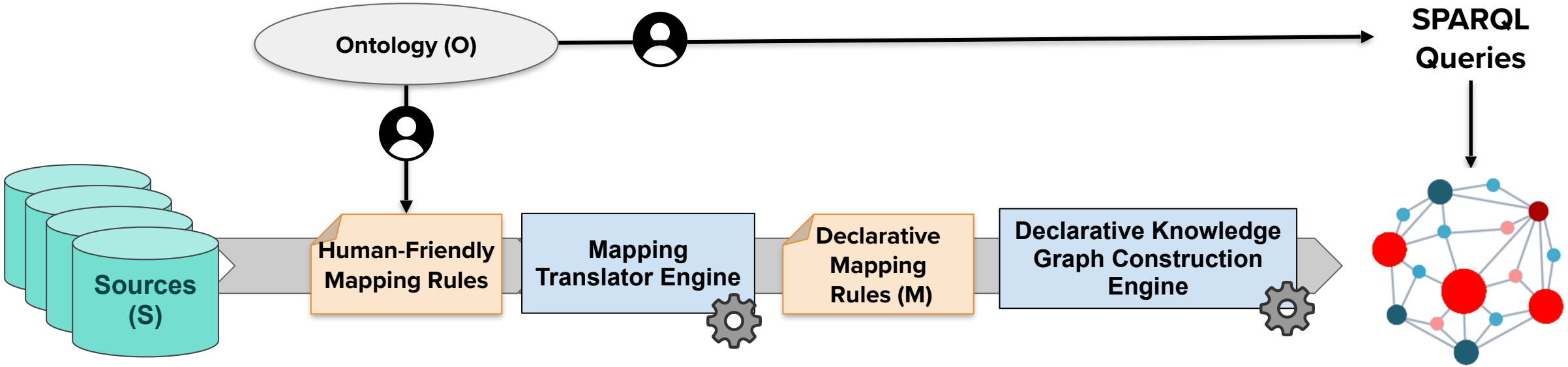
Scalable

Maintainable

Robust

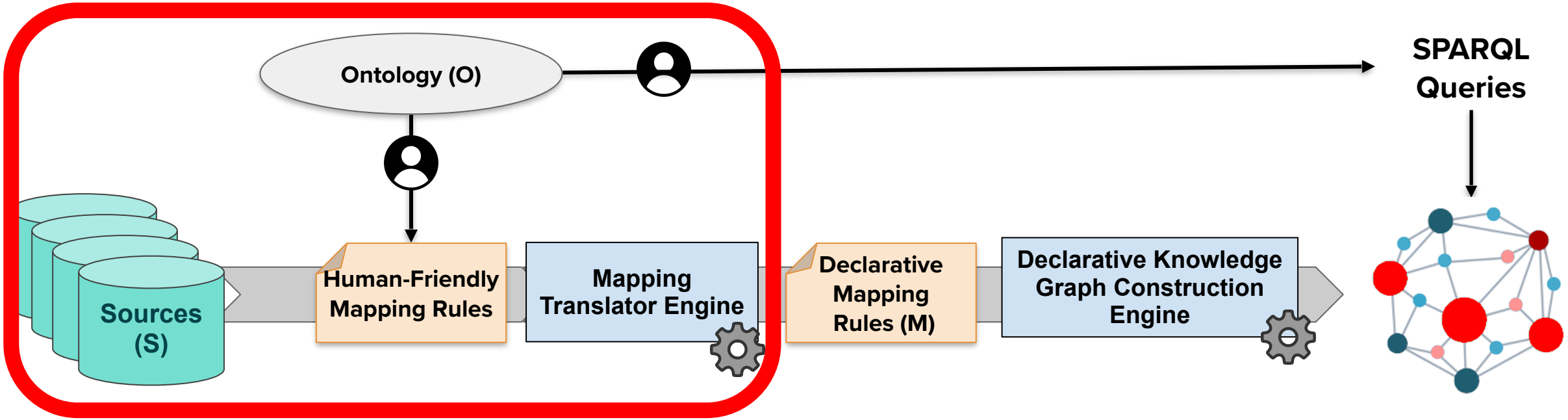
Reproducible

Knowledge Graph Construction = Data Integration System (DIS) = <S, M, O>



Efficient Scalable Maintainable Robust Reproducible

Knowledge Graph Construction = Data Integration System (DIS) = <S, M, O>



Efficient Scalable Maintainable Robust Reproducible

- Motivation
- **Preliminaries: RML, YARRRML & RDF-star**
- YARRRML-star & YATTER
- Validation & Comparison
- Conclusions and Future Work

One of the most used mapping languages is **RML**

- Extension of W3C Recommendation R2RML for heterogeneous data
- RDF syntax based (not very human-friendly)
- Specification → https://w3id.org/rml/portal

YARRRML is a human-based serialization of RML based on YAML

- Facilitates the construction of the rules
- Compact syntax
- Widely used in most of the real-life projects
- Even used by Google for their Entity Reconciliation API
- Specification → https://w3id.org/kg-construct/yarrrml

ID	DATE	MARK	PERSON
1	2022-03-21	4.80	Angelica
2	2022-03-19	4.85	Katerina

ID	DATE	MARK	PERSON
1	2022-03-21	4.80	Angelica
2	2022-03-19	4.85	Katerina

```
<#innerTM> a rr:TriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
   rr:template ":{PERSON}" ] ;
rr:predicateObjectMap [
   rr:predicate :jumps ;
   rml:objectMap [
      rml:reference
   "MARK" ] ] .
```

ID	DATE	MARK	PERSON
1	2022-03-21	4.80	Angelica
2	2022-03-19	4.85	Katerina


```
<#innerTM> a rr:TriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
   rr:template ":{PERSON}" ] ;
rr:predicateObjectMap [
   rr:predicate :jumps ;
   rml:objectMap [
      rml:reference
   "MARK" ] ] .
```

```
mappings:
 innerTM:
  source:
      - [marks.csv]
  subject: ":{PERSON}"
  predicateobjects:
      - [:jumps, $(MARK)]
```

ID	DATE	MARK	PERSON
1	2022-03-21	4.80	Angelica
2	2022-03-19	4.85	Katerina

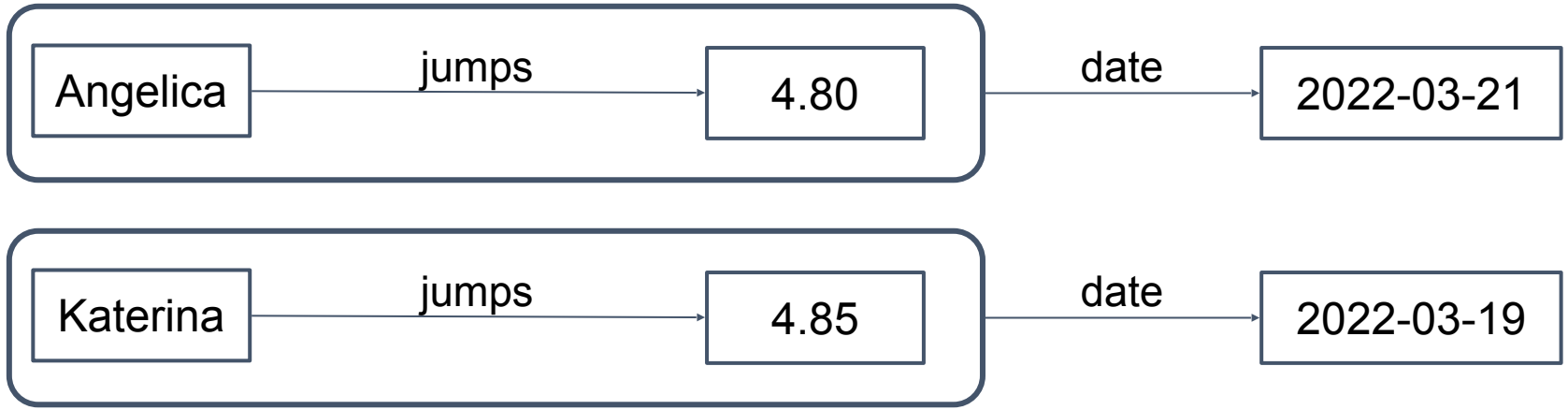
```
<#innerTM> a rr:TriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
   rr:template ":{PERSON}" ] ;
rr:predicateObjectMap [
   rr:predicate :jumps ;
   rml:objectMap [
      rml:reference
   "MARK" ] ] .
```


```
mappings:
 innerTM:
  source:
      - [marks.csv]
  subject: ":{PERSON}"
  predicateobjects:
      - [:jumps, $(MARK)]
```

```
:Angelica :jumps "4.80".
:Katerina :jumps "4.85"
```

How can we describe statements about statements in RDF?

ID	DATE	MARK	PERSON
1	2022-03-21	4.80	Angelica
2	2022-03-19	4.85	Katerina



Triples that include a **triple as a subject or an object** are known as RDF-star triples

An RDF-star graph is a **set of RDF-star triples**.

SPARQL-star extends SPARQL to query RDF-star graphs

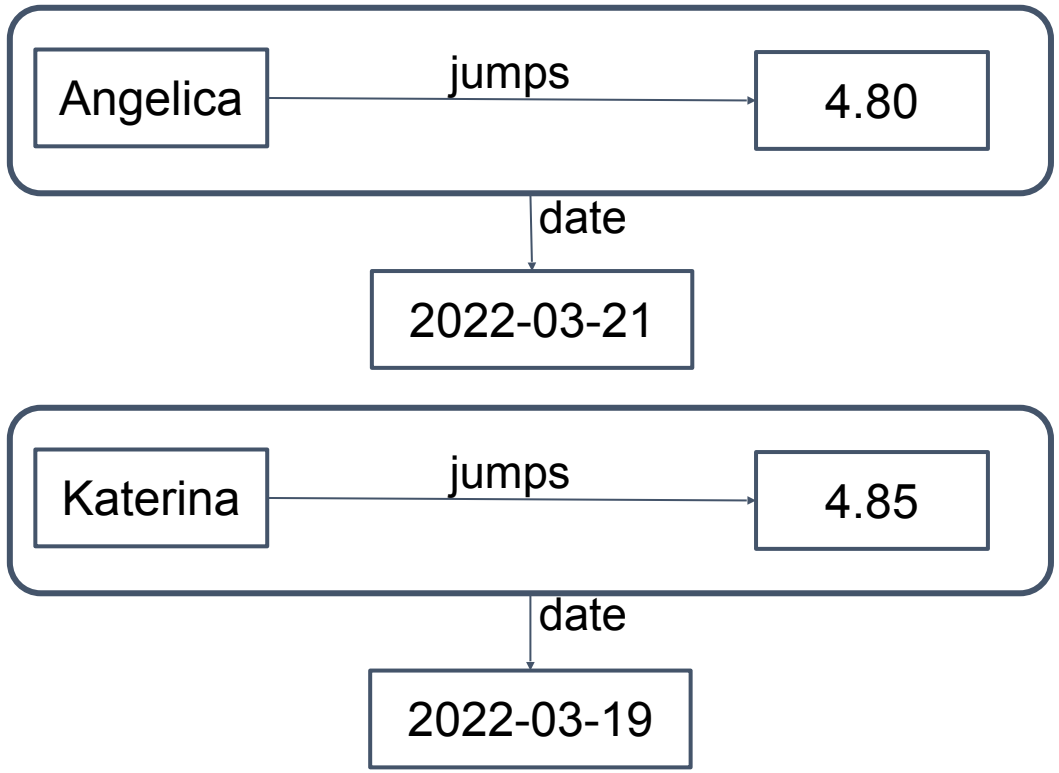
Towards RDF 1.2 with 16 specifications being reviewed

Triples that include a **triple as a subject or an object** are known as RDF-star triples

An RDF-star graph is a **set of RDF-star triples.**

SPARQL-star extends SPARQL to query RDF-star graphs

Towards RDF 1.2 with 16 specifications being reviewed



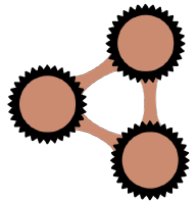

```
<< :Angelica :jumps "4.80" >> :date
    "2022-03-21" .
<< :Katerina :jumps "4.85" >> :date
    "2022-03-19" .


SELECT ?jumper ?mark ?date WHERE {
  << ?jumper :jumps ?mark >> :date ?date
}
```

Wide adoption of the approach from industry and vendors

Standardization process through the World Wide Web Consortium (W3C)

No sustainable procedure to generate RDF-star graphs






```
#row 1
<< :Angelica :jumps "4.80" >>
        :date "2022-03-21" .


#row 2
<< :Katerina :jumps "4.85" >>
        :date "2022-03-19" .
```

```
<#innerTM> a rml:NonAssertedTriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
    rr:template ":{PERSON}" ] ;
rr:predicateObjectMap [
    rr:predicate :jumps ;
    rml:objectMap [
        rml:reference "MARK" ] ] .

<#outerTM> a rr:TriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
    rml:quotedTriplesMap <#innerTM> ] ;
rr:predicateObjectMap [
    rr:predicate :date ;
    rml:objectMap [
        rml:reference "DATE" ] ] .
        c
```


Delva, T., Arenas-Guerrero, J., Iglesias-Molina, A., Corcho, O., Chaves-Fraga, D., & Dimou, A. (2021). RML-star: A declarative mapping language for RDF-star generation. In ISWC2021, the International Semantic Web Conference

Arenas-Guerrero, J., Iglesias-Molina, A., Garijo, D., Chaves-Fraga, D., Corcho, O., & Dimou, A. (2023). Declarative generation of RDF-star graphs from heterogeneous data. Under Review at Semantic Web Journal

```
#row 1
<< :Angelica :jumps "4.80" >>
        :date "2022-03-21" .


#row 2
<< :Katerina :jumps "4.85" >>
        :date "2022-03-19" .
```

```
<#innerTM> a rml:NonAssertedTriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
    rr:template ":{PERSON}" ] ;
rr:predicateObjectMap [
    rr:predicate :jumps ;
    rml:objectMap [
        rml:reference "MARK" ] ] .

<#outerTM> a rr:TriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
    rml:quotedTriplesMap <#innerTM> ] ;
rr:predicateObjectMap [
    rr:predicate :date ;
    rml:objectMap [
        rml:reference "DATE" ] ] .
        c
```


Delva, T., Arenas-Guerrero, J., Iglesias-Molina, A., Corcho, O., Chaves-Fraga, D., & Dimou, A. (2021). RML-star: A declarative mapping language for RDF-star generation. In ISWC2021, the International Semantic Web Conference

Arenas-Guerrero, J., Iglesias-Molina, A., Garijo, D., Chaves-Fraga, D., Corcho, O., & Dimou, A. (2023). Declarative generation of RDF-star graphs from heterogeneous data. Under Review at Semantic Web Journal

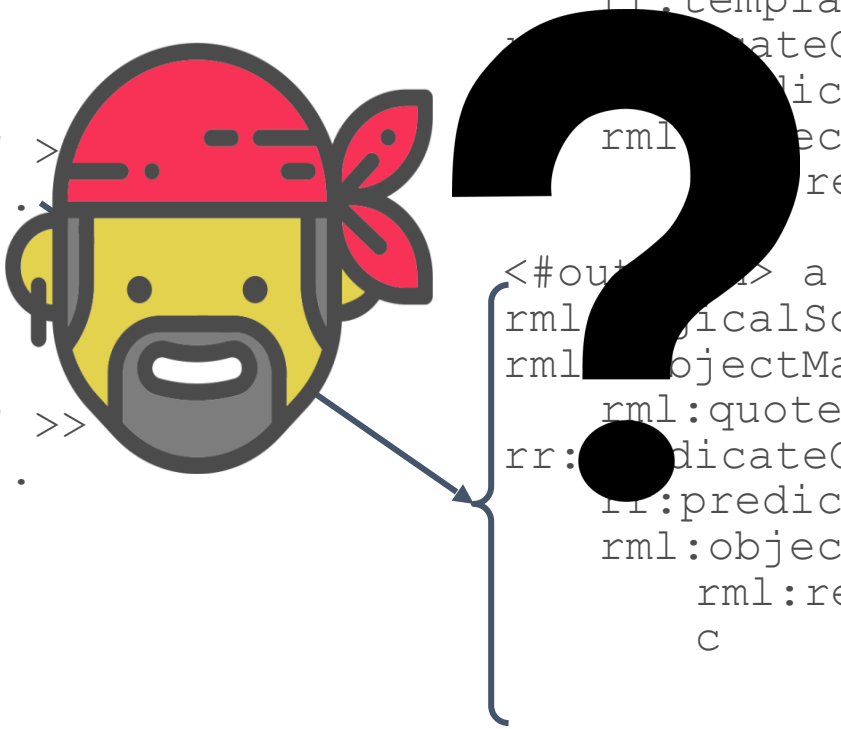
```
#row 1
<< :Angelica :jumps "4.80" >>
        :date "2022-03-21" .


#row 2
<< :Katerina :jumps "4.85" >>
        :date "2022-03-19" .
```

```
<#innerTM> a rml:NonAssertedTriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
    rr:template ":{PERSON}" ] ;
rr:predicateObjectMap [
    rr:predicate :jumps ;
    rml:objectMap [
        rml:reference "MARK" ] ] .

<#outerTM> a rr:TriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
    rml:quotedTriplesMap <#innerTM> ] ;
rr:predicateObjectMap [
    rr:predicate :date ;
    rml:objectMap [
        rml:reference "DATE" ] ] .
        c
```


Delva, T., Arenas-Guerrero, J., Iglesias-Molina, A., Corcho, O., Chaves-Fraga, D., & Dimou, A. (2021). RML-star: A declarative mapping language for RDF-star generation. In ISWC2021, the International Semantic Web Conference

Arenas-Guerrero, J., Iglesias-Molina, A., Garijo, D., Chaves-Fraga, D., Corcho, O., & Dimou, A. (2023). Declarative generation of RDF-star graphs from heterogeneous data. Under Review at Semantic Web Journal

```
#row 1
<< :Angelica :jumps "4.80" >>
        :date "2022-03-21" .


#row 2
<< :Katerina :jumps "4.85" >>
        :date "2022-03-19" .
```



```
<#innerTM> a rml:NonAssertedTriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
    rr:template ":{PERSON}" ] ;
    ateObjectMap [
    icate :jumps ;
    ectMap [
    reference "MARK" ] ] .

<#ou    > a rr:TriplesMap ;
rml    icalSource :marks ;
rml    bjectMap [
    rml:quotedTriplesMap <#innerTM> ] ;
rr:    dicateObjectMap [
    r:predicate :date ;
    rml:objectMap [
        rml:reference "DATE" ] ] .
        c
```


Delva, T., Arenas-Guerrero, J., Iglesias-Molina, A., Corcho, O., Chaves-Fraga, D., & Dimou, A. (2021). RML-star: A declarative mapping language for RDF-star generation. In ISWC2021, the International Semantic Web Conference

Arenas-Guerrero, J., Iglesias-Molina, A., Garijo, D., Chaves-Fraga, D., Corcho, O., & Dimou, A. (2023). Declarative generation of RDF-star graphs from heterogeneous data. Under Review at Semantic Web Journal

- Motivation
- Preliminaries: RML, YARRRML & RDF-star
- **YARRRML-star & YATTER**
- Validation & Comparison
- Conclusions and Future Work

RML-Star VS YARRRML-Star

```
<#innerTM> a rml:NonAssertedTriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
    rr:template ":{PERSON}" ] ;
rr:predicateObjectMap [
    rr:predicate :jumps ;
    rml:objectMap [
        rml:reference "MARK" ] ] .

<#outerTM> a rr:TriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
    rml:quotedTriplesMap <#innerTM> ] ;
rr:predicateObjectMap [
    rr:predicate :date ;
    rml:objectMap [
        rml:reference "DATE" ] ] .
```

```
<#innerTM> a rml:NonAssertedTriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
    rr:template ":{PERSON}" ] ;
rr:predicateObjectMap [
    rr:predicate :jumps ;
    rml:objectMap [
        rml:reference "MARK" ] ] .

<#outerTM> a rr:TriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
    rml:quotedTriplesMap <#innerTM> ] ;
rr:predicateObjectMap [
    rr:predicate :date ;
    rml:objectMap [
        rml:reference "DATE" ] ] .
```


```
mappings:
 innerTM:
  source:
       - [marks.csv]
  subject: ":{PERSON}"
  pom:
       - [:jumps, $(MARK)]

 outerTM:
  source:
       - [marks.csv]
  subject:
       quotedNonAsserted:
innerTM
  pom:
   - [:date, $(DATE)]
```

```
<#innerTM> a rml:NonAssertedTriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
    rr:template ":{PERSON}" ] ;
rr:predicateObjectMap [
    rr:predicate :jumps ;
    rml:objectMap [
        rml:reference "MARK" ] ] .

<#outerTM> a rr:TriplesMap ;
rml:logicalSource :marks ;
rml:subjectMap [
    rml:quotedTriplesMap <#innerTM> ] ;
rr:predicateObjectMap [
    rr:predicate :date ;
    rml:objectMap [
        rml:reference "DATE" ] ] .
```


```
mappings:
 innerTM:
  source:
       - [marks.csv]
  subject: ":{PERSON}"
  pom:
       - [:jumps, $(MARK)]

 outerTM:
  source:
       - [marks.csv]
  subject:
       quotedNonAsserted:
innerTM
  pom:
   - [:date, $(DATE)]
```

<< :Angelica :jumps "4.80" >> :date "2022-03-21" .
<< :Katerina :jumps "4.85" >> :date "2022-03-19" .

```
mappings:
 innerTM:
  source:
       - [marks.csv]
  subject: ":{PERSON}"
  pom:
       [:jumps, $(MARK)]

 outerTM:
  source:
       - [marks.csv]
  subject:
       quotedNonAsserted: innerTM
  pom:
   - [:date, $(DATE), xsd:$(DATATYPE)]
   - [:name, $(PERSON), $(COUNTRY)~lang]
   - predicates: :jumps
     objects:
      - function: join(mapping=innerTM, equal(child=$(ID), parent=$(ID))
```

Additional extensions:

- Dynamic datatypes
- Dynamic language
- Inline joins (reducing verbosity)

```
mappings:
 innerTM:
  source:
       - [marks.csv]
  subject: ":{PERSON}"
  pom:
       [:jumps, $(MARK)]

 outerTM:
  source:
       - [marks.csv]
  subject:
       quotedNonAsserted: innerTM
  pom:
   - [:date, $(DATE), xsd:$(DATATYPE)]
   - [:name, $(PERSON), $(COUNTRY)~lang]
   - predicates: :jumps
     objects:
      - function: join(mapping=innerTM, equal(child=$(ID), parent=$(ID))
```

Additional extensions:

- Dynamic datatypes
- Dynamic language
- Inline joins (reducing verbosity)

Built-in function implemented by all RML engines

Unofficial Draft

12. RDF-Star

An YARRRML-star mapping defines a mapping from any data in a structured source format to RDF-star. It enables the generation of quoted triples in the position of subject and/or object. Quoted triples can be created by referencing the mapping identifier using the `quoted` keyword. It is also possible to create non-asserted quoted triples using the `quotedNonAsserted` keyword.

12.1 Keys

quoted
Quoted mapping.
Required profiles: RMLStar.
Datatype: Mapping reference.

quotedNonAsserted
Non-asserted quoted mapping.
Required profiles: RMLStar.
Datatype: Mapping reference.

12.2 Examples

YARRRML RML

EXAMPLE 110: Quoted mapping in object

```
mappings:
  person:
    subject: http://example.org/person/$(Name)
    predicateobjects:
     - predicates: ex:confirms
       objects:
        - quoted: student
  student:
    subject: http://example.org/student/$(Student)
    predicateobjects:
     - [a, ex:Student]
```

YATTER: Your open source engine for YARRRML-star

Easy to read outputs (following Turtle RDF syntax with BN)

Translation from YARRRML-star to R2RML/RML/RML-star and viceversa

Continuous integration/development of test-cases

Code coverage of ~85% of with the test-cases

PyPi module (easy to install)

Follows Open Science good practices (GitHub + Zenodo)

YATTER

```
[ ]  1 !pip install yatter
```

```
1 import yatter
2 import yaml
3
4 # Translate the input mapping into [R2]RML
5 rml_content = yatter.translate(yaml.safe_load(open("my_mapping.yml")))
6 # Run your favourite RML Engine with the output
```

- Motivation
- Preliminaries: RML, YARRRML & RDF-star
- YARRRML-star & YATTER
- **Validation & Comparison**
- Conclusions and Future Work

Comparison w.r.t. other user-friendly **serializations**

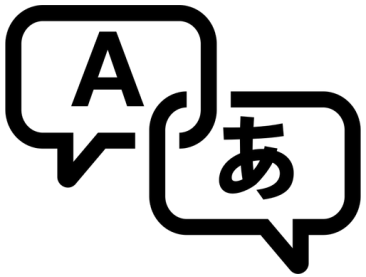
ShExML Vs SMS2 Vs XRM

Comparison w.r.t. other user-friendly **serializations**

ShExML Vs SMS2 Vs XRM

Comparison w.r.t. other **RML translators** (XX2RML)

ShExML translator Vs Stardog Vs
XRM translator Vs YARRRML-parser

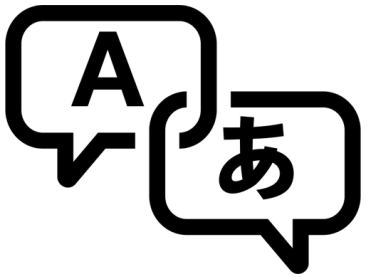


Comparison w.r.t. other user-friendly **serializations**

ShExML Vs SMS2 Vs XRM

Comparison w.r.t. other **RML translators** (XX2RML)

ShExML translator Vs Stardog Vs
XRM translator Vs YARRRML-parser



Validation through **test-cases**

Manually development of >50 test-cases for YARRRML

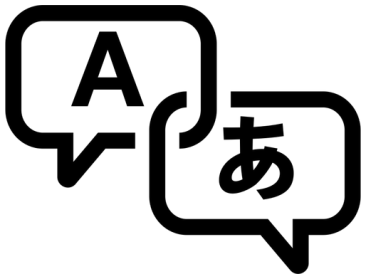
YARRRML[-star] ↔ [R2]RML-[star]

Comparison w.r.t. other user-friendly **serializations**

ShExML Vs SMS2 Vs XRM

Comparison w.r.t. other **RML translators** (XX2RML)

ShExML translator Vs Stardog Vs
XRM translator Vs YARRRML-parser



Validation through **test-cases**

Manually development of >50 test-cases for YARRRML
YARRRML[-star] ↔ [R2]RML-[star]

Two real use cases where YARRRML-star + YATTER are used

Feature/Serialization	ShExML	SMS2	XRM	YARRRML-star
Subject Term	BN, IRI	BN, IRI, *T	IRI	BN, IRI, *T
Subject Generation	C, D (1..1)	C, D, (1..1)	C, D, (1..1)	C, D (0..1)
Predicate Term	BN, IRI	IRI	IRI	IRI
Predicate Generation	C (1..1)	C, D (1..1)	C (1..1)	C, D (1..N)
Object Term	IRI, L	BN, IRI, L, *T	IRI, L	BN, IRI, L, *T
Object Generation	C, D (1..1)	C, D (1..N)	C, D (1..1)	C, D (1..N)
Datatypes	C, D (0..1)	C (0..1)	C (0..1)	C, D (0..1)
Language tag	C, D (0..1)	C, D (0..1)	C (0..1)	C, D (0..1)
Named graphs	C (0..1)	C (1..1)	C, D (0..N)	C, D (0..N)
Data source description	Input	Input	Input	Input, output
Data source linking	Yes	No	No	Yes
Nested hierarchies	Yes	No	No	No
Functions	Yes	Yes	No	Yes
Conditions	Yes	No	No	Yes

Feature/Serialization	ShExML	SMS2	XRM	YARRRML-star
Subject Term	BN, IRI	BN, IRI, *T	IRI	BN, IRI, *T
Subject Generation	C, D (1..1)	C, D, (1..1)	C, D, (1..1)	C, D (0..1)
Predicate Term	BN, IRI	IRI	IRI	IRI
Predicate Generation	C (1..1)	C, D (1..1)	C (1..1)	C, D (1..N)
Object Term	IRI, L	BN, IRI, L, *T	IRI, L	BN, IRI, L, *T
Object Generation	C, D (1..1)	C, D (1..N)	C, D (1..1)	C, D (1..N)
Datatypes	C, D (0..1)	C (0..1)	C (0..1)	C, D (0..1)
Language tag	C, D (0..1)	C, D (0..1)	C (0..1)	C, D (0..1)
Named graphs	C (0..1)	C (1..1)	C, D (0..N)	C, D (0..N)
Data source description	Input	Input	Input	Input, output
Data source linking	Yes	No	No	Yes
Nested hierarchies	Yes	No	No	No
Functions	Yes	Yes	No	Yes
Conditions	Yes	No	No	Yes

Feature/Serialization	ShExML	SMS2	XRM	YARRRML-star
Subject Term	BN, IRI	BN, IRI, *T	IRI	BN, IRI, *T
Subject Generation	C, D (1..1)	C, D, (1..1)	C, D, (1..1)	C, D (0..1)
Predicate Term	BN, IRI	IRI	IRI	IRI
Predicate Generation	C (1..1)	C, D (1..1)	C (1..1)	C, D (1..N)
Object Term	IRI, L	BN, IRI, L, *T	IRI, L	BN, IRI, L, *T
Object Generation	C, D (1..1)	C, D (1..N)	C, D (1..1)	C, D (1..N)
Datatypes	C, D (0..1)	C (0..1)	C (0..1)	C, D (0..1)
Language tag	C, D (0..1)	C, D (0..1)	C (0..1)	C, D (0..1)
Named graphs	C (0..1)	C (1..1)	C, D (0..N)	C, D (0..N)
Data source description	Input	Input	Input	Input, output
Data source linking	Yes	No	No	Yes
Nested hierarchies	Yes	No	No	No
Functions	Yes	Yes	No	Yes
Conditions	Yes	No	No	Yes

Feature/Serialization	ShExML	SMS2	XRM	YARRRML-star
Subject Term	BN, IRI	BN, IRI, *T	IRI	BN, IRI, *T
Subject Generation	C, D (1..1)	C, D, (1..1)	C, D, (1..1)	C, D (0..1)
Predicate Term	BN, IRI	IRI	IRI	IRI
Predicate Generation	C (1..1)	C, D (1..1)	C (1..1)	C, D (1..N)
Object Term	IRI, L	BN, IRI, L, *T	IRI, L	BN, IRI, L, *T
Object Generation	C, D (1..1)	C, D (1..N)	C, D (1..1)	C, D (1..N)
Datatypes	C, D (0..1)	C (0..1)	C (0..1)	C, D (0..1)
Language tag	C, D (0..1)	C, D (0..1)	C (0..1)	C, D (0..1)
Named graphs	C (0..1)	C (1..1)	C, D (0..N)	C, D (0..N)
Data source description	Input	Input	Input	Input, output
Data source linking	Yes	No	No	Yes
Nested hierarchies	Yes	No	No	No
Functions	Yes	Yes	No	Yes
Conditions	Yes	No	No	Yes

Feature/Serialization	ShExML	SMS2	XRM	YARRRML-star
Subject Term	BN, IRI	BN, IRI, *T	IRI	BN, IRI, *T
Subject Generation	C, D (1..1)	C, D, (1..1)	C, D, (1..1)	C, D (0..1)
Predicate Term	BN, IRI	IRI	IRI	IRI
Predicate Generation	C (1..1)	C, D (1..1)	C (1..1)	C, D (1..N)
Object Term	IRI, L	BN, IRI, L, *T	IRI, L	BN, IRI, L, *T
Object Generation	C, D (1..1)	C, D (1..N)	C, D (1..1)	C, D (1..N)
Datatypes	C, D (0..1)	C (0..1)	C (0..1)	C, D (0..1)
Language tag	C, D (0..1)	C, D (0..1)	C (0..1)	C, D (0..1)
Named graphs	C (0..1)	C (1..1)	C, D (0..N)	C, D (0..N)
Data source description	Input	Input	Input	Input, output
Data source linking	Yes	No	No	Yes
Nested hierarchies	Yes	No	No	No
Functions	Yes	Yes	No	Yes
Conditions	Yes	No	No	Yes

Feature/Serialization	ShExML	SMS2	XRM	YARRRML-star
Subject Term	BN, IRI	BN, IRI, *T	IRI	BN, IRI, *T
Subject Generation			1)	C, D (0..1)
Predicate Term				IRI
Predicate Generation				C, D (1..N)
Object Term				BN, IRI, L, *T
Object Generation				C, D (1..N)
Datatypes				C, D (0..1)
Language tag				C, D (0..1)
Named graphs				C, D (0..N)
Data source description				Input, output
Data source linking				Yes
Nested hierarchies	Yes	No	No	No
Functions	Yes	Yes	No	Yes
Conditions	Yes	No	No	Yes

YARRRML-star is able to describe

1) **recursive triples** in subject and object
2) **dynamic** language tags and datatypes
3) input sources but also **output**
4) **functions and conditions**

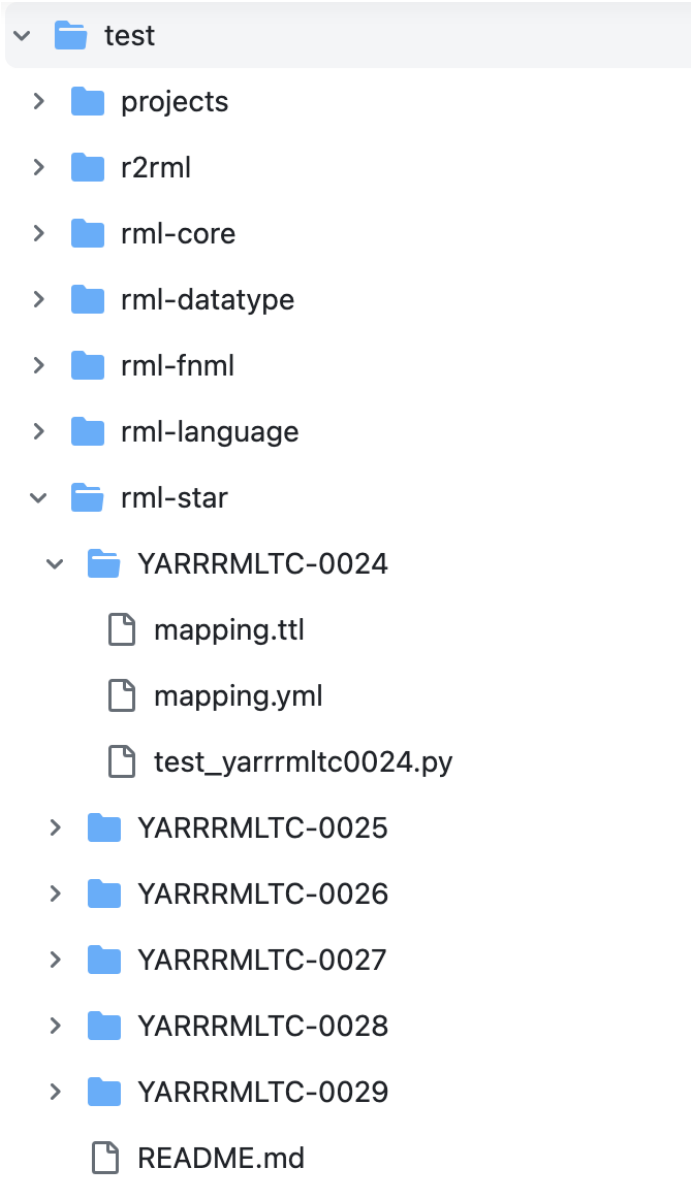
	YARRRML-parser	ShExML	XRM	Stardog	YATTER
Availability	Open source	Open source	Commercial	Commercial	Open source
Programming language	Javascript	Java	Java	Java	Python
Input Data sources	RDB, NoSQL DB, tabular, hierarchical	RDB, tabular, hierarchical, RDF	RDB, tabular, Hierarchical (XML)	RDB, tabular, hierarchical (JSON), GraphQL	RDB, NoSQL DB, tabular, hierarchical
Input Serialization	YARRRML, RML, R2RML	ShExML syntax	XRM syntax	R2RML, SMS, SMS2	YARRRML, RML-star,R2RML
Output Serialization	RML, R2RML, YARRRML	RML, RDF (KG)	RML, R2RML, CARML, CSVW	RDF (KG)	YARRRML, RML-star,R2RML
RDF-star	No	No	No	Yes	Yes
Dyn. Datatype	No	Yes	No	No	Yes
Dyn. Language	No	Yes	No	No	Yes

	YARRRML-parser	ShExML	XRM	Stardog	YATTER
Availability	Open source	Open source	Commercial	Commercial	Open source
Programming language	Javascript	Java	Java	Java	Python
Input Data sources	RDB, NoSQL DB, tabular, hierarchical	RDB, tabular, hierarchical, RDF	RDB, tabular, Hierarchical (XML)	RDB, tabular, hierarchical (JSON), GraphQL	RDB, NoSQL DB, tabular, hierarchical
Input Serialization	YARRRML, RML, R2RML	ShExML syntax	XRM syntax	R2RML, SMS, SMS2	YARRRML, RML-star,R2RML
Output Serialization	RML, R2RML, YARRRML	RML, RDF (KG)	RML, R2RML, CARML, CSVW	RDF (KG)	YARRRML, RML-star,R2RML
RDF-star	No	No	No	Yes	Yes
Dyn. Datatype	No	Yes	No	No	Yes
Dyn. Language	No	Yes	No	No	Yes

	YARRRML-parser	ShExML	XRM	Stardog	YATTER
Availability	Open source	Open source	Commercial	Commercial	Open source
Programming language	Javascript	Java	Java	Java	Python
Input Data sources	RDB, NoSQL DB, tabular, hierarchical	RDB, tabular, hierarchical, RDF	RDB, tabular, Hierarchical (XML)	RDB, tabular, hierarchical (JSON), GraphQL	RDB, NoSQL DB, tabular, hierarchical
Input Serialization	YARRRML, RML, R2RML	ShExML syntax	XRM syntax	R2RML, SMS, SMS2	YARRRML, RML-star,R2RML
Output Serialization	RML, R2RML, YARRRML	RML, RDF (KG)	RML, R2RML, CARML, CSVW	RDF (KG)	YARRRML, RML-star,R2RML
RDF-star	No	No	No	Yes	Yes
Dyn. Datatype	No	Yes	No	No	Yes
Dyn. Language	No	Yes	No	No	Yes

More than 50 test-cases for YARRRML-star

- Extend R2RML and RML official test-cases
- assess the conformance of any YARRRML engine
- Cover all current specs:
  - R2RML, RML-core, RML-functions, RML-star, RML-language, RML-datatype and RML-IO
- Cover language keys/shortcuts/etc
- Test correctness and prettiness generated output



Features	YARRRML parser	ShExML	XRM Translator	Stardog	YATTER
[R2]RML-core	18/23	18/18	11/11	13/13	23/23
RML-star	N/A	N/A	N/A	2/2	6/6
RML-language	3/3	3/3	1/1	3/3	3/3
RML-datatype	N/A	0/2	N/A	N/A	2/2
RML-IO	6/6	N/A	N/A	N/A	6/6
RML-functions	0/10	0/4	N/A	6/6	10/10
Total w.r.t. its serialization	64% (27/42)	77% (21/27)	100% (12/12)	100% (24/24)	100% (50/50)
Total w.r.t. all features	54% (27/50)	42% (21/50)	24% (12/50)	48% (24/50)	100% (50/50)

YATTER passes all test-cases

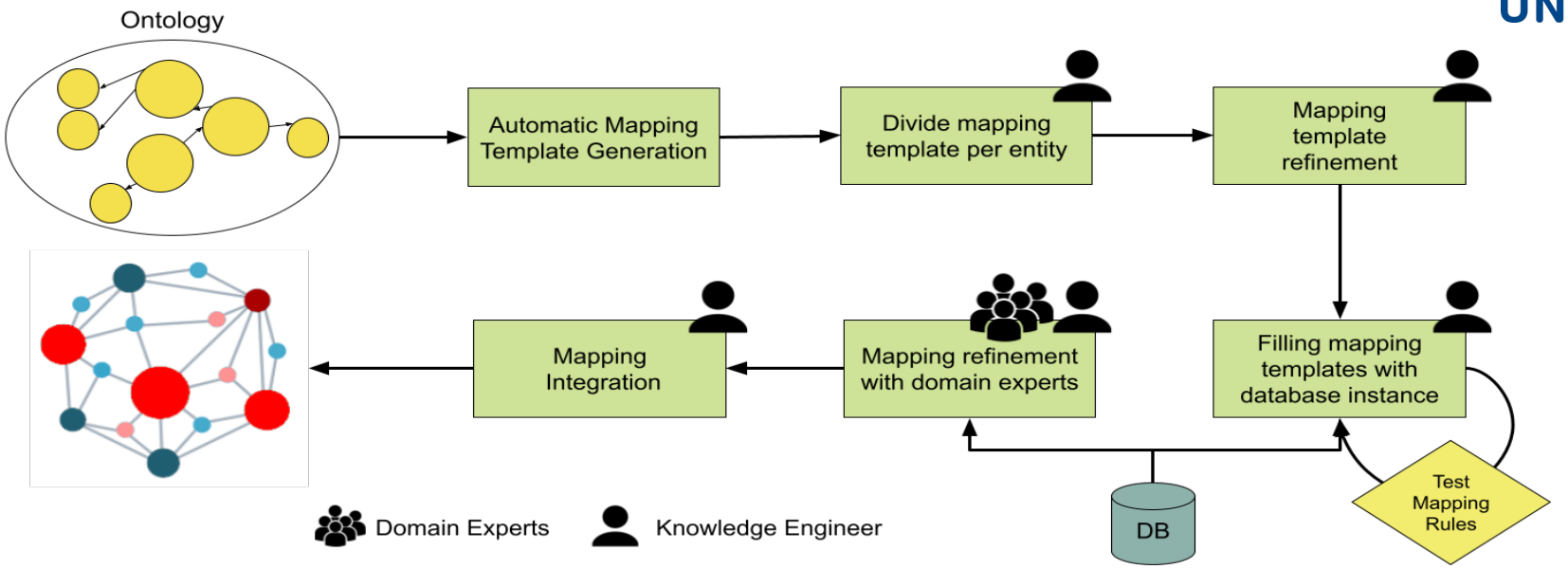
Stardog and XRM pass all supported features by their serialization

YARRRML-parser and ShExML provide >60% of the language coverage

The EU Data Space on Public Procurement

- Homogenize the access to all public procurement data across Europe
- First semantic data architecture of this magnitude in the world
- Calculate standard transparency indicators for each member state
- Proof of concept with YARRRML-Star, YATTER and RMLMapper

Constructing RDF graphs for Research-Performing Organizations



Conclusions:

- Extending YARRRML for a new generation of KG Construction systems
- Supporting Knowledge Engineers with a new translator
- Ensuring the sustainability of YARRRML with test-cases
- Comparing YARRRML+YATTER w.r.t. other similar solutions

Future Work:

- Support the transition from “old” (YARR)RML to the “new” RML
- User evaluation with human-friendly serializations
- Fully coverage of the YARRRML specification with test-cases

Human-Friendly RDF Graph Construction: Which one do you chose?

Ana Iglesias-Molina[1], **David Chaves-Fraga[1,2,3],**
Ioannis Dasoulas[2,3,4], Anastasia Dimou[2,3,4]

[1]Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
[2]Declarative Languages and Artificial Intelligence, KULeuven, Belgium
[3]Flanders Make, Belgium
[4]Leuven.AI, Belgium

david.chaves@upm.es dchavesf

07/06/2023 ICWE