TOOLS FOR MIXED REALITY IN SMART ENVIRONMENTS

LAB 4: Decision-making

November 12, 2020

Ph.D. Esteban Guerrero
esteban@cs.umu.se

UMEÅ UNIVERSITY

AGENDA

- Decision-making with SWRL
- Smart environments key aspects for engineering
- Built-in platforms for BDI and simulation agents engineering

BEFORE START…

- Any question about topics of the previous lecture/lab?

  Go to **www.menti.com** and use the code **39 57 81 9**

MICRO SURVEY

Question 1:

- Can you write two (or more) key words that were interesting from previous lecture/laboratory?

Go to **www.menti.com** and use the code **39 57 81 9**

MICRO SURVEY

Question 2:

- What do you consider is missing from previous lecture/laboratory?

Go to **www.menti.com** and use the code **39 57 81 9**

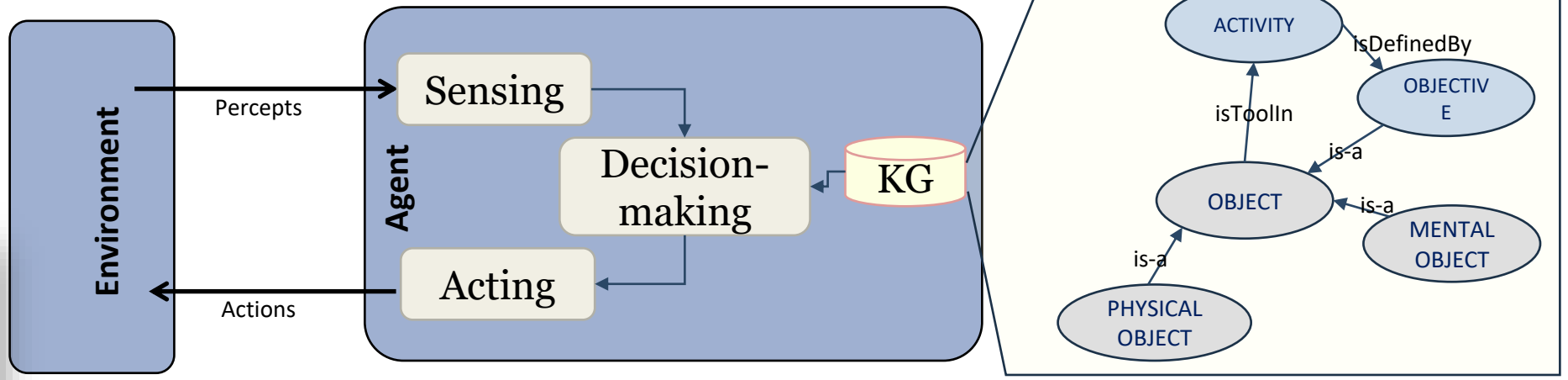
DECISION-MAKING WITH SWRL IN DIFFERENT LANGUAGES

UMEÅ UNIVERSITY

THE GOAL

Given a sensor-based input, how a decision is produced?

Input



SWRL WITH DIFFERENT LANGUAGES

UMEÅ UNIVERSITY

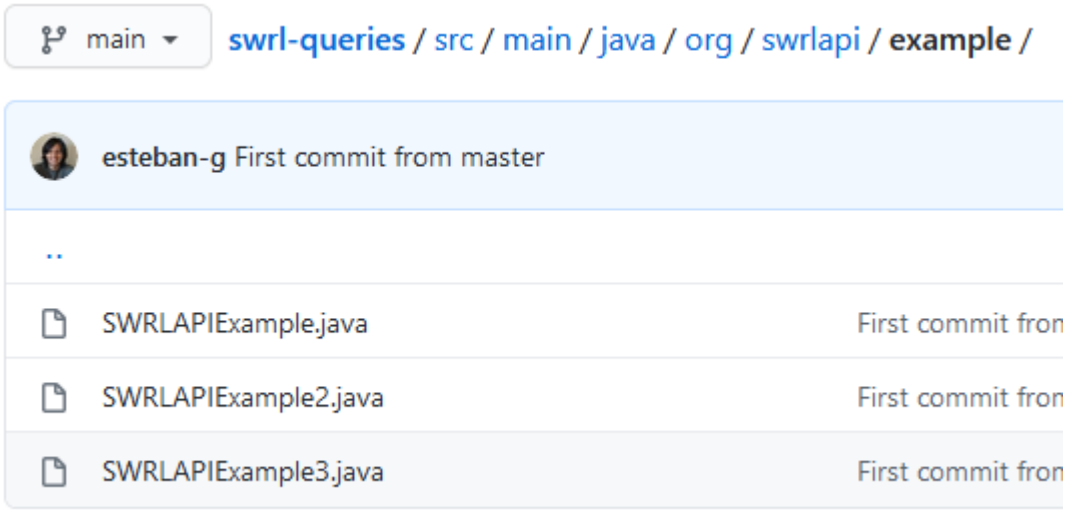
SWRL IN JAVA

SWRLAPI

- https://github.com/protegeproject/swrlapi

Demo: https://github.com/esteban-g/swrl-queries

```
// Create SQWRL query engine using the SWRLAPI
SQWRLQueryEngine queryEngine2 = SWRLAPIFactory.createSQWRLQueryEngine(peopleLOntology);

SQWRLResult result2 = queryEngine2.runSQWRLQuery("q2", "people:adult(?i) -> sqwrl:select(?i)

// Process the SQWRL result
if (result2.next()) {
    System.out.println("query result 1: ");
    System.out.println("\t data 1:" + result2.getNamedIndividual(0));
}
```



UMEÅ UNIVER

SWRL IN PYTHON

Owlready2

- https://pypi.org/project/Owlready2/

Demo: drug.py

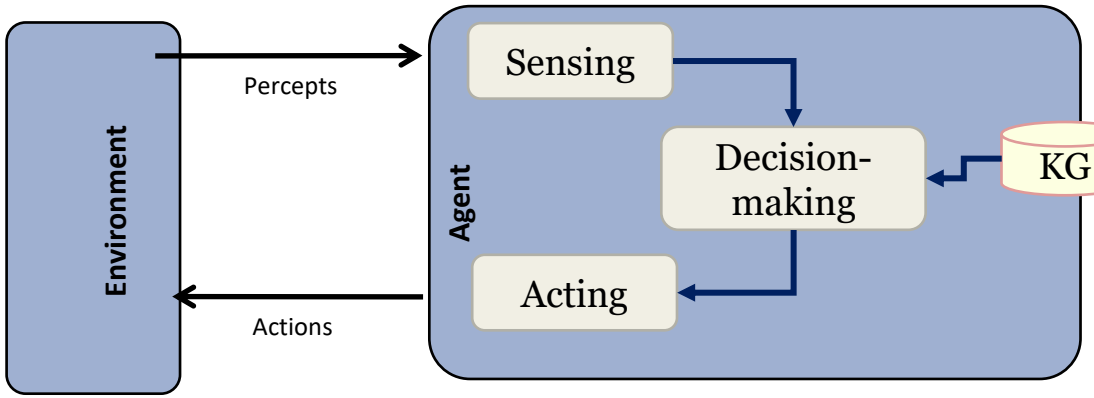
```
from owlready2 import *


# Create the ontology from scratch

onto = get_ontology("http://test.org/onto.owl")

with onto:
    class Drug(Thing):
        def take(self): print("I took a drug")

    class ActivePrinciple(Thing):
        pass

    class has_for_active_principle(Drug >> ActivePrinciple):
        python_name = "active_principles"

rule = Imp()
rule.set_as_rule(
    """Drug(?d), price(?d, ?p), number_of_tablets(?d, ?n), divide(?r, ?p, ?n) -> price_per_tablet(?d, ?r)""")

sync_reasoner_pellet(infer_property_values=True,
                     infer_data_property_values=True)

print("-->drug0 drug price per tablet:", drug0.price_per_tablet)
```

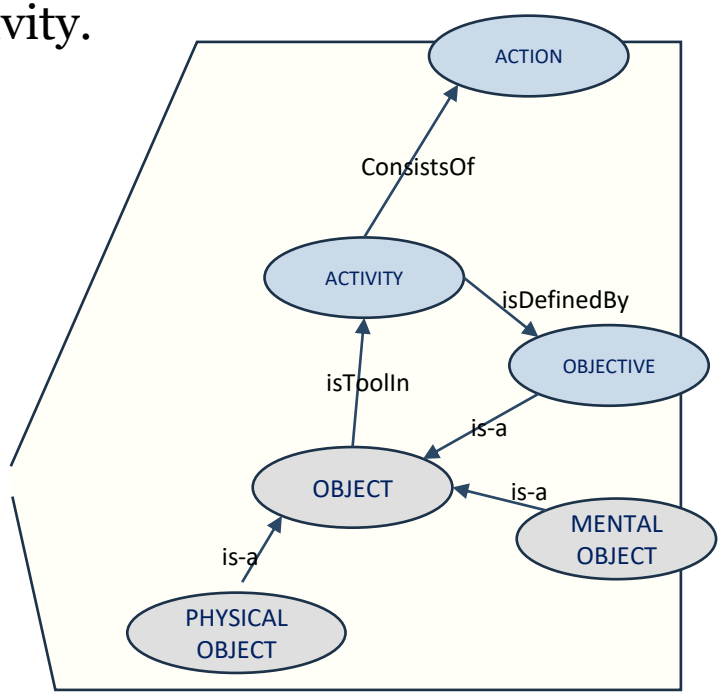
SMART ENVIRONMENTS
ENGINEERING PERSPECTIVE

UMEÅ UNIVERSITY

KEY ASPECTS TO CONSIDER DURING THE DESIGN

1. The agent's knowledge (agent=smart environment).
   - type of information: context, user profile, etc.
   - status of information: incomplete, inconsistent, etc.
2. The data input.
   - available data: sensor-based, manual/user input, etc.
   - status of data: incomplete, uncertain, etc.
3. The output.
   - feedback concurrency: during or after a user performs an activity.
   - feedback modality: audio, visual, haptic, etc.
4. The decision-making algorithm
   - Computational and time consumption

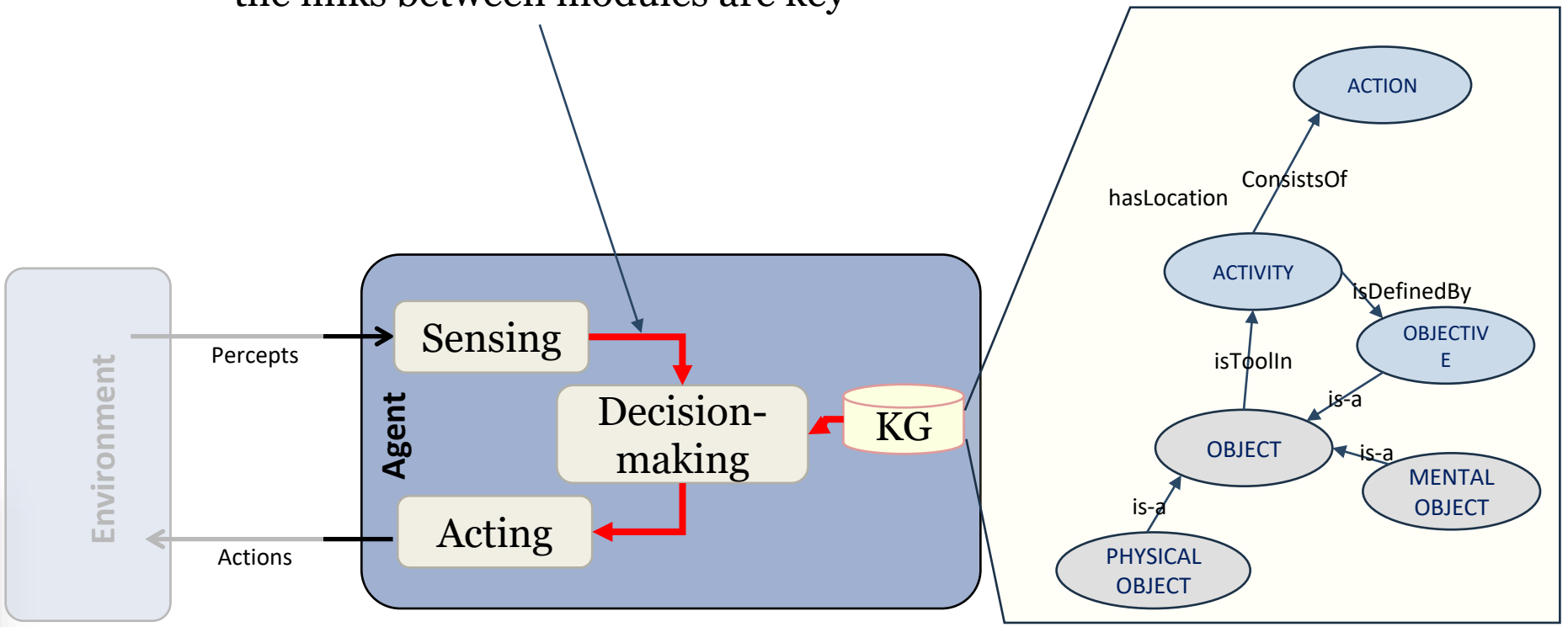




KEY ASPECTS TO CONSIDER DURING THE DEVELOPMENT

From a software engineering perspective,
the links between modules are key

Input



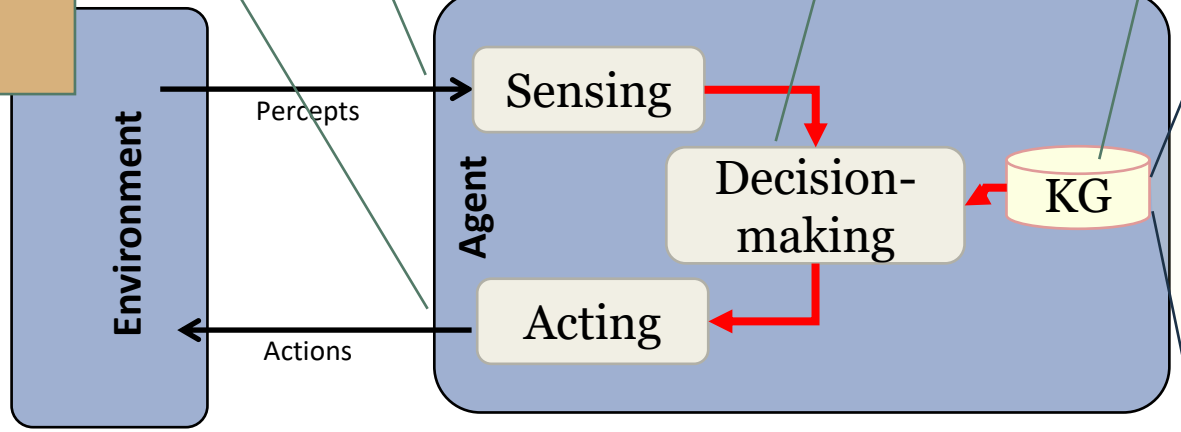
EXAMPLE CROSS COUNTRY SKIING

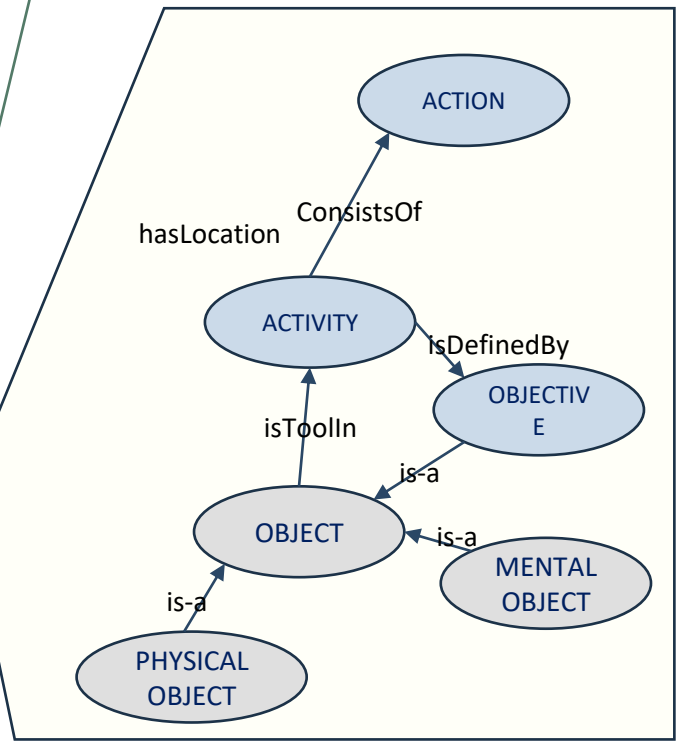
INPUT
- Limited video
- Human activity skeleton vectors
- Supervised ML: AdaBoost + random forest

OUTPUT
- feedback during skiing
- feedback modality: audio and visual
- Projected AR
- C#+Python

DECISION MAKING
- Answer-set programming
- DLV system for reasoning

KNOWLEDGE
- X-country skiing ontology
- Coaches (humans) made rules
- OWL/RDF





OTHER ALTERNATIVES FOR BDI AGENTS

UMEÅ UNIVERSITY

BDI PLATFORMS

1. JaCaMo
   URL: http://jacamo.sourceforge.net
   Language: AgentSpeak + Java

*bob.asl

```
1
2  /* Initial goals */
3  !start.
4  !send(0).
5
6  /* Plans */
7  +!start <- .wait(1000); message("hello world."); !start.
8  +!send(X)  <- .wait(1500); .send(alice, tell, hello(X)); !send(X+1).
9
10
11
12
13 { include("$jacamoJar/templates/common-cartago.asl") }
14 { include("$jacamoJar/templates/common-moise.asl") }
15 { include("$jacamoJar/templates/org-obedient.asl") }
16
```

2. GAMA platform
   URL: https://gama-platform.github.io
   Language: GAML language

Demo: GAMA firefighter agents

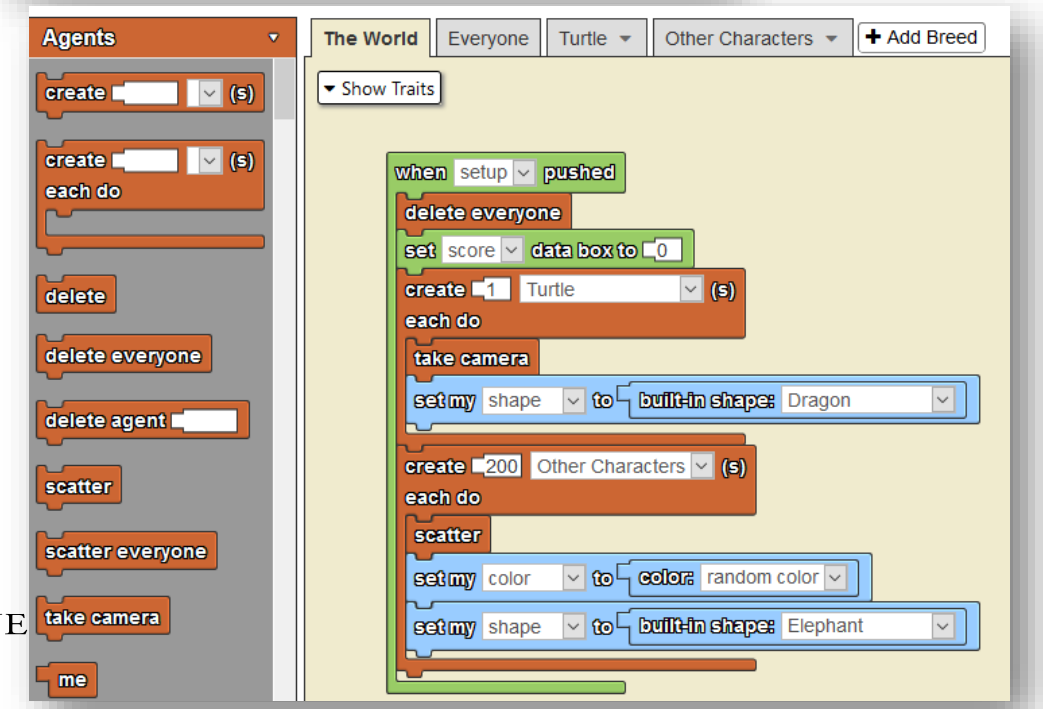
```
74     //The rules are used to create a desire from a belief. We can specify the priority of the desire with a statement
75     rule belief: new_predicate(fireLocation) new_desire: get_predicate(get_belief_with_name(fireLocation));
76     rule belief: no_water_predicate new_desire: water_predicate strength: 10.0;
77
78     //The plan to do when the intention is to patrol.
79     plan patrolling intention:patrol_desire{
80         do wander amplitude: 30.0 speed: 2.0;
81     }
82
83     //The plan that is executed when the agent got the intention of extinguish a fire.
84     plan stopFire intention: new_predicate(fireLocation) priority:5{
85         point target_fire <- point(get_predicate(get_current_intention()).values["location_value"] );
86         if(waterValue>0){
87             if (self distance_to target_fire <= 1) {
88                 fireArea current_fire <- fireArea first_with (each.location = target_fire);
89                 if (current_fire != nil) {
90                     waterValue <- waterValue - 1.0;
91                     current_fire.size <-  current_fire.size - 1;
92                     if ( current_fire.size <= 0) {
93                        ask  current_fire {do die;}
94                        do remove_belief(get_predicate(get_current_intention()));
95                        do remove_intention(get_predicate(get_current_intention()), true);
```

BDI-SOCIAL SIMULATION PLATFORMS

1. Netlogo
   URL: https://www.netlogoweb.org/
   Language: Netlogo

```
globals [
  sample-car
]

turtles-own [
  speed
  speed-limit
  speed-min
]

to setup
  clear-all
  ask patches [ setup-road ]
  setup-cars
  watch sample-car
  reset-ticks
end
```

2. StarLogo Nova
   URL: https://www.slnova.org/
   Language: Netlogo + graphical



https://www.slnova.org/esteban_g/projects/772397

Demo: Dragon eating elephants

UMEÅ UNIVE

OTHER RESOURCES

UMEÅ UNIVERSITY

OWLREADY2 DOCS

- https://owlready2.readthedocs.io/en/latest/rule.html

```
>>> from owlready2 import *
>>> onto_path.append("/path/to/your/local/ontology/repository")
>>> onto = get_ontology("http://www.lesfleursdunormal.fr/static/_downloads/pizza_onto.owl")
>>> onto.load()

>>> class NonVegetarianPizza(onto.Pizza):
...   equivalent_to = [
...     onto.Pizza
...   & ( onto.has_topping.some(onto.MeatTopping)
...     | onto.has_topping.some(onto.FishTopping)
...     ) ]

...   def eat(self): print("Beurk! I'm vegetarian!")
```

RDFLIB JS

Javascript library for working with RDF files, e.g. orntologies

- https://github.com/linkeddata/rdflib.js
- https://linkeddata.github.io/rdflib.js/Documentation/webapp-intro.html


THANK YOU