BIOPAX EXPLORER

- Manipuler des réseaux biologiques interopérables en Python en se basant sur le modèle de données.

F. Moreews
PEGASE - DYLISS

RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

INRAE

BIOPAX EXPLORER

- Manipuler des réseaux biologiques interopérables en Python en se basant sur le modèle de données BIOPAX.

Collaboration UMR PEGASE et UMR IRISA

PEGASE : F.Moreews, F. Gondret, C. Juigné (INRAE)
IRISA : E. Becker, O. Dameron, (Univ. Rennes)

RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

INRAE

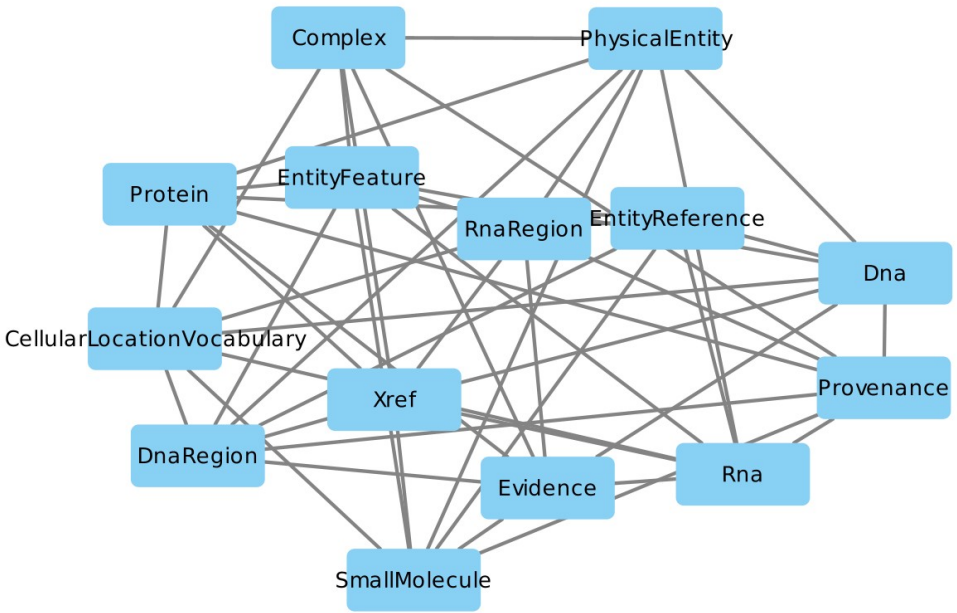
BIOPAX EXPLORER

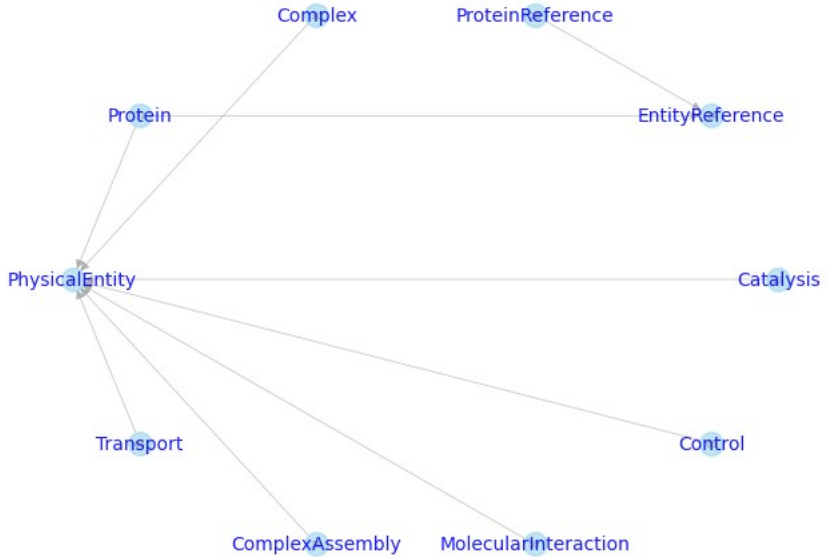
BIOPAX : modèle défini en RDF via OWL

```
<owl:Ontology rdf:about="">
  <rdfs:comment rdf:datatype="&xsd;string" >This is version 1.0 of the BioPAX Level 3 ontology.
  The goal of the BioPAX group is to develop a common exchange format for biological pathway data.
  More information is available at http://www.biopax.org.
  This ontology is freely available under the LGPL (http://www.gnu.org/copyleft/lesser.html).
  </rdfs:comment>
</owl:Ontology>
```

BIOPAX EXPLORER

- BIOPAX : modele des interactions biologiques, genes, metabolisme, p/p. Plus de 40 entités interconnectés



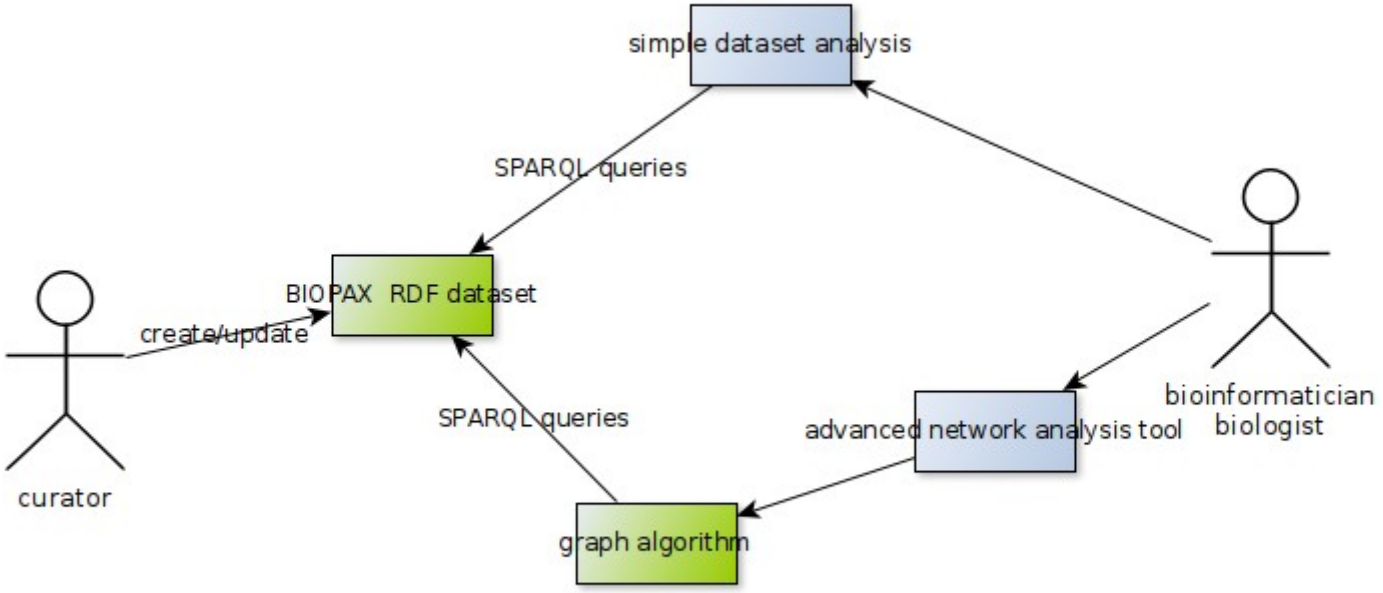


RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

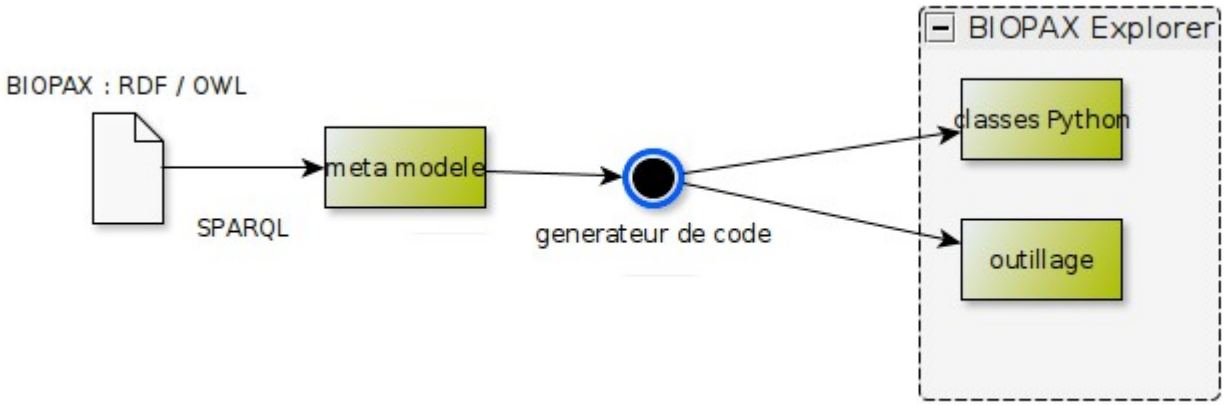
INRAE

BIOPAX EXPLORER

Cas d'utilisation

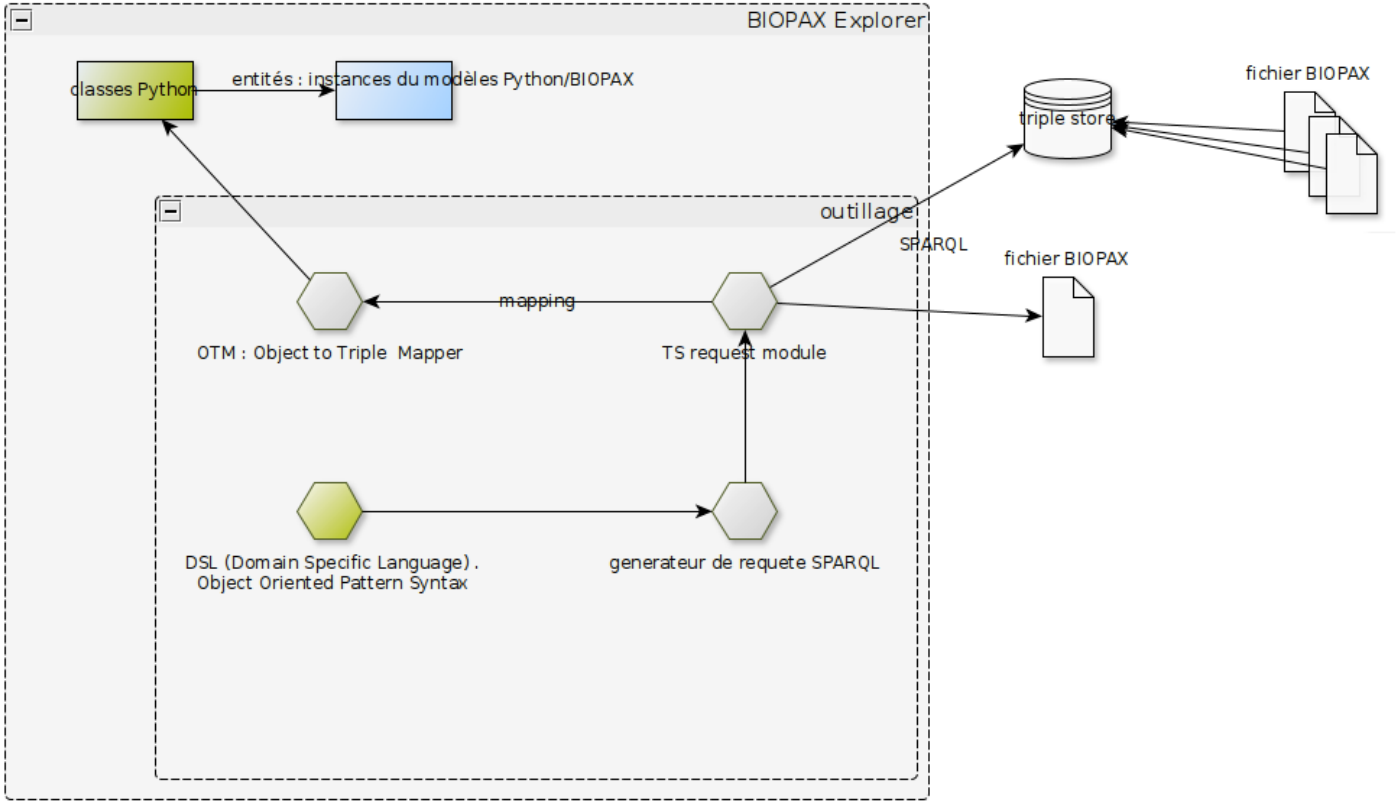


BIOPAX EXPLORER



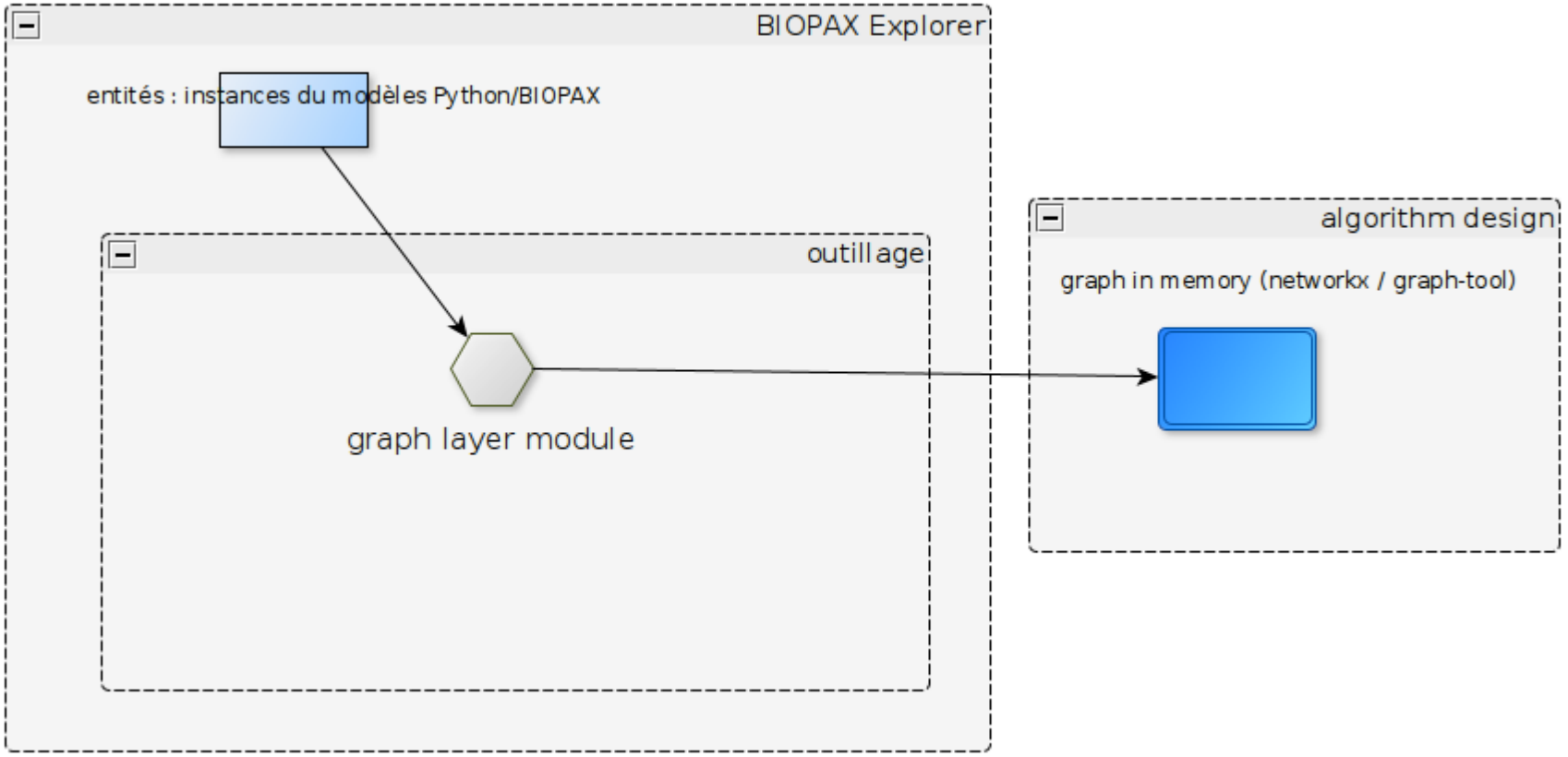
Approche orienté modèle : garantissant la compatibilité entre specification OWL et implémentation
pour un modèle complexe

BIOPAX EXPLORER



Pattern => sparql => liste d’entités interconnectées

BIOPAX EXPLORER



entités => graphe en mémoire=>(transformation)=>algorithme

BIOPAX EXPLORER

Biopax Explorer Rack : une collection de motifs de reference (inspirer de PAXTOOLS pattern /Java)

controlsMetabolicCatalysis :

Pattern for a Protein controlling a reaction whose participant is a small molecule.

The controller is in a Complex

notBlackboxComplexInComplex :

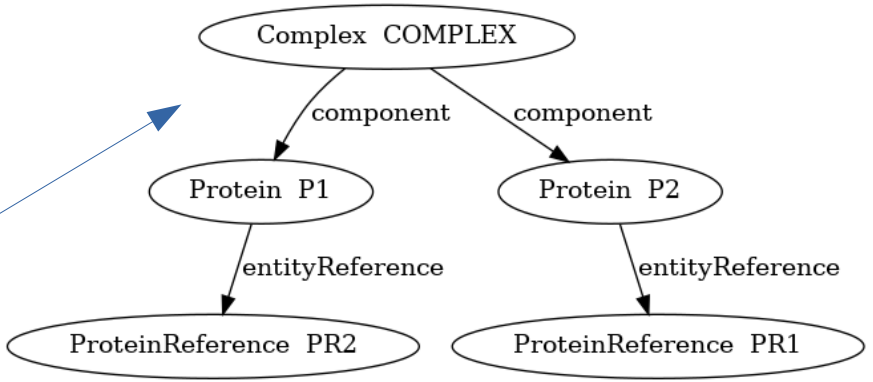
Pattern to detect inconsistant complex (recursive complexes of complexes)

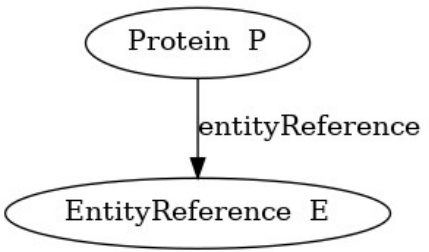
[...]

BIOPAX EXPLORER

```
datasetP = "panther"
db = "http://db:3030" # with local triple store deployed with docker-compose
#####################################################
r = Rack()
pe  = PatternExecutor(db,datasetP) # create a Pattern executor for a dataset
p = r.inComplexWith()
#####################################################
p=Pattern()
prot = EntityNode("P", Protein())
entityReference=EntityNode("E", EntityReference())
prot.connectedWith(entityReference, "entityReference")
prot.whereAttribute("comment", "Protein_9a","CONTAINS")
p.define(prot,entityReference)
#####################################################
results = pe.executePattern(p)

for entity_row in results:
  for entity in entity_row:
    print("uri:%s" %(entity.pk))
    if entity.meta_label in ['P','E']:
      print("   core entity: referenced in Pattern %s" %(type(entity)))
    else:
      print("   linked entity: neighbour from a core entity %s" %(type(entity)))
```





DSL : Object Oriented Pattern Syntax=> SPARQL=> [entities]

BIOPAX EXPLORER

```
from biopax_explorer.graph import view
gl=view.GraphDatasetLayer()
gl.model_instance_dict=gl.mpop.populate_domain_instance(
          dburl,dataset,gu.prefix(),gu.domain())
st=['Protein','SmallMolecule']
gl.g = gl.newGraph()
collec=[]
for pk in gl.model_instance_dict.keys():
   inst=gl.model_instance_dict[pk]
   if inst.__class__.__name__ in  st:
       collec.append(inst)
gl.build(collec)
gl.write_graphml('datasetlayer.graphml')
```