TRANSFORMATION DES DAC VERS OWL-S

Au sommaire de ce chapitre :

1. *Introduction.*
2. *Transformation de DAC vers OWL-S.*
3. *Implémentation de l'approche.*
4. *Exemple.*
5. *Conclusion.*

Chapitre 04 : Transformation des DAC vers OWL-S

4.1. Introduction

Après avoir présenté dans l'état de l'art la modélisation avec UML, les ontologies des services web OWL-S et la transformation de modèles, nous présentons dans cette section une approche qui consiste à générer automatiquement des ontologies OWL-S à partir des diagrammes UML (Diagramme d'activité).

Notre objectif est de faire un passage d'un diagramme UML, à savoir le diagramme d'activité vers une ontologie OWL-S. Cet objectif va être réalisé par la conversion des termes de diagramme d'activité vers des concepts d'ontologie OWL-S. Cette ontologie permet de représenter et intégrer d'une façon cohérente les connaissances des diagrammes UML.

Nous montrons dans ce chapitre les différentes phases de transformation de diagramme d'activité vers l'ontologie OWL-S, les outils de développement, les règles de transformation, la réalisation de cette transformation et comment elle fonctionne.

4.2. Transformation de DAC vers OWL-S

4.2.1. Vue d'ensemble

Les diagrammes d'activités ont été utilisés largement pour la modélisation des processus métier et des services web. Ils ont cette facilité qui permet de satisfaire les différents niveaux de modélisation des processus métiers, depuis la description générale jusqu'à la conception détaillée.

En revanche, le langage OWL-S a été aussi utilisé pour modéliser les services web, elle fournit des constructions pour indiquer des informations sur la façon de composer et d'accéder à des services web. Les deux formalismes ont des correspondances :

4.2.1.1. Méta modèle de DAC

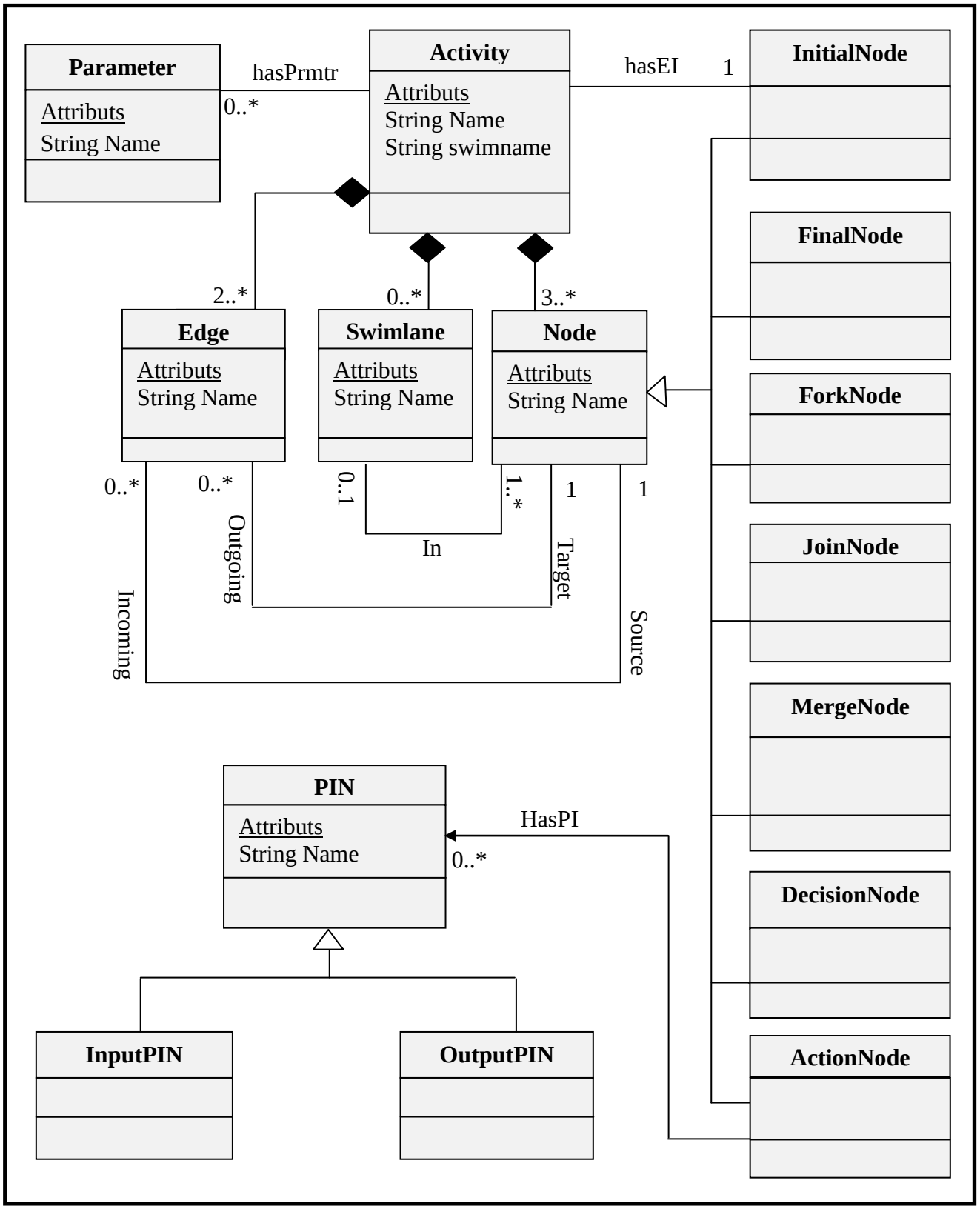


Figure 4.1 : Méta modèle de diagramme d'activité.

- **Description de Méta modèle :** Ce méta modèle se compose de plusieurs classes et des relations entre elles telles que :
  - ✓ Une "Activity" se compose au moins de trois (03) "Node", deux (02) "Edge" et zéro (0) ou plusieurs "Swimlane".
  - ✓ Une "Activity" a exactement un "InitialNode" et zéro (0) ou plusieurs "Paramater".
  - ✓ "InitialNode", "FinalNode", "MergeNode"," DecisionNode", "ForkNode", "JoinNode" et "ActionNode" sont des "Node"
  - ✓ Le "Target" d'un "Edge" est un "Node".
  - ✓ La " Source "Edge" est un "Node".
  - ✓ Les "Outgoing" d'un "Node" sont des "Edge".
  - ✓ Les "Incoming" d'un "Node" sont des "Edge".
  - ✓ Une "ActionNode" a plusieurs "Pin".
  - ✓ "InputPIN" et "OutputPIN" sont des "Pin".
  - ✓ Un "Node" peut être appartient à un "Swimlane".
  - ✓ Un "Swimlane" contient un ou plusieurs "Node".

4.2.1.2. Méta modèle d'OWLS

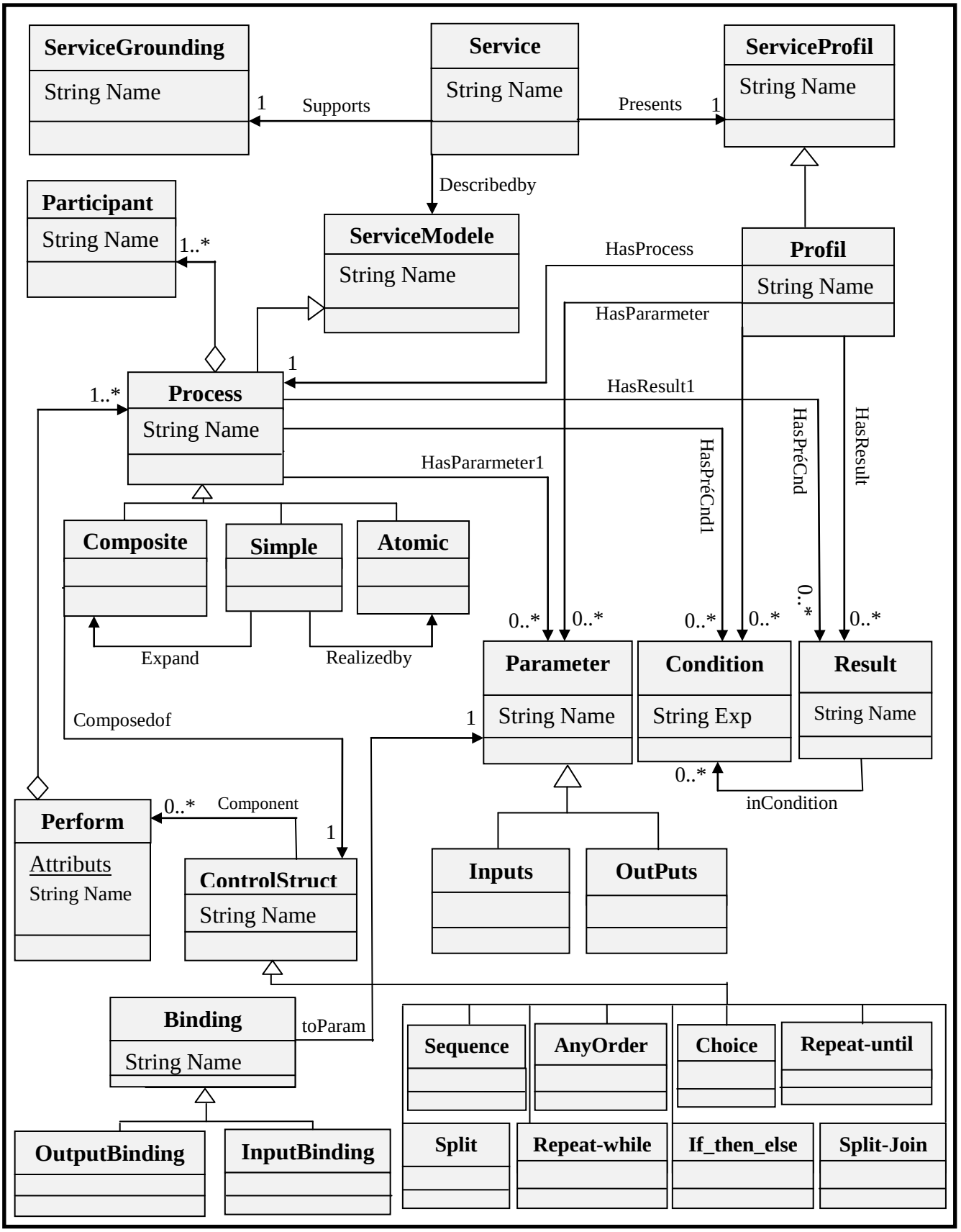


Figure 4.2 : Méta modèle d'OWL-S (Process + Profil).

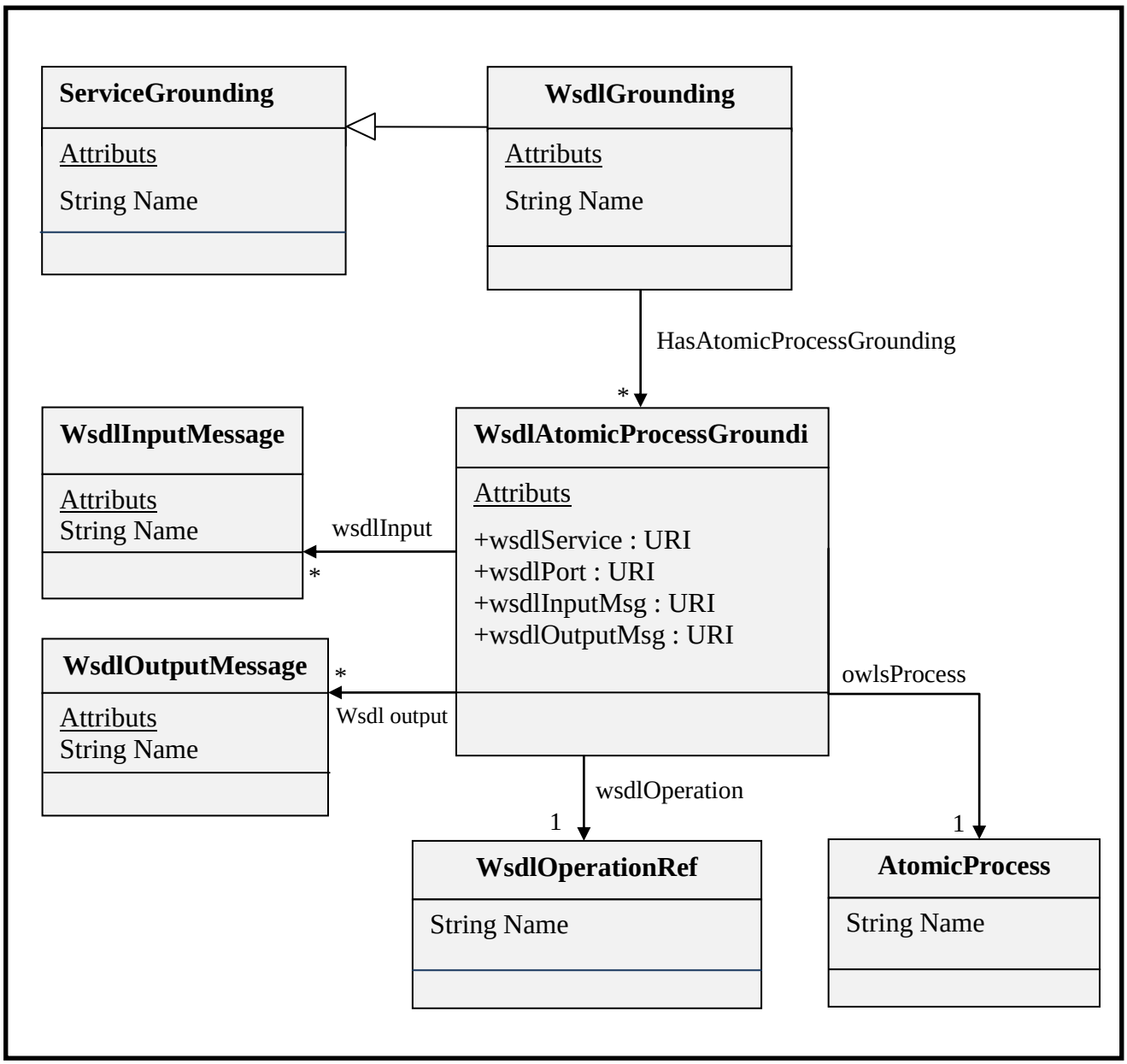


Figure 4.3 : Méta modèle d'OWL-S (Grounding).

- **Description de Méta modèle :** Ce méta modèle se compose de plusieurs classes et des relations entre elles telles que :
  - ✓ Un "Service" présente par un "ServiceProfil", décrit par un "ServiceModele" et elle est supportée par un "ServiceGrounding".
  - ✓ un "Profil" est un "ServiceProfil". Il spécifie les fonctionnalités offertes par le service qui sont représentées par des "Inputs", "Outputs", "Condition", "Result" et "Parameter".
  - ✓ Un "Process" est un "ServiceModele". Un "Process" peut être "Simple", "Atomic" ou "Composite".

- ✓ La classe "Process" a des relations avec les classes : "Inputs", "Outputs", "Condition", "Result" et "Parameter".
- ✓ Un "Composite" est composé d'une seule "ControlStruct".
- ✓ "Sequence", "Choice", "If_then_else", "Repeat-while", "Repeat-until", "Split-Join", "Split" et "AnyOrder" sont des "ControlStruct".
- ✓ "InputBinding" et "OutputBinding" sont des "Binding".
- ✓ Un "ServiceGrounding" indique comment accéder correctement au service.
- ✓ Un "WsdlGrounding" est un "ServiceGrounding".
- ✓ La classe "WsdlGrounding" a une association "HasAtomicProcessGrounding" avec la classe "WsdlAtomicProcessGrounding".
- ✓ La classe "WsdlAtomicProcessGrounding" a des attributs suivants : "wsdlService", "wsdlPort", "wsdlInputMsg" et "wsdlOutputMsg". Elle a une association "owlsProcess" avec la classe "AtomicProcess", une association "wsdlOperation" avec la classe "WsdlOPerationRef", une association "wsdlOutput" avec la classe "WsdlOutputMessage" et une association "wsdlInutput" avec la classe "WsdlInputMessage".

4.2.1.3. Phases de transformation

Dans notre étude, nous avons utilisé l'approche à base de Visiteurs, on transforme un modèle (DAC) vers texte (OWL-S) sous forme d'un fichier (.owls) pour générer un code OWL-S. Cette approche a deux mécanismes : une pour le parcours du modèle (chargement et navigation) et l'autre pour la génération du code.

En général, on peut représenter notre transformation par le schéma suivant :

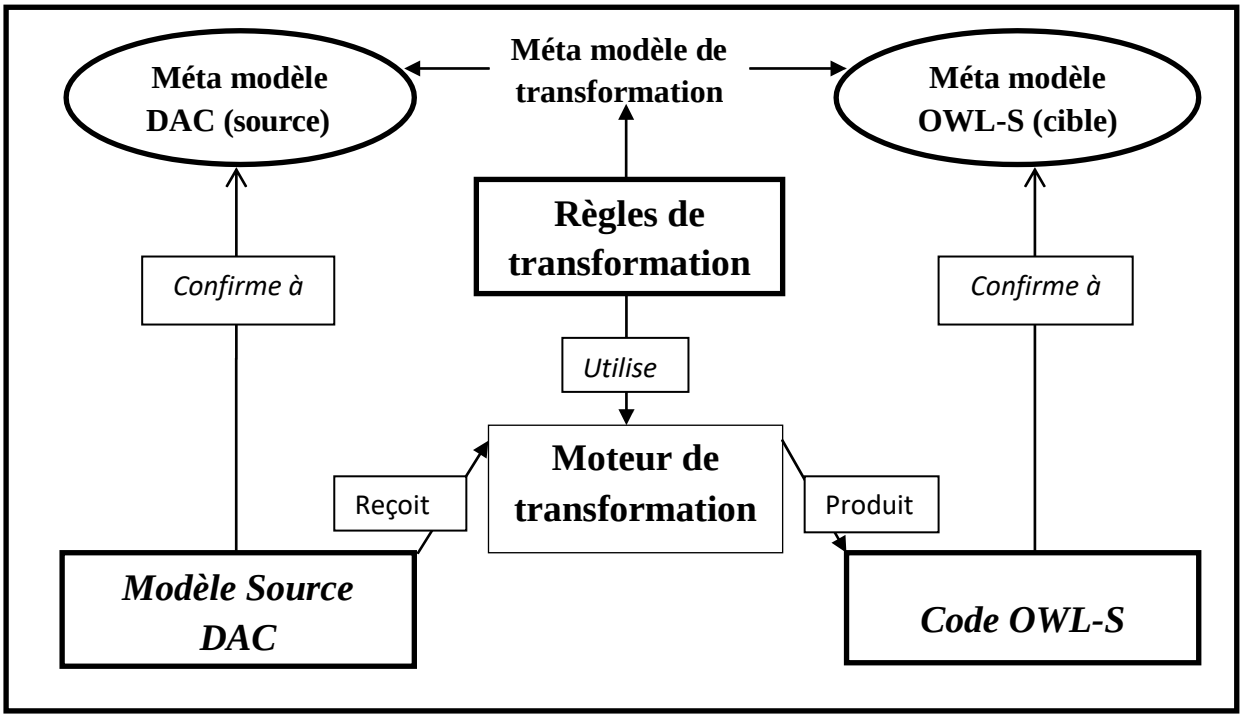
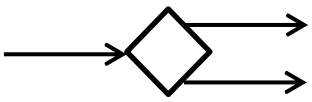
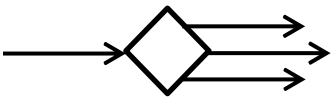


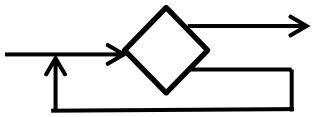
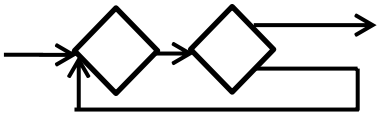
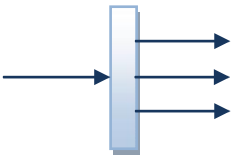
Figure 4.4 : Principe de transformation de DAC vers OWL-S

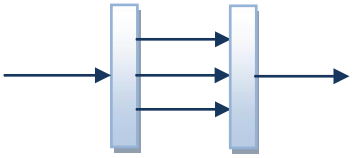
4.2.2. Règles de transformation

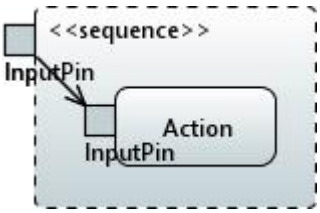
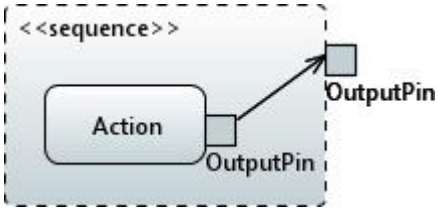
Pour transformer les concepts de DAC vers OWL-S, nous avons proposé un tableau des correspondances inspirées de [31]. [56. [60]. [61]. [62] :

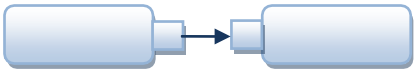
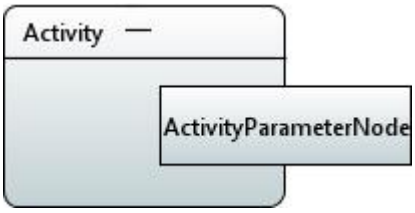
DAC (UML)	OWL-S
Activité 	**CompositeProcess** <process:CompositeProcess rdf:ID="**nom_activité**"> ……. </ process:CompositeProcess>
Une activité dans un diagramme d'activité est transformée vers un CompositeProcess dans l'ontologie OWL-S.	

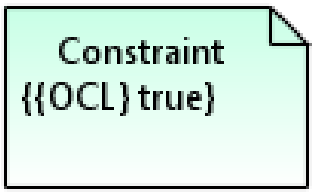
Nœud initial	**Invocable** <process:invocable rdf:datatype = "http://www.w3.org/2001/XMLSchema#boolean"> true </process:invocable>
Un nœud initial dans un diagramme d'activité est transformé vers un processus invocable dans l'ontologie OWL-S.	
Nœud de fin flot	**Nil** <process:Process>nil</process:Process>
Un nœud de fin de flot dans un diagramme d'activité est transformé vers un processus NIL dans l'ontologie OWL-S.	
Décision 	**If-then-else** < process:If-then-else> < process:If-Condition> **condition**</ process:If-Condition> < process:then>…..... </ process:then> < process:Else>…..... </ process:Else> </ process:If-then-else>
La Décision avec deux arcs sortants devient un test If-then-else dans l'ontologie OWL-S.	
Décision 	**Choice** < process:Choice> < process:Sequence> .....</ process:Sequence> < process:Sequence> .....</ process:Sequence> ……. </ process:Choice>
La Décision plus de deux arcs sortants devient choix dans l'ontologie OWL-S.	

Séquence	**Séquence** < process:Sequence>……</ process:Sequence>
Une Séquence dans un diagramme d'activité UML reste une Séquence dans l'ontologie OWL-S.	
Décision  condition	**Repeat-until** < process:Repeat-until> < process:untilProcess > ..... </ process:untilProcess > <process:untilCondition>**condition**</process:untilCondition> </ process:Repeat-until>
Une Décision avec une boucle dans un DAC devient un Repeat-until dans l'ontologie OWL-S.	
Merge+Décision  condition	**Repeat-while** < process:Repeat-while> <process:whileCondition>**condition**</process:whileCondition> <process:whileProcess> .....</process:whileProcess> </ process:Repeat-while>
Une Merge+Décision avec une boucle devient un Repeat-while dans l'ontologie OWL-S.	
Fork 	**Split** < process:Split> < process:Sequence> .....</ process:Sequence> < process:Sequence> .....</ process:Sequence> ……. </ process:Split>
Un nœud Fork dans un diagramme d'activité est transformé vers un Split dans l'ontologie OWL-S.	

Fork + Join 	**Split + Join** < process:Split-Join> < process:Sequence> .....</ process:Sequence> < process:Sequence> .....</ process:Sequence> …… </ process:Split-Join>
Si on a dans un diagramme d'activité un enchainement qui mène un Fork vers un Join, on le transforme vers un Split-Join en OWL-S.	
Action	**Atomic Process** <process:AtomicProcess rdf:ID="**nom_action**"> …… </process:AtomicProcess>
Une Action dans un diagramme d'activité est transformée vers un AtomicProcess dans l'ontologie OWL-S.	
Pin d'entrée (InputPin)	**Input** <process:Input rdf:ID="**nom_InputPin**"> ..... </process:Input>
Un InputPin dans un diagramme d'activité devient un Input dans l'ontologie OWL-S.	
Pin de sortie (OutputPin)	**Output** <process:Output rdf:ID="**nom_OutputPin**"> ...</process:Output>
Un OutputPin dans un diagramme d'activité devient un Output dans l'ontologie OWL-S.	

Pin to Pin  	**InputBinding** <process:InputBinding> <process:theParam rdf:resource="#**nom_InputAction**"/> ……. ……. <process:valueOf> <process:theParam rdf:resource="#**nom_InputComposite**"/> <process:fromProcess rdf:resource="#**nom_Composite**"/> </process:valueOf> …… </process:InputBinding>
Lorsqu'on a une Action Simple et un Action Composite, et un pin d'entrée vers un pin d'entrée, dans ce cas on les transforme vers un InputBinding dans l'ontologie OWL-S.	
Pin to Pin  	**OutputBinding** <process:OutputBinding> <process:theParam rdf:resource="#**nom_Output_Composite**"/> …… <process:valueOf> <process:theParam rdf:resource="# **nom_Output_Action** "/> <process:fromProcess rdf:resource="#**nom_Coposite**"/> </process:valueOf> …… </process:OutputBinding>
Lorsqu'on a une Action Simple et un Action Composite, et un pin de sortie vers un pin de sortie, dans ce cas on les transforme vers un OutputBinding dans l'ontologie OWL-S.	

Pin to Pin 	**Binding** <process:Binding> <process:theParam rdf:resource="#**nom_InputPin2**"/> <process:valueSource> <process:ValueOf> <process:theParam rdf:resource="#**nom_OutputPin1**"/> <process:fromProcess rdf:resource = "#**nom_action1**"/> </process:ValueOf> </process:valueSource> </process:Binding>
Lorsqu'on a deux Actions simples dans le même Action Composite, et un pin de sortie vers un pin d'entrée, dans ce cas on les transforme vers un Binding dans l'ontologie OWL-S.	
Partition 	**Participant** <process:Parameter rdf:ID="**nom_Partition**"/>
Une Partition dans un diagramme d'activité est transformée vers un Participant dans l'ontologie OWL-S. En général, un participant peut être les clients ou le serveur.	
ActivityParameter  	**Parameter** < process:Parameter rdf:ID = "**nom_Paramètre**"> …… < /process:Parameter>
Un nœud Paramètre d'activité dans un diagramme d'activité est transformé vers un Paramètre dans l'ontologie OWL-S.	

Contrainte  	**Pré-condition** <process:hasPrecondition> <expr:**langage de condition**-Condition> ……..…… </expr: **langage de condition** -Condition> </process:hasPrecondition>
	Result <process:hasResult> <process:Result> <process:inCondition> <expr: **langage de condition** -Condition> …....…… <process:hasEffect> < expr: **langage de condition** -Condition> …… </process:Result> </process:hasResult>
	Effect <process:hasEffect> <expr: **langage de condition** -Condition> <expr:expressionBody>…….</expr:expressionBody> </expr: **langage de condition** -Condition> </process:hasEffect>
Une Contrainte dans un diagramme d'activité est transformée vers un Pré-condition / Result ou Effect dans l'ontologie OWL-S.	

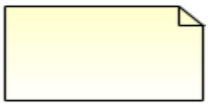
Commentaire (Comment) 	**Comment** <process:Comment>**Body_commentaire** </process:Comment>
Un Commentaire dans un diagramme d'activité reste un Commentaire dans l'ontologie OWL-S.	

Tableau 4.1 : Tableau de correspondances entre DAC et OWL-S.

4. 3. Implémentation de l'approche

4. 3.1. Outils de développement

4.3.1.1. Eclipse : plate-forme

- ***Définition***

Pour le choix de l'environnement de développement, on a opté pour Eclipse car il possède de nombreux points forts qui sont à l'origine de son énorme succès, alors Eclipse est un environnement de développement intégré (*IDE*) réalisé par la fondation Eclipse. Il est gratuit (open source), extensible, universel et polyvalent, permettant la mise en place des projets de développement utilisant n'importe quel langage de programmation [39].

- ***Points fort d'Eclipse***

Éclipse possède de nombreux pions forts qui sont à l'origine de son énorme succès dont les principaux sont [38]:

- Une plateforme ouverte pour le développement d'applications et extensible grâce à un mécanisme de plugins;
- Support de plusieurs plates-formes d'exécution : Windows, Linux, Mac OS;
- Malgré son écriture en Java, Eclipse est très rapide à l'exécution grâce à l'utilisation de la bibliothèque SWT;

- Eclipse IDE est principalement rédigé en Java (à l'aide de la bibliothèque graphique SWT (Standard Widgets Toolkit), d'IBM, et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions.

4.3.1.2. Papyrus : outil pour la méta-modélisation

Papyrus est open source (licence EPL), il est basé sur la technologie Eclipse. Il s'agit d'une version graphique du plugin UML2 de Eclipse.

- ***Caractéristiques:***
  - ✓ Il est basé sur le plugin UML2 de Eclipse.
  - ✓ Il respect du standard UML2 défini à l'OMG.
  - ✓ Il respect de la norme Diagram Interchange défini à l'OMG.
  - ✓ Il a une architecture extensible pour des nouveaux diagrammes et générations de code.
  - ✓ gestion des profils.
  - ✓ Il Supporte bien le diagramme d'activité UML.

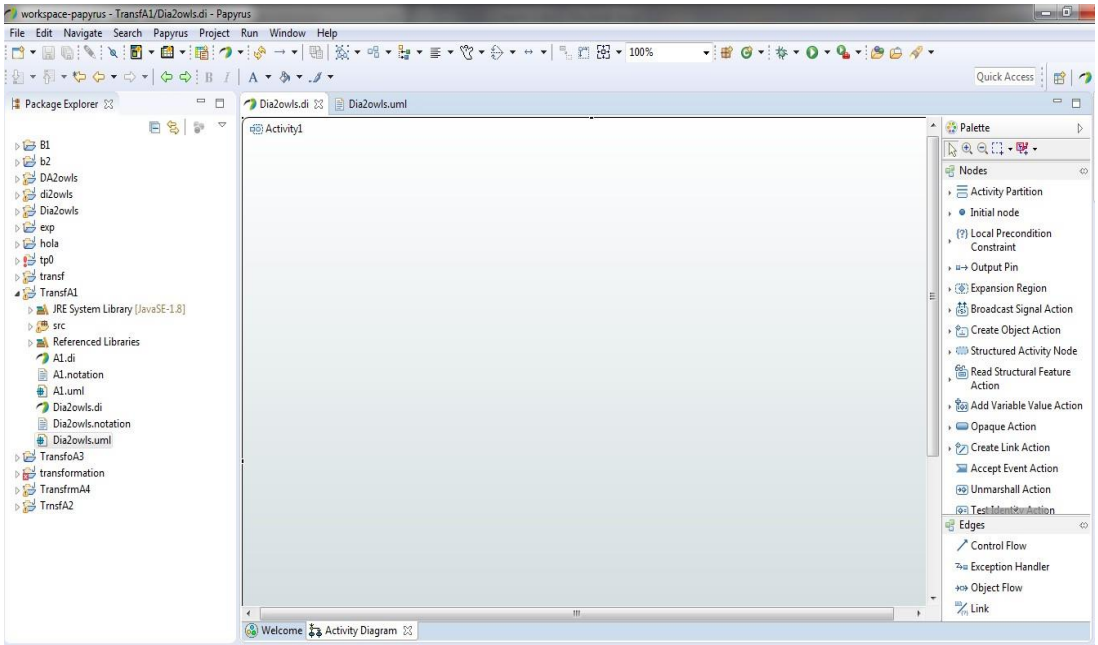


Figure 4.5 : L'outil Papyrus avec la palette diagramme d'activité.

4.3.1.3. Java langage de programmation

Java est un langage de programmation orienté objet élaboré par Sun Microsystems (aujourd'hui repris par Oracle) [57]. C'est un langage orienté objet simple ce qui réduit les risques d'incohérence, il est portable, il peut être utilisé sous Windows, sous Linux, sous Macintosh et sur

d'autres plateformes sans aucun souci, enfin il possède une riche bibliothèque de classes comprenant des fonctions diverses tels que les fonctions standards, le système de gestion de fichiers, les fonctions multimédia et beaucoup d'autres fonctionnalités [58].
Dans notre approche, on l'a choisi car il est très souple dans la manipulation de l'XML.

4.3.1.4. DOM

DOM est l'acronyme de *Document Object Model*. C'est une spécification du W3C pour proposer une API qui permet de modéliser, de parcourir et de manipuler un document XML. Le principal rôle de DOM est de fournir une représentation mémoire d'un document XML sous forme d'un arbre d'objets et d'en permettre la manipulation (parcours, recherche et mise à jour).

JDOM est une API open source Java, vu comme un modèle de documents objets dont le but est de représenter et manipuler un document XML de manière intuitive pour un développeur Java sans requérir une connaissance pointue de XML. JDOM propose aussi une intégration de SAX, DOM, XSLT et XPath.

Un document XML est encapsulé dans un objet de type Document. Les éléments d'un document sont encapsulés dans des classes dédiées : Element, Attribute, Text, Processing Instruction, Namespace, Comment, DocType, EntityRef, CDATA. JDOM permet aussi de vérifier que les données contenues dans les éléments respectent la norme XML.

JDOM propose des réponses à certaines faiblesses de SAX et DOM. La simplicité d'utilisation de JDOM lui permet d'être une API dont l'utilisation est assez répandue [58].

4.3.1.5. Protégé

L'outil Protégé est conçu pour libérer l'ontologiste de la complexité de la construction de l'ontologie. Il permet de générer automatiquement le code OWL correspondant à l'ontologie [59].

4.3.1.6. OWL-S Editor pour Protégé

OWL-S Editor pour Protégé est un outil graphique visé à la conception des services Web sémantiques. Il fournit un ensemble complet de fonctionnalités pour créer et maintenir des descriptions de service OWL-S, avec interaction facile. Il est organisé autour de la structure conceptuelle d'OWL-S. [59]

4.3.2. Présentation de l'application

4.3.2.1. Scénario de la transformation

Cette transformation se fait par une génération automatique d'une ontologie OWL-S représentée en RDF/XML à partir d'un diagramme d'activité UML édité en Papyrus, cette solution est implémentée en Eclipse, Elle se déroule en plusieurs étapes (cf. Figure. 4.6) :

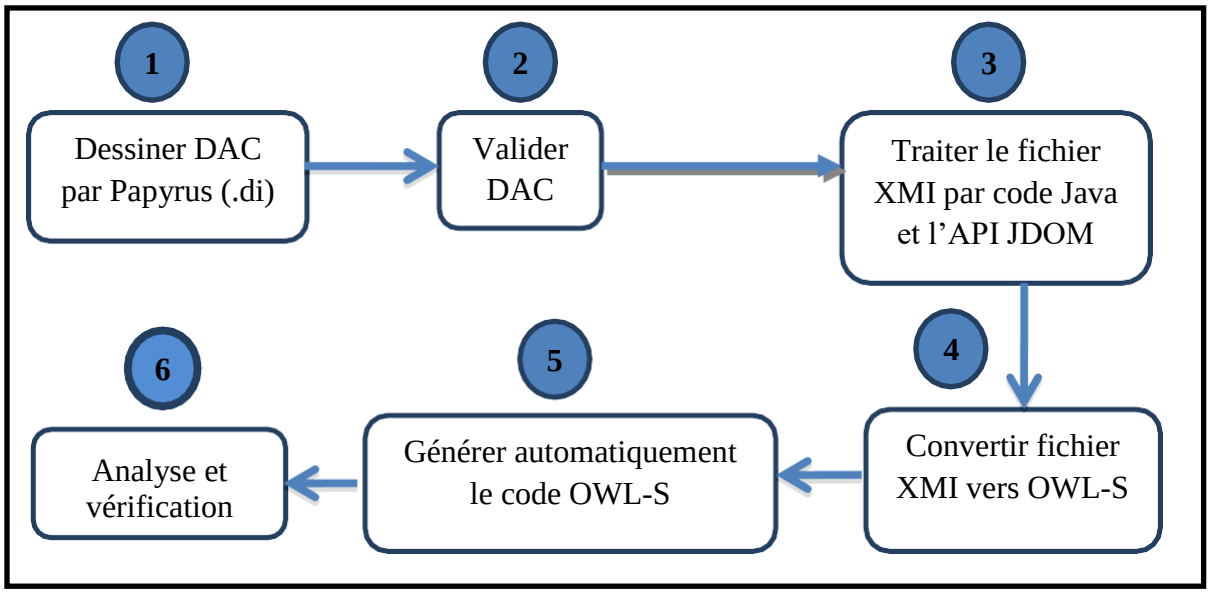


Figure 4.6 : Processus générale de la transformation de DAC vers OWL-S.

- **Phase 01 : (Dessiner DAC par Papyrus (.di))** : Pour commencer la construction de notre ontologie, il faut d'abord dessiner notre diagramme d'activité par l'outil Papyrus, qui permet de modéliser le diagramme d'activité à partir de la palette qui contient son Méta modèle. Quand on sauvegardera le DAC, on a trois fichiers créés : le fichier dont l'extension « .di » qui contient l'apparence graphique du diagramme, le fichier « .nottation » et le fichier « .uml » qui on le traitera.

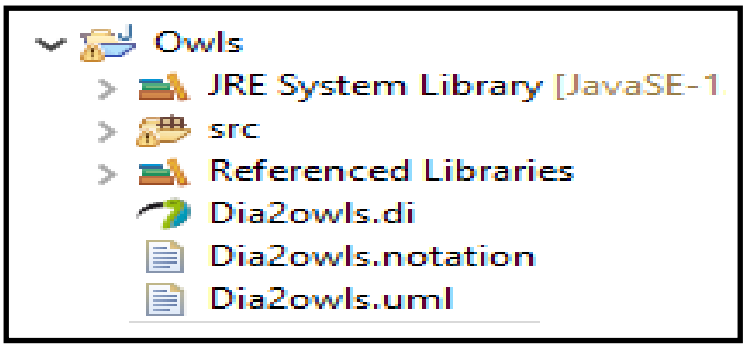


Figure 4.7 : Les trois fichiers créés pour un modèle Papyrus - DAC.

- **Phase 02 : (Valider DAC)** : Avant le transformer notre DAC, il faut d'abord le valider. Et pour valider, on a plusieurs cas variés, mais on s'est concentré sur les points suivants :
  - ✓ Chaque activité contient un seule nœud initial et au moins un nœud final.
  - ✓ Une activité contient au moins un nœud action.
  - ✓ Un nœud initial a un seul arc sortant et pas des arcs entrants.
  - ✓ Un nœud final et nœud de flux final ont des arcs entrants et pas des arcs sortants.
  - ✓ Chaque nœud (sauf l'état initial et final) a des arcs entrants et sortants.
  - ✓ Dans une activité, l'action a un seul arc sortant.
  - ✓ Action n'a pas des arcs bidirectionnels, ni boucle.
  - ✓ Nœud "DecisionNode" et "ForkNode" contient plusieurs sorties et une seule entrée.
  - ✓ Il faut nommer les arcs sortants d'un nœud "DecisionNode" par des gardes de décision.
  - ✓ Nœud "JoinNode" et "MergeNode" contient une seule sortie et plusieurs entrées.
  - ✓ Nœud paramètre n'a pas d'arcs ni d'entrées ni de sorties.

```
public static void validate_mod(Element current)
{List<Attribute> atrc=current.getAttributes();
if (atrc.get(0).getValue().equals("uml:Activity"))
{          Integer n=0, c=0; n=cherche_elem(current,"uml:InitialNode");
if (n==0) {System.out.println("Erreur : Activity
\""+current.getAttributeValue("name")+"\" manque d'un Noeud initial");
valide=false;}
}
else
{
if (n>1) {System.out.println("Erreur : Activity
\""+current.getAttributeValue("name")+"\" contient plus d'un Noeud initial");
valide=false;}
else
{          n=cherche_elem(current,"uml:ActivityFinalNode");
if (n==0) {System.out.println("Erreur : Activity
\""+current.getAttributeValue("name")+"\" manque d'un Noeud final");
valide=false;}
……
……
……
……
……
……
……

{if (at.equals("uml:Activity")) {validate_mod(child); if (!valide) b=false; } else
if (at.equals("uml:SequenceNode"))       {
if ((child.getAttributeValue("incoming")==null) || (child.getAttributeValue("outgoing")==null))
{System.out.println("Erreur
: Noeud de Sequence \""+child.getAttributeValue("name")+"\" a un seul flèche entrant et un seul sortant");
valide=false; b=false;}
else      {
String[] sout=child.getAttributeValue("outgoing").split(" ");
if (sout.length>1)
{System.out.println("Erreur : Noeud de Sequence
\""+child.getAttributeValue("name")+"\" a "+sout.length+" flèches sortants"); valide=false; b=false;          }
else {    String[] sin=child.getAttributeValue("incoming").split(" "); if (sin.length>1)
{System.out.println("Erreur : Noeud de Sequence
\""+child.getAttributeValue("name")+"\" a "+sin.length+" un flèches entrant"); valide=false; b=false;        }
}          } }
if ((c==0) && valide)    { System.out.println("Erreur : Activity
\""+current.getAttributeValue("name")+"\" ne contient pas aucun node");
valide=false; b=false; }
else      { if ((c!=0) && valide)
{ bouc(racin);
if (boucler) {System.out.println("Erreur : Noeud \""+cherche_nom(bid2)+"\" a boucler avec Noeud
\""+cherche_nom(bid1)+"\"" ); valide=false; b=false;}}
}
}}}}}
```

Figure 4.8 : La procédure de validation du modèle "validate_mod".

- **Phase 03 : (Traiter le fichier XMI par code Java et l'API JDOM)** : C'est une phase très importante dans notre travail. On a traité le fichier XMI (.uml) par un code Java et le parcourt (arbre) par une API JDOM, en utilisant la récursivité. Et pour faciliter notre travail, on utilise les pointeurs (les listes chainées bidirectionnelles illustrées dans la figure 4.9 et les piles illustrées dans la figure 4.10) comme des structures de données. Les listes contiennent tous les nœuds dans notre DAC, ces derniers ont des prédécesseurs et des successeurs. Pour les piles, on empile lorsqu'on ouverte une structure de contrôle et on dépile pour la fermeture.

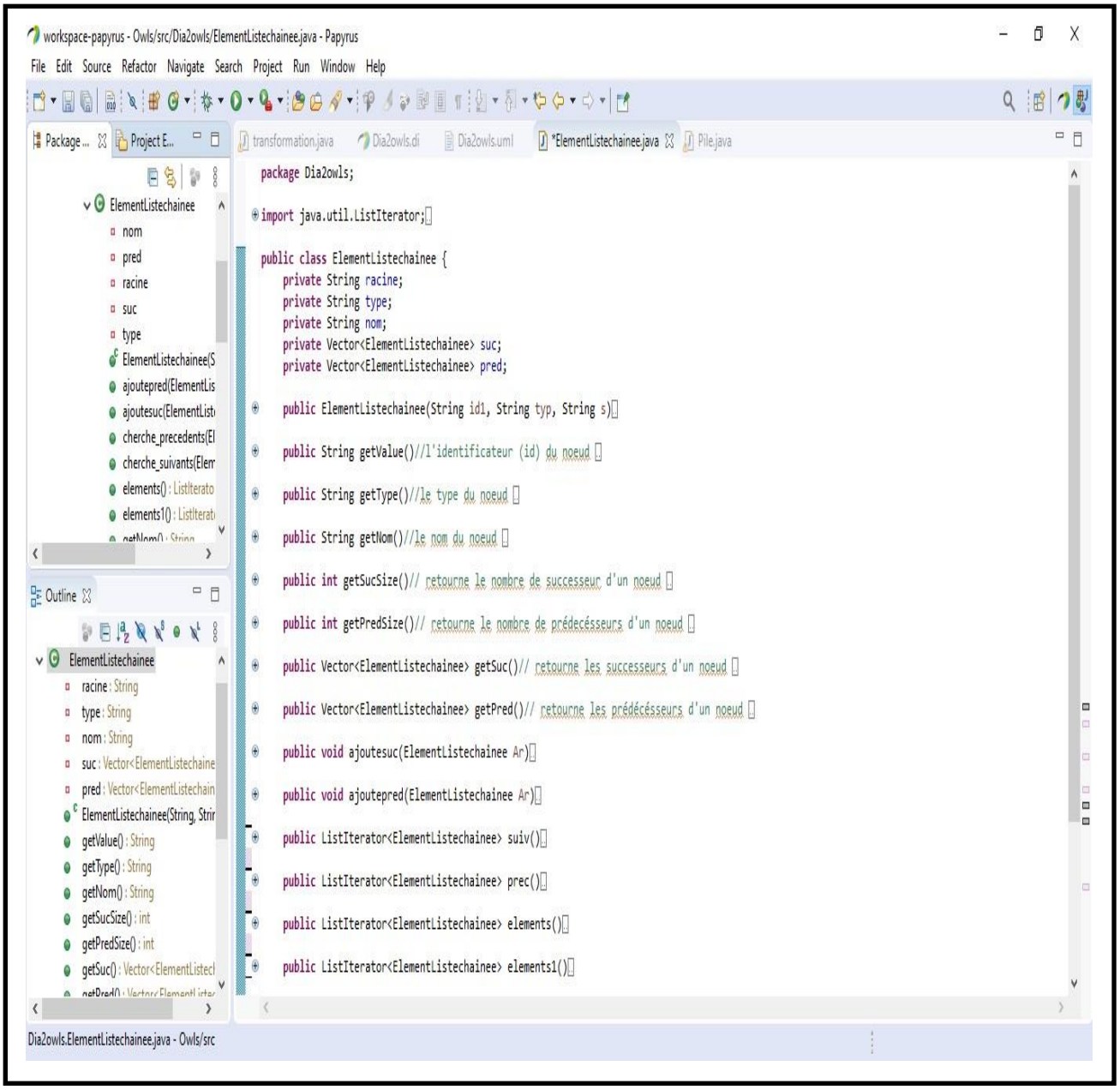


Figure 4.9 : La classe ElementListechainee.java.

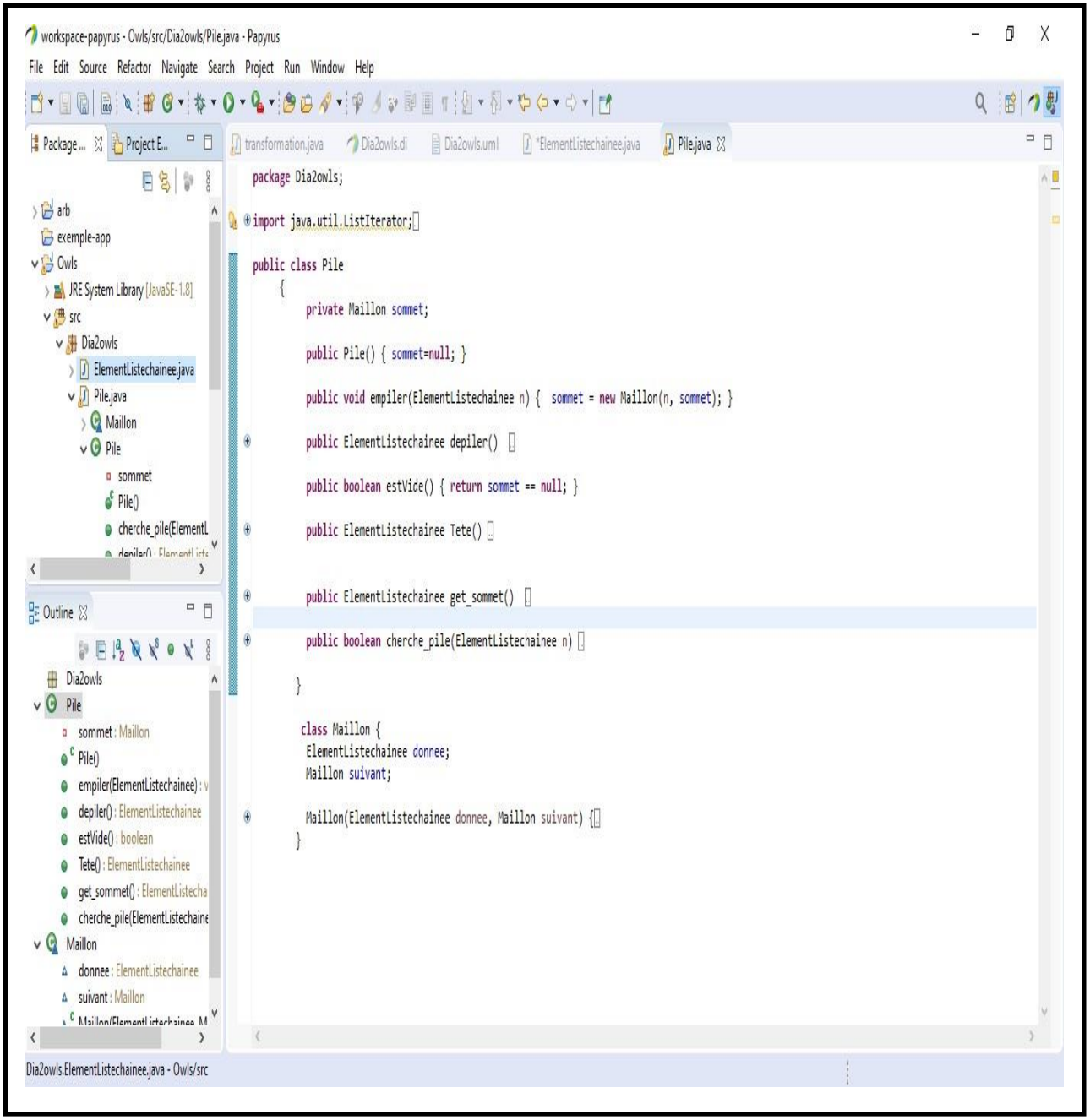


Figure 4.10 : La classe Pile.java

- **Phase 04 : (Convertir fichier XMI vers OWL-S)** : C'est la plus importante phase dans notre travail. À partir des règles de transformation obtenues, on fait la conversion, où chaque règle a une correspondance en Java.

```
public static void parcourir_arb(FileWriter file2, String s) throws IOException  {p =new Pile(); depth=2;
        file2.write("\n\n<!-- Instance Definition of "+s+ "_Process -->\n");
        file2.write("\n\n<process:CompositeProcess rdf:ID=\""+s+"_Process\">\n");
        file2.write("  <process:invocable
rdf:datatype=\"http://www.w3.org/2001/XMLSchema#boolean\"> true</process:invocable>\n");
        file2.write(" <process:Sequence>\n"); parcourir(file2,tete.getSuc().get(0),depth);
         file2.write(" </process:Sequence>\n"); file2.write("</process:CompositeProcess>\n\n"); }

public static void parcourir(FileWriter file2, ElementListechainee elem, int depth)throws IOException
        { if (elem.getType().equals("uml:SequenceNode"))
          {  p.empiler(elem);  ElementListechainee elem_supp=p.depiler();
          ElementListechainee elem_supp=p.depiler();  aff(elem.getValue());
          file2.write(printSpaces(depth)+"<process:CompositeProcess
          rdf:about=\"#"+elem.getNom()+"\"/>\n");
            if (!(elem.getSuc().get(0).getType().equals("uml:ActivityFinalNode")))
                parcourir(file2,elem.getSuc().get(0),depth)}
           else {  if (elem.getType().equals("uml:FlowFinalNode")) {
                  p.empiler(elem); aff(elem.getValue());
                 file2.write(printSpaces(depth)+"<process:Process>nil</process:Process>\n");
                ElementListechainee elem_supp=p.depiler();  }
                Else { if (elem.getType().equals("uml:MergeNode")) //test pour une boucle while
                 { if (!if_aff(elem.getValue()))
                   {  ElementListechainee elemd=elem.getSuc().get(0);
                    if (repeat_while(elemd.getValue(),elemd.getType()))
                    { Boolean b=false; int i=0;  String cond="";
               while ((i<elemd.getSucSize()) && (!b))
                     {    anc=elemd.getValue();
                 if (b_fils_anc(elemd.getValue(),elemd.getSuc().get(i).getValue()))
                 {  aff(elem.getValue());
          cond=get_edge(racin,elemd.getValue(),elemd.getSuc().get(i).getValue());
                          p.empiler(elemd); aff(elemd.getValue());
          file2.write(printSpaces(depth)+"<process:Repeat-while>\n");   depth++;
          file2.write(printSpaces(depth)+"<process:whileCondition>"+cond+"</process:whileCondition>\
          n");       file2.write(printSpaces(depth)+"<process:whileProcess>\n"); depth++;
          parcourir(file2,elemd.getSuc().get(i),depth);  b=true; }
                          else i++;} depth--; depth--;
                for (int j=0; j<elemd.getSucSize(); j++) // aprés la boucle while
                   {……
……
……
   ….. if (elem.getValue().equals(p.get_sommet().getValue
       {ElementListechainee elem_sup= p.depiler();
         if (elem_sup.getType().equals("uml:DecisionNode"))
          if (elem_sup.getSucSize()==2) {  depth--; file2.write(printSpaces(depth)+"</process:Sequence>\n");
               depth--; file2.write(printSpaces(depth)+"</process:Else>\n");
               depth--; file2.write(printSpaces(depth)+"</process:If-then-else>\n");}
           else {file2.write(printSpaces(depth)+"</process:Choice>\n");}
         else {depth--;  file2.write(printSpaces(depth)+"</process:Sequence>\n");
              create_fils_fork(elem_sup); String sp=""; depth--;
              if (split_join(elem_sup))  sp="Split-Join"; else sp="Split";
        file2.write(printSpaces(depth)+"</process:"+sp+">\n");}}}}}}}}
```

Figure 4.11 : Les deux procédures de transformation de fichier XMI.

- **Phase 05 : (Générer automatiquement le code OWL-S)** : Trois fichiers (ResultService.owl, ResultProcess.owl et ResultGrounding.owl) sont générés automatiquement et contiennent l'ontologie OWL-S. Ils sont stockés sur disque.

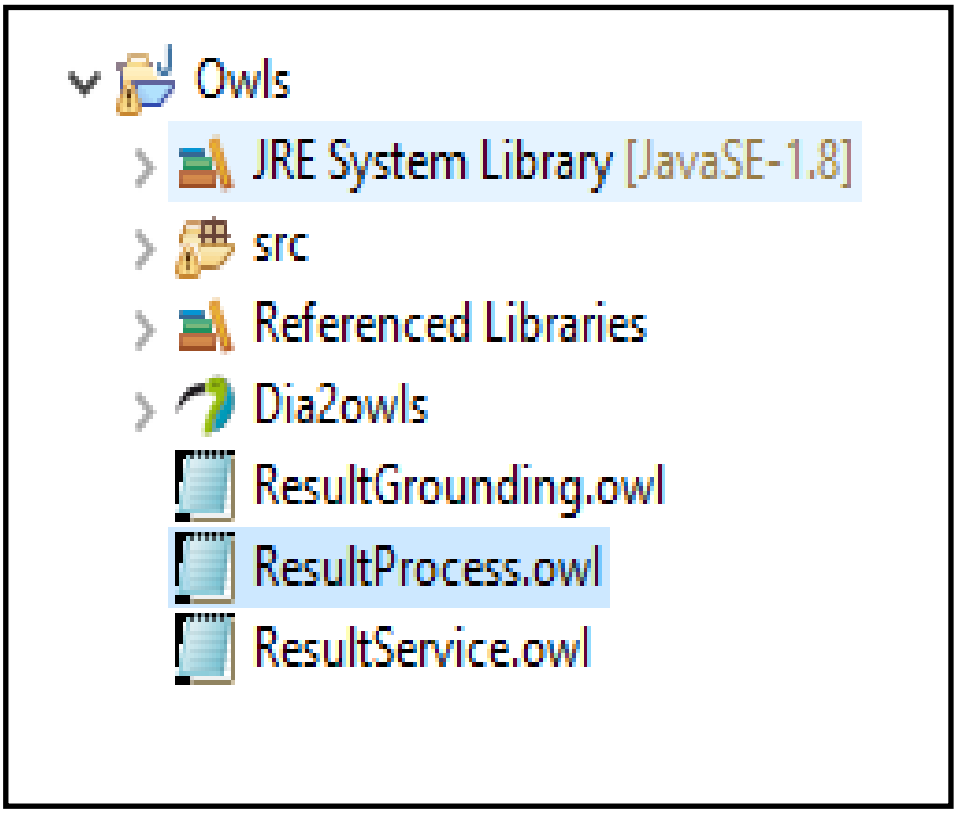


Figure 4.12 : Les trois fichiers résultats (.owl).

- **Phase 06 : (Analyse et vérification)** : On charge le code OWL-S (on merge les trois fichiers obtenus : ResultService.owl, ResultProcess.owl et ResultGrounding.owl) généré sur l'outil OWL-S Editor pour Protégé pour le vérifier.

4.4. Exemple

Pour pouvoir concrétiser notre travail, on a essayé de l'appliquer sur l'exemple de diagramme d'activité présenté dans la figure ci-dessous. L'exemple a été expliqué dans le chapitre1. Il est à noter que cet exemple ne prétend pas être exhaustif mais montre quelques pistes dans notre transformation tel que la manipulation des actions, décisions, merges, forks, joins, séquences.

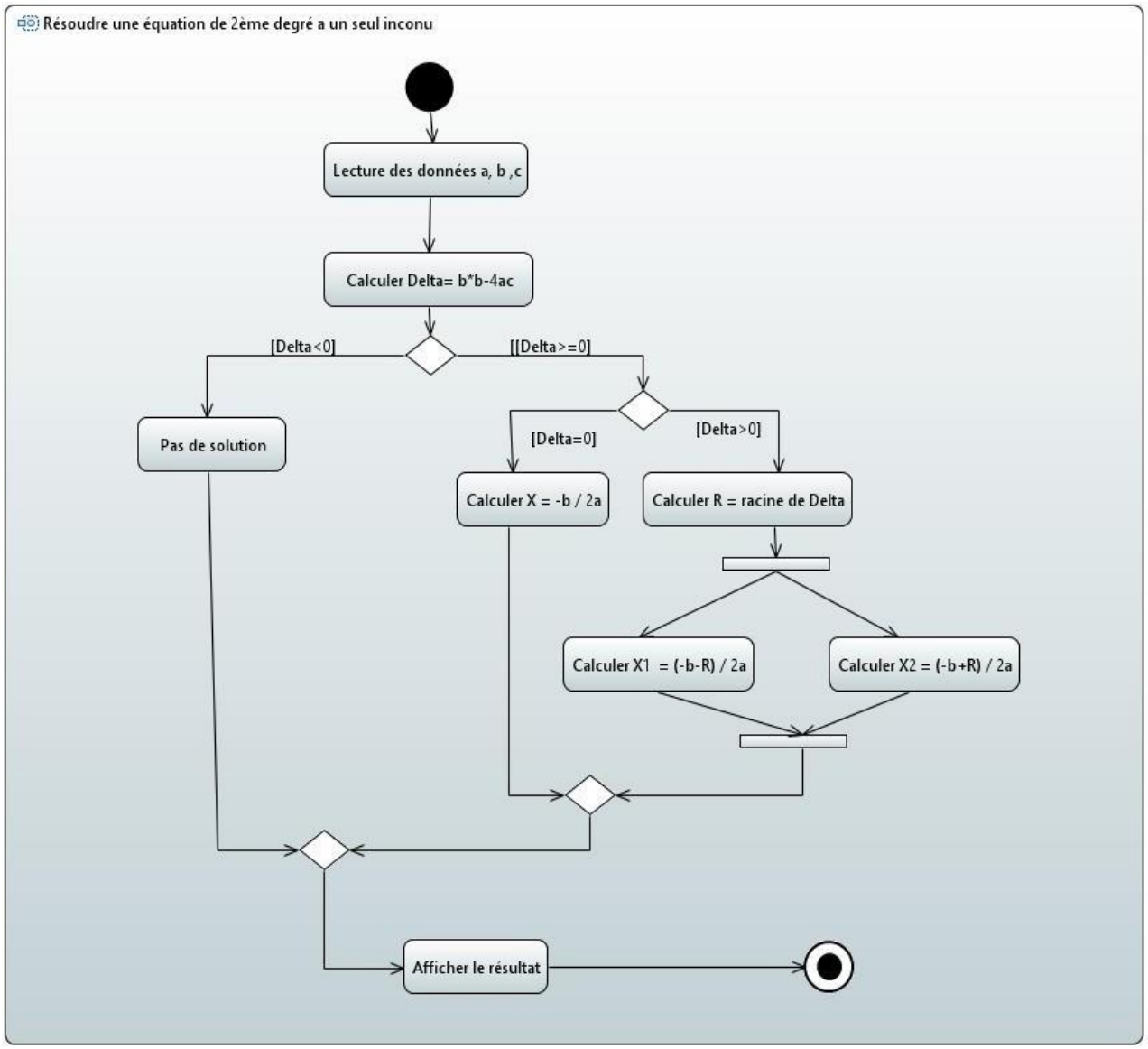


Figure 4.13 : Exemple de diagramme d'activité "Dia2owls"

Et comme nous avons vu précédemment chaque diagramme édité est stocké dans un fichier avec extension (.uml), dans notre cas le fichier "Dia2owls.uml" dans la figure 4.14 :

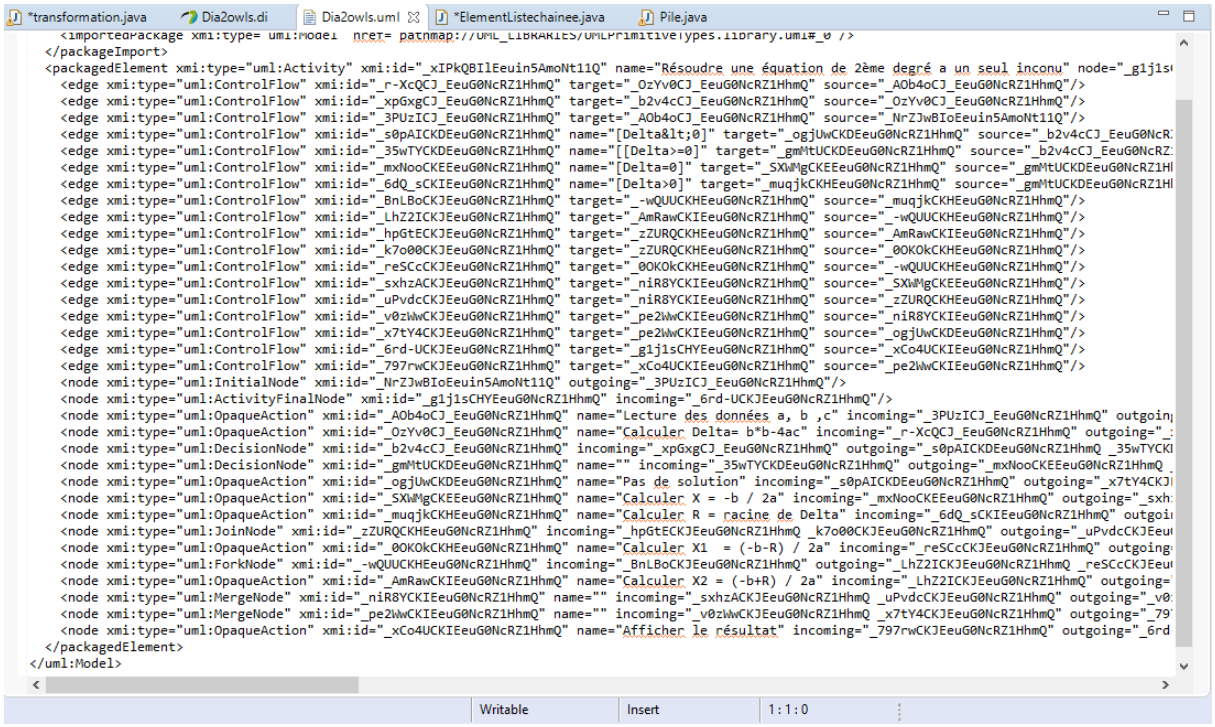


Figure 4.14 : Le fichier "Dia2owls.uml" correspondant au fichier "Dia2owls.di".

Après l'application de notre transformation sur le fichier "Dia2owls.uml", nous obtenons trois fichiers (.owl) qui contiennent notre ontologie OWL-S (Tableau 4.2) :

Fichier ResultService.owl
<?xml version='1.0' encoding='ISO-8859-1'?> <!DOCTYPE uridef[<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns"> <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema"> <!ENTITY owl "http://www.w3.org/2002/07/owl"> <!ENTITY xsd "http://www.w3.org/2001/XMLSchema"> <!ENTITY service "http://www.daml.org/services/owl-s/1.0/Service.owl"> <!ENTITY r_process "http://www.daml.org/services/owl-s/1.0/ResultProcess.owl"> <!ENTITY r_grounding "http://www.daml.org/services/owl-s/1.0/ResultGrounding.owl"> <!ENTITY DEFAULT "http://www.daml.org/services/owl-s/1.0/ResultService.owl">]> <rdf:RDF xmlns:rdf = "&rdf;#" xmlns:rdfs ="&rdfs;#" xmlns:owl = "&owl;#" xmlns:xsd ="&xsd;#" xmlns:service= "&service;#"

```
  xmlns          ="&DEFAULT;#"
>

 <owl:Ontology about="">
    <owl:versionInfo>$Id: ResultSetvice.owl,v 1.1 2020/10/15 01:00:00 martin Exp
$</owl:versionInfo>
    <rdfs:comment> This ontology represents the OWL-S service description for the web
service example. </rdfs:comment>
    <owl:imports rdf:resource="&service;" />
    <owl:imports rdf:resource="&r_process;" />
 </owl:Ontology>

<service:Service rdf:ID="Résoudre une équation de 2ème degré a un seul inconuService">
<service:describedBy rdf:resource="&r_process;#Résoudre une équation de 2ème degré a un
seul inconu_ProcessModel"/>
<service:supports rdf:resource="&r_grounding;#Grounding_Résoudre une équation de 2ème
degré a un seul inconu"/>
</service:Service>

</rdf:RDF>
```

Fichier ResultProcess.owl

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.0/Service.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.0/Process.owl">
  <!ENTITY r_service "http://www.daml.org/services/owl-s/1.0/ResultService.owl">
  <!ENTITY DEFAULT "http://www.daml.org/services/owl-s/1.0/ResultProcess.owl">
]>

<rdf:RDF
  xmlns:rdf      = "&rdf;#"
  xmlns:rdfs     ="&rdfs;#"
  xmlns:owl = "&owl;#"
  xmlns:xsd      ="&xsd;#"
  xmlns:service=          "&service;#"
  xmlns:process=          "&process;#"
  xmlns          ="&DEFAULT;#"

 <owl:Ontology about="">
    <owl:versionInfo>$Id: ResultProcess.owl,v 1.1 2020/10/15 08:30:00 martin Exp
```

```
$</owl:versionInfo>
    <rdfs:comment> Example for OWL-S Process Model </rdfs:comment>
    <owl:imports rdf:resource="&service;" />
    <owl:imports rdf:resource="&r_process;" />
 </owl:Ontology>

<process:ProcessModel rdf:ID="Résoudre une équation de 2ème degré a un seul
inconu_ProcessModel">
   <process:hasProcess rdf:resource="#Résoudre une équation de 2ème degré a un seul
   inconu_Process" />
   <service:describes rdf:resource="&r_service;#Résoudre une équation de 2ème degré a un
   seul inconuService"/>
</process:ProcessModel>

<!-- Instance Definition of Résoudre une équation de 2ème degré a un seul inconu_Process -->

<process:CompositeProcess rdf:ID="Résoudre une équation de 2ème degré a un seul
inconu_Process">
   <process:invocable rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"> true
   </process:invocable>
   <process:Sequence>
      <process:Process>#Lecture des données a, b ,c</process:Process>
      <process:Process>#Calculer Delta= b*b-4ac</process:Process>
      <process:If-then-else>
         <process:If-Condition>Delta<0</process:If-Condition>
         <process:then>
            <process:Sequence>
               <process:Process>#Pas de solution</process:Process>
            </process:Sequence>
         </process:then>
         <process:Else>
            <process:Sequence>
               <process:If-then-else>
                  <process:If-Condition>Delta=0</process:If-Condition>
                  <process:then>
                     <process:Sequence>
                        <process:Process>#Calculer X = -b / 2a</process:Process>
                     </process:Sequence>
                  </process:then>
                  <process:Else>
                     <process:Sequence>
                        <process:Process>#Calculer R = racine de Delta</process:Process>
                        <process:Split-Join>
                           <process:Sequence>
                              <process:Process>#Calculer X2 = (-b+R) / 2a</process:Process>
                           </process:Sequence>
                           <process:Sequence>
                              <process:Process>#Calculer X1 = (-b-R) / 2a</process:Process>
```

```
                    </process:Sequence>
                  </process:Split-Join>
                </process:Sequence>
              </process:Else>
            </process:If-then-else>
          </process:Sequence>
        </process:Else>
      </process:If-then-else>
      <process:Process>#Afficher le résultat</process:Process>
    </process:Sequence>
</process:CompositeProcess>

<process:AtomicProcess rdf:ID="Lecture des données a, b ,c"/>
<process:AtomicProcess rdf:ID="Calculer Delta= b*b-4ac"/>
<process:AtomicProcess rdf:ID="Pas de solution"/>
<process:AtomicProcess rdf:ID="Calculer X = -b / 2a"/>
<process:AtomicProcess rdf:ID="Calculer R = racine de Delta"/>
<process:AtomicProcess rdf:ID="Calculer X1  = (-b-R) / 2a"/>
<process:AtomicProcess rdf:ID="Calculer X2 = (-b+R) / 2a"/>
<process:AtomicProcess rdf:ID="Afficher le résultat"/>
</rdf :RDF>
```

Fichier ResultGrounding.owl

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.0/Service.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.0/Grounding.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.0/Process.owl">
  <!ENTITY r_service "http://www.daml.org/services/owl-s/1.0/ResultService.owl">
  <!ENTITY r_process "http://www.daml.org/services/owl-s/1.0/ResultProcess.owl">
  <!ENTITY ResultGroundingWSDL "http://www.daml.org/services/owl-
s/1.0/ResultGrounding.wsdl">
  <!ENTITY DEFAULT "http://www.daml.org/services/owl-s/1.0/ResultGrounding.owl">
]>

<rdf:RDF
  xmlns:rdf      = "&rdf;#"
  xmlns:rdfs     ="&rdfs;#"
  xmlns:owl = "&owl;#"
  xmlns:xsd      ="&xsd;#"
  xmlns:service=        "&service;#"
  xmlns:process=        "&grounding;#"
```

```
 xmlns          ="&DEFAULT;#"

 <owl:Ontology about="">
   <owl:versionInfo>$Id: ResultGrounding.owl,v 1.1 2020/10/16 085:15:00 martin Exp
$</owl:versionInfo>
   <rdfs:comment> Example for OWL-S release 1.0 Grounding description </rdfs:comment>
   <owl:imports rdf:resource="&service;" />
   <owl:imports rdf:resource="&grounding;" />
   <owl:imports rdf:resource="&r_service;" />
   <owl:imports rdf:resource="&r_process;" />
 </owl:Ontology>

<grounding:WsdlGrounding rdf:ID="Grounding_Résoudre une équation de 2ème degré a un seul
inconu">
  <service:supportedBy rdf:resource="&r_service;#Résoudre une équation de 2ème degré a un seul
inconuService"/>
  <grounding:hasAtomicProcessGrounding rdf:resource="#WsdlGrounding_Lecture des données a, b
,c"/>
  <grounding:hasAtomicProcessGrounding rdf:resource="#WsdlGrounding_Calculer Delta= b*b-
4ac"/>
  <grounding:hasAtomicProcessGrounding rdf:resource="#WsdlGrounding_Pas de solution"/>
  <grounding:hasAtomicProcessGrounding rdf:resource="#WsdlGrounding_Calculer X = -b / 2a"/>
  <grounding:hasAtomicProcessGrounding rdf:resource="#WsdlGrounding_Calculer R = racine de
Delta"/>
  <grounding:hasAtomicProcessGrounding rdf:resource="#WsdlGrounding_Calculer X1 = (-b-R) /
2a"/>
  <grounding:hasAtomicProcessGrounding rdf:resource="#WsdlGrounding_Calculer X2 = (-b+R) /
2a"/>
  <grounding:hasAtomicProcessGrounding rdf:resource="#WsdlGrounding_Afficher le résultat"/>
</grounding:WsdlGrounding>


<grounding:WsdlAtomicProcessGrounding rdf:ID="#WsdlGrounding_Lecture des données a, b ,c">
  <grounding:owlsProcess rdf:resource="&r_process;#Lecture des données a, b ,c"/>
  <grounding:wsdlOperation>
    <grounding:WsdlOperationRef rdf:ID="WSDLOperation_Lecture des données a, b ,c"/>
  <grounding:wsdlInputMessage>
  </grounding:wsdlInputMessage>
  <grounding:wsdlInputMessageParts rdf:parseType="Collection">
    <grounding:WsdlMessageMap>
      <grounding:wsdlMessagePart>
      </grounding:wsdlMessagePart>
    </grounding:WsdlMessageMap>
  </grounding:wsdlInputMessageParts>
  <grounding:wsdlOutputMessage>
  </grounding:wsdlOutputMessage>
  <grounding:wsdlOutputMessageParts rdf:parseType="Collection">
    <grounding:WsdlMessageMap>
      <grounding:wsdlMessagePart>
      </grounding:wsdlMessagePart>
    </grounding:WsdlMessageMap>
```

```
  </grounding:wsdlOutputMessageParts>
  <grounding:wsdlReference>
    <xsd:uriReference rdf:value="http://www.w3.org/TR/2001/NOTE-wsdl-20010315"/>
  </grounding:wsdlReference>
</grounding:WsdlAtomicProcessGrounding>

<grounding:WsdlAtomicProcessGrounding rdf:ID="#WsdlGrounding_Calculer Delta= b*b-4ac">
……————————————————————
</grounding:WsdlAtomicProcessGrounding>

<grounding:WsdlAtomicProcessGrounding rdf:ID="#WsdlGrounding_Pas de solution">
  ————————————————————————————————————————
</grounding:WsdlAtomicProcessGrounding>

<grounding:WsdlAtomicProcessGrounding rdf:ID="#WsdlGrounding_Calculer X = -b / 2a">
  ————————————————————————————.
</grounding:WsdlAtomicProcessGrounding>

<grounding:WsdlAtomicProcessGrounding rdf:ID="#WsdlGrounding_Calculer R = racine de Delta">
  ————————————————————————————————————————
</grounding:WsdlAtomicProcessGrounding>

 <grounding:WsdlAtomicProcessGrounding rdf:ID="#WsdlGrounding_Calculer X1 = (-b-R) / 2a">
  ————————————————————————————————————..
</grounding:WsdlAtomicProcessGrounding>

<grounding:WsdlAtomicProcessGrounding rdf:ID="#WsdlGrounding_Calculer X2 = (-b+R) / 2a">
  ————————————————————————————————————————————..
</grounding:WsdlAtomicProcessGrounding>

<grounding:WsdlAtomicProcessGrounding rdf:ID="#WsdlGrounding_Afficher le résultat">
  ————————————————————————————————————————————————————————————————————————————————
</grounding:WsdlAtomicProcessGrounding>
</rdf:RDF>
```

Tableau 4.2 : Tableau résultat d'exécution de l'exemple "Dia2owls".

4.5. Conclusion

Dans ce chapitre, nous avons vu comment l'implémentation de l'application qui fait la transformation automatique des diagrammes d'activité UML vers code OWL-S, en utilisant les outils tels que : Eclipse, Papyrus, Java, JDOM. Nous avons commencé par l'explication du principe de transformation, leurs phases, ses règles puis l'implémentation de notre application.

CONCLUSION GÉNÉRALE

Conclusion générale

Dans le contexte de notre travail, nous nous intéressons au domaine de l'ingénierie dirigée par les modèles.

L'objectif de ce travail est de proposer une approche automatique de transformation des diagrammes d'activités UML vers des ontologies OWL-S basée sur la transformation à base de visiteurs.

Nous avons commencé par une présentation des deux langages, UML comme un langage de modélisation Orienté Objet et OWL-S comme un langage de description des ontologies, nous avons fait un tour sur la transformation des modèles considérés comme le cœur de l'IDM, puis nous avons proposé des règles de transformation des diagrammes d'activité UML vers des ontologies OWL-S et on a terminé par l'implémentation de notre application.

Notre produit final est un Framework complet pour le développement automatique des ontologies OWL-S. Nous avons essayé de l'appliquer sur des exemples, et nous avons obtenu de bons résultats.

Dans un travail futur, nous proposerons de continuer la transformation des autres éléments avancés de diagramme d'activité, et cela pour couvrir plus de problèmes critiques. Nous comptons aussi d'intégrer d'autres diagrammes UML dans l'approche.

BIBLIOGRAPHIE ET WEBOGRAPHIE

Bibliographie et Webographie

[1] "***Débutez l'analyse logicielle avec UML : Les différents types de diagrammes*** " : https://openclassrooms.com/fr/courses/2035826-debutez-lanalyse-logicielle-avec-uml/2048781-les-differents-types-de-diagrammes. Consulté en Mars 2020.

[2] "***UML 2 - De l'apprentissage à la pratique"*** : https://laurent-audibert.developpez.com/cours- uml/. Consulté en Mars 2020.

[3] Bendiaf Masaoud. "***Spécification vérification des systèmes embarqués temps réel en utilisant la logique de réécriture"***. Thèse de doctorat, Université Mohamed Khider - Biskra, 2018.

[4] "***Modélisation UML - Introduction et concepts de base"*** : https://fr.m.wikiversity.org/wiki/Modélisation_UML/Introduction_et_concepts_de_base. Consulté en Mars 2020.

[5] Wafa L. Houda R. "***Une approche automatique de transformation des diagrammes d'activités UML vers les modèles EPC"***. Thèse de master, Université de Jijel, 2019.

[6] "***UML (informatique)"*** : https://fr.wikipedia.org/wiki/UML_(informatique). Consulté en Mars 2020.

[7] "***Qu'est-ce que le language UML (langage de modélisation unifié) ?"*** : https://www.lucidchart.com/pages/fr/langage-uml. Consulté en Mars 2020.

[8] "***Méthodologie des systèmes d'information – UML"*** : http://fdigallo.online.fr/cours/uml.pdf. Cours dispensé par Annick Lassus, CNAM ANGOULEME 2000-2001 .Consulté en Mars 2020.

[9] "***UML"*** : https://www.supinfo.com/articles/single/2834-uml#idm44858302319376 . Consulté en Mars 2020.

[10] Rima Saouchi, Hanane Boukerzaza. "***Conception et réalisation d'un site web dynamique pour un magazine en ligne"***, Thèse de licence ACAD, Université Mentouri Constantine, 2011.

[11] "***Unified Modeling Language (UML)"*** : https://www.techopedia.com/definition/3243/unified-modeling-language-uml. Consulté en Mars 2020.

[12] "***Conception et programmation orientées objet "*** : http://www-inf.it-sudparis.eu/COURS/CSC4002/EnLigne/Cours/CoursUML/3.4.html. Consulté en Mars 2020.

[13] Asma K. Meryem B. "***Une approche dirigée par les modèles basée sur UML pour la mise en œuvre de service de web composé"***. Thèse de master, Université de Djilali Bounàama Khemis Miliana, 2017.

[14] Joseph G. David G. "***UML 2 analyse et conception"***. Dunod, Paris, 2008.

[15] "***UML 2 - De l'apprentissage à la pratique/ 1. Chapitre 1 Introduction à la modélisation objet***": https://laurent-audibert.developpez.com/Cours-UML/?page=introduction-modelisation-objet#L1. Consulté en Mars 2020.

[16] Fien Van Der Heyde. Laurent Debrauwer. "***UML2"***, ENI 2éme éditions, mai 2010.

[17] "***Qu'est-ce qu'un diagramme d'activité UML ?"*** : https://www.lucidchart.com/pages/fr/diagramme-dactivite-uml. Consulté en Mars 2020.

[18] Houda Hamrouche. "***Une approche de transformation des diagrammes d'activité d'UML vers CSP basée sur la transformation de graphes"***. Thèse de Magister, Université de Skikda.

[19] "***Système"*** : https://fr.wikipedia.org/wiki/Système. Consulté en Avril 2020.

[20] Jamal Abd-Ali. "***Méta modélisation et transformation automatique de PSM dans une approche MDA"***. Thèse de M.Sc, Université du Québec en Outaouais, mai 2006.

[21] ***"Langage de modélisation"*** : https://fr.wikipedia.org/wiki/Langage_de_modélisation. Consulté en Avril 2020.

[22] "***Métamodèle"*** : https://fr.wikipedia.org/wiki/Métamodèle. Consulté en Avril 2020.

[23] Amour Nasreddine. "***Une approche MDA pour la transformation des diagrammes de classe UML vers une base de données basant sur l'outil EMF"***. Thèse de master, Université De M'sila, 2014.

[24] Combemale Benoit. "***Ingénierie dirigée par les modèles (IDM) – État de l'art".*** Institut de recherche en informatique de Toulouse, 2008, hal-00371565.

[25] Diaw Samba, Lbath Rédouane and Coulette Bernard. "***Etat de l'art sur le développement logiciel basé sur les transformations de modèles"***. Article dans Techniques et sciences informatiques, Mai 2010.

[26] Kerkouche El-hillali. "***Modélisation multi-paradigme : Une approche basée sur la transformation de graphes"***. Thèse de doctorat,Université Mentouri de Constantine, 2011.

[27] Xavier Blanc. "***MDA en action Ingénierie logicielle guidée par les modèles***". Eyrolles, 1ère édition, 2005.

[28] Sylvain Andre. "***MDA (Model Driven Architecture) principes et états de l'art"***. Thèse d'ingénieur, Centre D'enseignement De Lyon. 2004.

[29] "***Généralités sur l'approche MDA"*** : http://laine.developpez.com/tutoriels/alm/mda/generalites-approche-mda/. Consulté en Mars 2020.

[30] Ismail Bouacha. "***Une architecture fondée sur les services pour la prise en compte du contexte dans les systèmes d'information ubiquitaires"***. Thèse de Magister, Université Badji Mokhtar-Annaba, 2010.

[31] Kui Yu , Hai Xu , Cuiqiong Jiang , Gang Xue, Shaowen Yao, "***Using OWL-S for Formalizing Workflow Patterns"***, École des sciences et de l'ingénierie de l'information, Université du Yunnan, Kunming, Chine, 2e Conférence internationale sur les technologies de l'information et du multimédia, Singapour, 2010.

[32] Firas Bacha. "***Une approche MDA pour l'intégration de la personnalisation du contenu dans la conception et la génération des applications interactives"***. Thèse de doctorat, Université de Valenciennes et du Hainaut-Cambresis, Français, 2013.

[33] Boudia Malika. "***Transformation des diagrammes d'états-transitions vers maude"***. Thèse de master, Université de M'sila, 2011.

[34] "***Transformation de modèles*** " : https://fr.wikipedia.org/wiki/Transformation_de_modèles. Consulté en Mars 2020.

[35] Abdelali Elmounadi. "***Architecture dirigée par les modèles : Méthodes et outils de transformation de modèles"***. Conférence : La cinquième édition des Journées Doctorales en Technologies de l'Information et de la Communication (JDTIC'13). At: Kénitra - Maroc, Novembre 2013.

[36] "***MDA Model driven architecture"*** : http://*sylvain.andre.free.fr*/MDA. Consulté en Mars 2020.

[37] Med Saad Djabellah mahdi. "***Une approche de transformation de diagramme d'activité UML vers pi-calcule basée sur la transformation de graphe"***. Thèse de master, Université 20 Aout-1995-Skikda, 2015.

[38] Halim Bouchaboub. "***Transformation d'une ontologie OWL-S vers un réseau de petri"***. Thèse de master, Université d'Oum El Bouaghi, 2012.

[39] Laifa A. Hadef Kh. "***Un module de modélisation graphique pour les chaines de Markov basé sur l'IDM"***. Thèse de master, Université L'arbi Ben M'hidi d'Oum El Bouaghi, 2017.

[40] Lossan Bondé. "***Transformation de modèles et interopérabilité dans la conception de systèmes Hétérogènes sur Puce à base d'IP"***. Thèse de doctorat, Université de Lille, 2006.

[41] Mecheri Nacera. "***Une approche hybride pour transformer les modèles"***. Thèse de master, Université d'Oran, 27 octobre 2015.

[42] "***Web sémantique***" : https://www.futura-sciences.com/tech/definitions/internet-Web-semantique-3993/. Consulté en avril 2020.

[43] "***Web sémantique"*** : https://fr.wikipedia.org/wiki/Web_sémantique. Consulté en avril 2020.

[44] Chelbabi. M. "***Découverte de Services Web Sémantiques : une Approche basée sur le Contexte"***. Thèse de Master, CERIST, Algérie, 2006, N°9.

[45] Victor Moraru. Tran Vinh Duc. "***Réseau de Petri Superviseur"***. Institut de la Francophonie pour l'Informatique, Juillet 2005.

[46] Lassila O. Swick R. Eds***. "Resource Description Framework (RDF) Model and syntax specification"***. Recommandation du W3C du 22 février 1999.

[47] Bach Thành Lê. "***Construction d'un Web sémantique multi-points de vue"***. Thèse de doctorat, École des Mines de Paris à Sophia Antipolis, soutenue le 23 octobre 2006.

[48] "***Les langages de description des ontologies : RDF & OWL***" : https://www.researchgate.net/publication/324209660_Les_langages_de_description_des_ontologies_RDF_OWL. Consulté le 27/03/2020.

[49] "***A translation approach to portable ontology specifications"*** :

https://www.sciencedirect.com/science/article/abs/pii/S1042814383710083?via%3Dihub.

[50] "***Service Web"*** : https://fr.wikipedia.org/wiki/Service_Web. Consulté le 18/04/2020.

[51] Bouchakour Ibtissem. "***Reconfiguration dynamique d'un service Web composé basée sur les QdS"***. Thèse de master, Université Larbi ben mhidi d'Oum el bouaghi, 2018.

[52] "***Les Web Services"*** : http://deptinfo.unice.fr/twiki/pub/Linfo/Organisation%20Rapports/rapport-WebServices.pdf. Consulté le 15/03/2020.

[53] D. L. Martin. M. Paolucci S. A. et all. "***Bringing Semantics to Web Services: The OWL-S Approach"***. In J. Cardoso and A. P. Sheth, editors, SWSWPC, volume 3387 of Lecture Notes in Computer Science, 2004.

[54] "***OWL-S***" : https://slideplayer.fr/slide/12805119/#.X1FSHGpHuG9.gmail. Consulté le 20/05/2020.

[55] Hacine Gherbi Ahcine. "***Construction d'une Ontologie pour le WEB Sémantique"***. Thèse de Magistère, Université FERHAT ABBES –SÉTIF1, 2014.

[56] "***OWL-S : Semantic Markup for Web Services***" : https://www.w3.org/Submission/OWL-S/?fbclid=IwAR23B5FLdSLGMk11ed1_l6OUBlU5xShSjZr-OHOcb28b7Zndt3UZ6gr1oW4#acknowledgments. Consulté le 22/05/2020.

[57] Lezghed Amir. "***La Génération Automatique des Ontologies à partir des Diagrammes de classes UML"***. Thèse de master, Université de 8 Mai 1945 – Guelma, juin 2017.

[58] Amina Bekkouche. **"*Composition des Services Web Sémantiques à base d'Algorithmes Génétiques*"**. Thèse de Magistère, Université Abou-bekr Belkaid Tlemcen, 2012.

[59] Farida Goumeidane. "***Utilisation des technologies de services Web sémantiques dans le domaine médical***". Thèse de masters, Université Larbi ben mhidi d'Oum el bouaghi, 2013.

[60] Gayathri Nadarajan and Yun-Heh Chen-Burger, "***An Ontology-Based Conceptual Mapping Framework for Translating FBPML to the Web Services Ontology"*.** Centre pour les systèmes intelligents et leurs applications (CISA), école d'informatique, Université d'Edimbourg, R.U.

[61] Roy Grønmo, Michael C. Jaeger, and Hjørdis Hoff, "***Transformations between UML and OWL-S"***, SINTEF information et de la communication, N-0314 Oslo, Norvège Document de conférence Lecture Notes in Computer Science, Novembre 2005.

[62] Chong Wang, Keqing He, Yangfan He, and Wei Qian, "***Mappings from OWL-s to UML for Semantic Web Services"***, State Key Laboratoire de génie logiciel, Université de Wuhan, 430072, P.R.Chine.