ONTOLOGIES DES SERVICES WEB OWL-S

Au sommaire de ce chapitre :

Chapitre 02: Ontologies des Services Web OWL-S

2.1. Introduction

Le web sémantique est une nouvelle révolution dans le monde du Web pour que les informations et les données puissent être manipulées de manière logique par des programmes (des agents logiciels) et des machines, afin qu'elles transforment ces informations et données en un réseau significatif.

Les Services Web sont apparus pour rendre le Web plus dynamique. Ce sont des programmes informatiques permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués.

La fusion du Web sémantique avec le Service Web a donné naissance au Service Web Sémantique, cette nouvelle technologie permet aux agents logiciels de découvrir, sélectionner, appeler, composer, invoquer et exécuter automatiquement des services Web sans l'intervention de l'être humain.

Pour fournir une représentation de l'information sémantique, à la fois, détaillée, riche et facile à manipuler par les machines, les travaux de recherche menés autour de la description des services Web utilisent de plus en plus les ontologies. Les ontologies permettent d'améliorer la description et la découverte des services Web. Un des langages utilisé pour représenter ces ontologies est l'OWL-S.

Dans ce chapitre nous allons voir c'est quoi le web sémantique, les Services Web sémantiques et leurs ontologies OWL-S.

2.2. Web sémantique

2.2.1. Définition du Web sémantique

Tim Berners-Lee (inventeur du Web et directeur du W3C), a proclamé que le Web sémantique est la prochaine évolution du Web. Donc, il s'agit d'arriver à un Web intelligent, où les informations ne seraient plus stockées mais comprises par les machines afin d'apporter à l'utilisateur ce qu'il cherche vraiment [42].

Selon le W3C, le Web sémantique fournit un modèle qui permet aux données d'être partagées, intégrées et réutilisées entre plusieurs applications diverses, entreprises et groupes d'utilisateurs, et d'automatiser des requêtes pour aider leurs utilisateurs à créer de nouvelles connaissances [43].

L'objectif du Web sémantique est de lever les difficultés rencontrées sur le Web d'aujourd'hui (recherche d'information, services, ...), en rendant la grande masse d'information disponible, accessible et interprétable par les ordinateurs, en plus de l'automatisation de certaines fonctionnalités grâce à la représentation sémantique du contenu des documents, des services, des ressources sur le Web au sens large [44].

2.2.2. Architecture du Web sémantique

Nous allons présenter les trois niveaux importants du Web Sémantique : le niveau de l'adressage, le niveau syntaxique et le niveau sémantique.

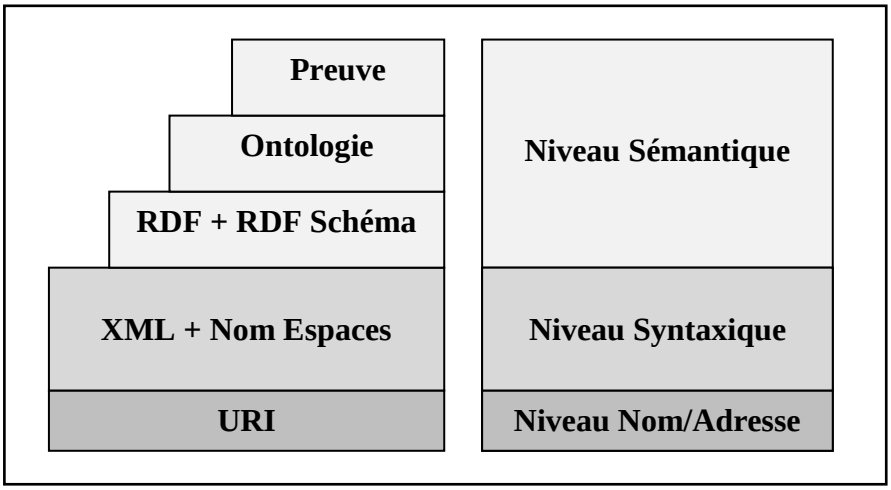

Figure 2.1 : Les trois niveaux du Web Sémantique [45].

- **Niveau «Nommage/Adressage»** : Tout ce qui est disponible sur Internet (ressource du Web soit page, adresse email, ou image, …) doit être identifié par un *URI* (Uniform Resource Identifier) de manière unique et non ambiguë.
- **Niveau Syntaxique** : C'est le niveau de la structuration des documents. La spécification de la structure logique des documents repose sur XML.
- **Niveau Sémantique**: Ce niveau contient les couches restantes, RDF, RDF Schéma, ontologie, cadre logique preuve.

2.2.3. Langages du Web sémantique

2.2.3.1. XML

XML est un métalangage permettant de définir d'autres langages de présentation de documents texte de manière arborescente en utilisant un système de balisage. Il est actuellement considéré comme un standard pour le transport de données sur le Web, et permet de stocker des données structurées dans un fichier texte [55].

2.2.3.2. RDF et RDFS

- **RDF (*Resource Description Framework*)** : RDF est un langage formel qui permet d'affirmer des relations entre des ressources. Ces ressources sont identifiées par les URI et déclarées sous forme « sujet – prédicat – objet », appelées des triplets [46].
- **RDF-S (*RDF Schéma*)** : est une extension sémantique de RDF. Il fournit des mécanismes pour décrire des groupes de ressources similaires et des relations entre ces ressources [47]. Il permet de spécifier des ontologies dites "légères" selon « Gandon et al. ».

2.2.3.3. DAML+OIL

DAML + OIL est un langage construit sur des normes précédentes du W3C telles que RDF et RDF Schéma, et étend ces langages avec des primitives de modélisation plus riches (l'intersection, l'union, la négation, la collection d'individus, la restriction sur l'application des propriétés et l'équivalence des ressources). C'est la fusion des deux langages DAML et OIL [55].

2.2.3.4. OWL (Ontology Web Langage)

OWL est un langage standard du W3C pour la représentation des ontologies, on en revient plus tard.

2.2.4. Les ontologies

2.2.4.1. Définition

Ontologie est un terme emprunté à la philosophie qui implique une branche de la philosophie qui traite la nature et l'organisation de la réalité.

Une ontologie est une spécification formelle, et explicite d'une conceptualisation partagée[49].

Aussenac et Gilles en 2000 ont proposé la définition : « Une ontologie organise dans un réseau des concepts représentant un domaine. Son contenu et son degré de formalisation sont choisis en fonction d'une application ».

2.2.4.2. Composants de l'ontologie

Les constituants de base principaux qui aident la portée des connaissances traduites par une ontologie sont : concepts, relations, fonctions, axiomes, instances [48].

- ***Concept*** : C'est la représentation abstraite des éléments (termes ou classes) du domaine.
- ***Relations*** : Elles expriment les associations entre les différents concepts.
- ***Fonctions*** : C'est un héritage de la relation, qui définit le nième élément de la relation en fonction de n-1 éléments précédents.
- ***Axiomes*** : Constituent des assertions considérées toujours comme vraies.
- ***Instances*** : Ce sont des exemples particuliers de concepts.

2.2.4.3. OWL

OWL est, comme RDF, un langage profitant de l'universalité syntaxique de XML, fondé sur la syntaxe de RDF/XML. Elle offre un moyen de décrire des ontologies Web (langage d'ontologies). OWL intègre des outils de comparaison des propriétés et des classes (identité,

équivalence, contraire, cardinalité, symétrie, transitivité, disjonction, etc.). Ainsi, OWL offre aux machines une plus grande capacité d'interprétation du contenu Web grâce à un vocabulaire plus large et à une vraie sémantique formelle [48].

OWL se compose de trois sous langages d'expressivité croissante : OWL Lite, OWL DL (*Description Logique*) et OWL Full.

- ✓ ***OWL-Lite*** : le plus simple, qui permet la création des hiérarchies de classification et l'expression de contraintes simples.
- ✓ ***OWL-DL*** : a été proposé afin de fournir plus d'expressivité tout en restant complet et décidable.
- ✓ ***OWL-Full*** : il n'assure pas la décidabilité, permet d'enrichir le vocabulaire prédéfini et garantie une expressivité maximale.

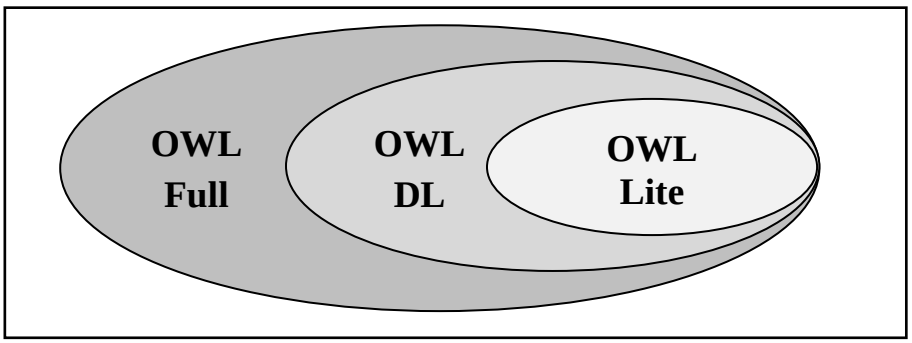

Figure 2.2 : Les trois sous langage d'OWL.

2.3. Services Web sémantique

2.3.1. Services Web

2.3.1.1. Définition du service Web

Un service Web est un programme informatique identifié par une *URI*, permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués (diverses plates-formes). Donc, il s'agit d'un ensemble de fonctionnalités exposées sur le Web (internet ou intranet), par et pour des applications ou machines, sans intervention humaine, de façon synchrone ou asynchrone [50].

Les services Web sont décrits par des documents *WSDL* (*Web Service Description Language),* qui précisent les méthodes pouvant être invoquées, leurs signatures et les points d'accès

du service (URL, port). Ils sont accessibles via *SOAP* (*Simple Object Access Protocol*), la requête et les réponses sont des messages XML transportés sur *HTTP* (*HyperText Transfer Protocol*). *UDDI* (*Universal, Description, Discovery and Integration*) est une réponse plus adaptée pour enregistrer les informations de fournisseurs de services Web et leurs services [50].

La figure 2.3 présente les technologies des Services Web et leurs relations.

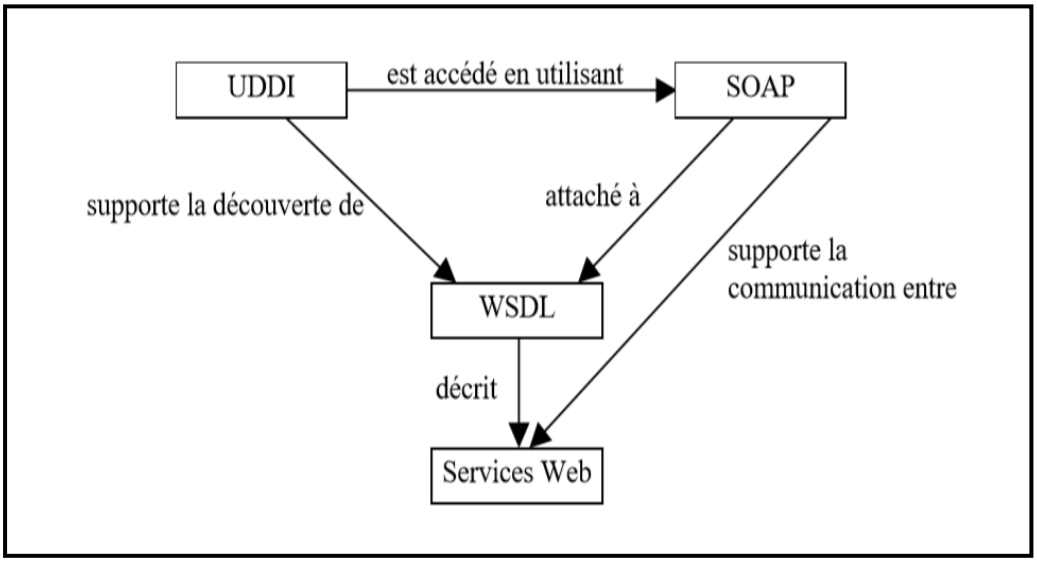

Figure 2.3 : Les principales technologies des Services Web.

2.3.1.2. Caractéristiques des services Web

Ils sont caractérisés par [51] :

- ✓ Ses descriptions sont stockées dans un annuaire ;
- ✓ Ils peuvent fonctionner indépendamment les uns des autres, ils interagissent par le biais de messages asynchrones basés sur XML, ce qui favorise la réduction des dépendances entre eux, ces messages sont transportés par des protocoles Internet (généralement HTTP);
- ✓ Ils fonctionnent de manière modulaire, Cela implique une flexibilité d'intégration et d'interopération entre ces modules ;
- ✓ Ils sont des composants logiciel avec possibilité de réutilisation et de composition pour construire de nouveaux services, de manière qu'un service peut agir dans plusieurs compositions ;
- ✓ Ils permettent de réduire les coûts des communications.

2.3.1.3. Fonctionnement des services Web

Le fonctionnement des services Web s'articule autour de trois acteurs principaux illustrés par la figure 2.4 suivante :

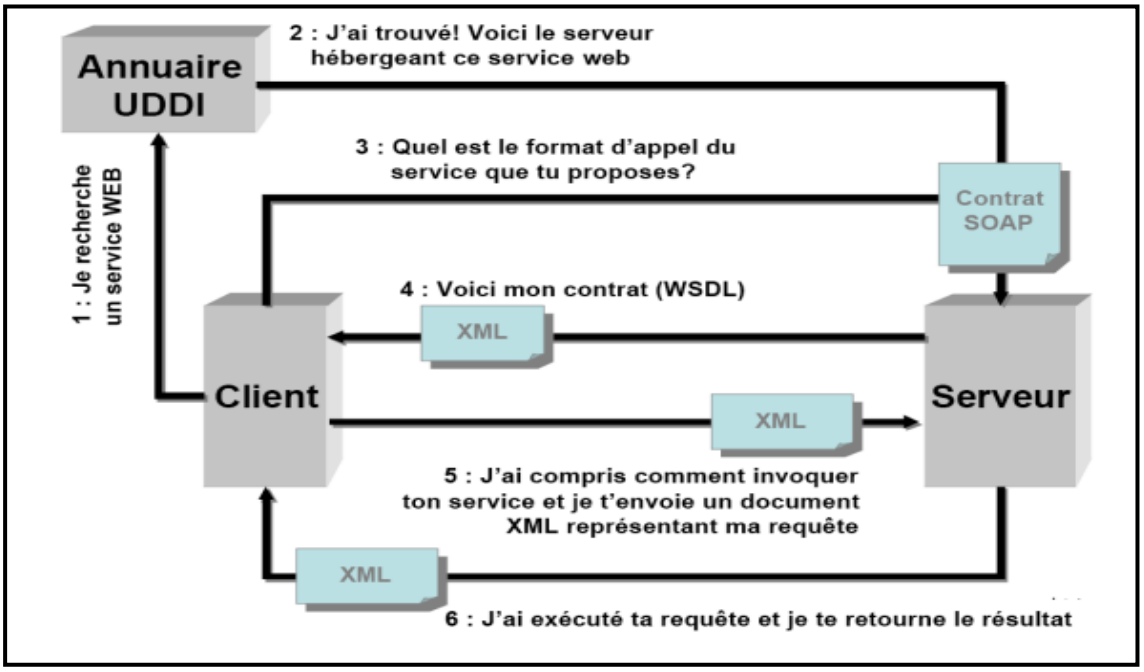

Figure 2.4 : Fonctionnement des services Web [52].

Les trois éléments les plus importants des services Web sont les fournisseurs de service, les annuaires de services et les consommateurs de service.

Cette architecture fonctionne de la manière suivante [52]:

1. Le client envoie une requête à l'annuaire de Service pour trouver le service Web dont il a besoin.
2. L'annuaire cherche pour le client, trouve le service Web approprié et renvoie une réponse au client en lui indiquant quel serveur détient ce qu'il recherche.
3. Le client envoie une deuxième requête au serveur pour obtenir le contrat de normalisation de ses données.
4. Le serveur envoie sa réponse sous la forme établie par WSDL en langage XML.
5. Le client peut maintenant rédiger sa requête pour traiter les données dont il a besoin.
6. Le serveur fait les calculs nécessaires suite à la requête du client, et renvoie sa réponse sous la même forme normalisée.

2.3.2. Du service Web au service Web sémantique

La découverte des services Web est faite au niveau du registre UDDI, elle est basée essentiellement sur la recherche syntaxique des descriptions WSDL des services Web. Les chercheurs exploitent les technologies du Web sémantique (*RDF*, *RDFS*, *OWL*), afin d'enrichir les services Web de descriptions sémantique. Alors, la combinaison des technologies des services Web et du Web sémantique a mené au concept des services Web sémantiques.

Donc, les services Web sémantique sont des services Web dotés de descriptions sémantiques. Cette sémantique est apportée grâce aux ontologies.

2.3.3. Services Web sémantique

2.3.3.1. Définition du service Web sémantique

Les services Web sémantiques sont la convergence de deux technologies de l'Internet : les services Web et le Web sémantique, ce sont des services Web à descriptions dotées de sémantique. Donc, ils prolongent les capacités d'un Web service en associant des concepts sémantiques au Web service interprétables par l'agent logiciel afin de permettre une meilleure recherche, découverte, choix, composition et intégration [44].

2.3.3.2. Langages des services Web sémantique

- ***WSDL* :** C'est un langage reposant sur la notation XML permettant de décrire les services Web, leurs emplacements ainsi que les opérations (méthodes, paramètres et valeurs de retour) que le service propose.
- ***SAWSDL (Semantic Annotations for WSDL and XML Schema)* :** C'est un langage sémantique de description de service Web. Il est évolutif et compatible avec les standards des services Web existants, et plus spécifiquement avec WSDL.
- ***WSMO (Web Service Modeling Ontology)*** : C'est un langage formel et une ontologie pour la description de divers aspects liés aux Web services sémantiques.
- ***OWL-S* :** C'est un langage permettant de décrire les services Web de façon non ambiguë et interprétable par des programmes. Ce langage est basé sur le langage d'ontologie du Web (*OWL*).

2.4. Ontologies des Services Web OWL-S

2.4.1. Définition

OWL-S est une ontologie construite au-dessus du langage d'ontologie Web (*OWL*) par le programme DARPA DAML. Il remplace l'ancienne ontologie DAML-S. Elle permet aux utilisateurs et aux agents logiciels de découvrir, invoquer, composer et surveiller automatiquement les ressources Web offrant des services, sous des contraintes spécifiées [53].

2.4.2. Composants d'ontologie OWL-S

OWL-S organise une description de service en quatre secteurs conceptuels: le «process model», le «profile», le «grounding», et le «service».

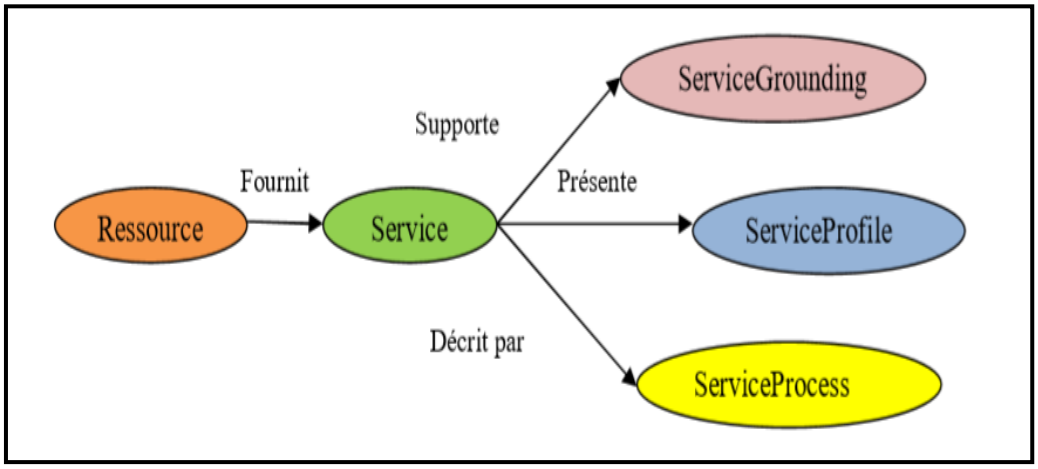

Figure 2.5 : Ontologie OWL-S [53].

- **Un modèle de processus (process model)** : Il décrit comment un service accomplit ses tâches. Il inclut des informations sur les entrées, les sorties, les conditions préalables et les résultats. Le modèle de processus a trois types: composés, atomiques, et simples.
  - ✓ ***Un processus atomique (Atomic Process)*** : Il correspond à une action que le service peut effectuer en une seule interaction avec le client.
  - ✓ ***Les processus composés (Composite Process)*** : Ils sont décomposables en d'autres processus (composés ou non) ; en utilisant un ensemble de structures de contrôles tels que : Sequence, Split, If-Then-Else, choice, repeat, etc.
  - ✓ ***Un processus simple (Simple Process)*** : Il est utilisé pour fournir une vue d'un processus atomique ou une représentation simplifiée d'un processus composite.

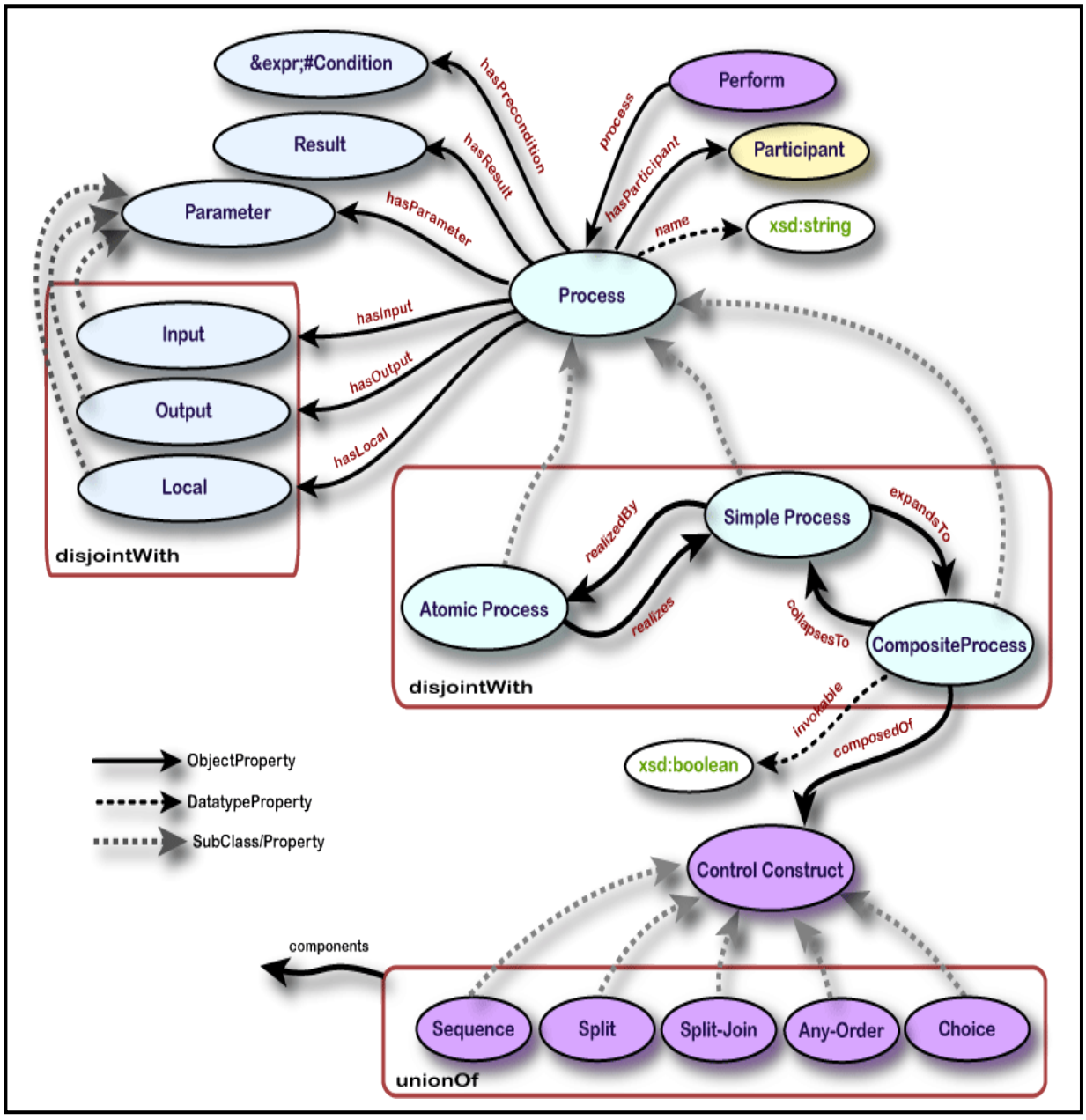

Figure 2.6 : L'ontologie de Process Model [56].

- **Un profil (profile)** : Il fournit une description générale de Web Service pour qu'il puisse être publié et partagé pour faciliter la découverte de service. Les profils peuvent inclure des propriétés fonctionnelles (entrées, sorties, les prés conditions et les post conditions) et des propriétés non fonctionnelles (nom de service, description des textes, information de contact, catégorie de service et paramètres additionnels de service (comme la qualité de service Web QoS)) [53].

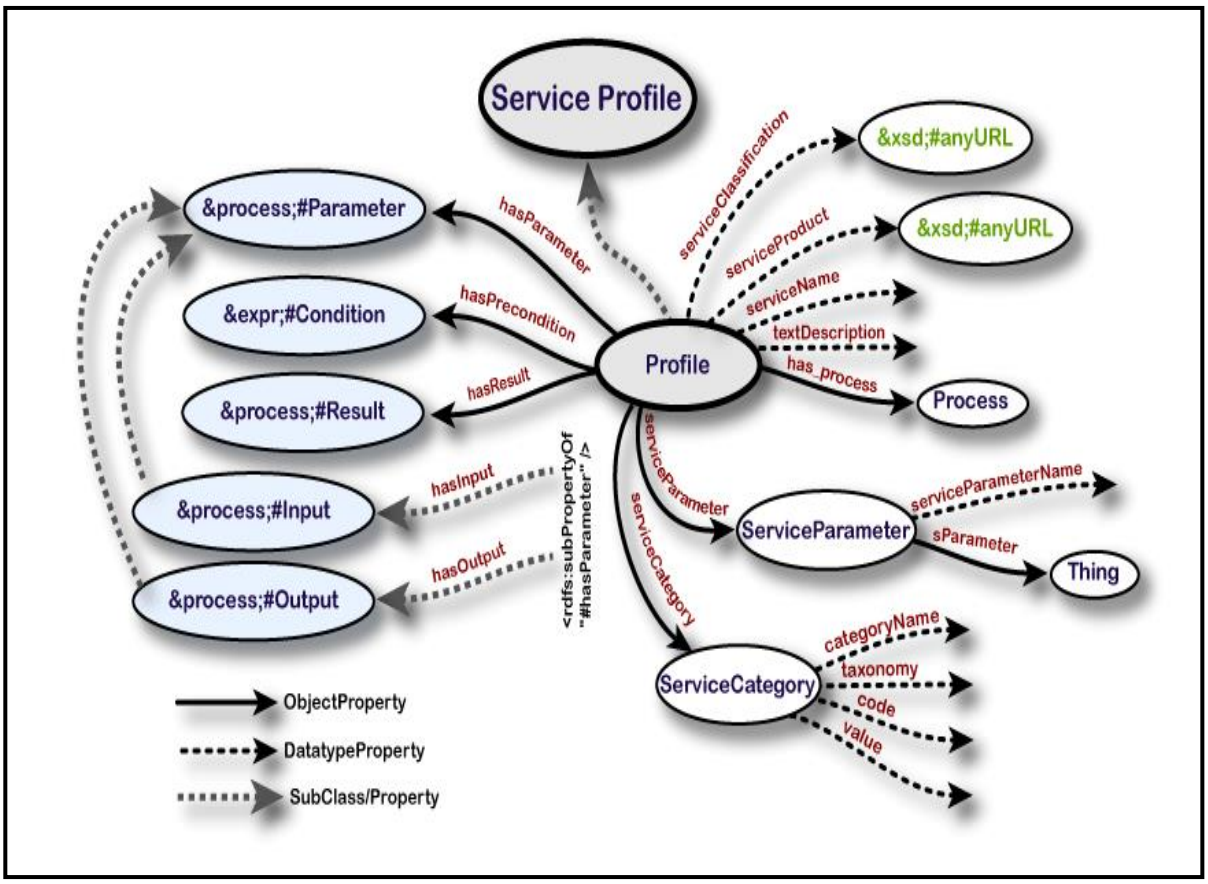

Figure 2.7 : Les classes et propriétés de service profile [56].

- **Le sol de service (grounding)** : Il indique comment accéder concrètement au service et fournit les détails concernant les protocoles, les formats de messages et les adresses physiques, etc. OWL-S a différents types de sol de service à utiliser, mais le seul type développé jusqu'ici est le WSDL grounding, qui permet à tous les Web services d'être marqués comme un Service Web Sémantique en utilisant OWL-S [53].

Le tableau suivant résume les composants d'OWL-S [38] :

Service Profile	ProcessModel	ServiceGrounding
-Présenté par un service -Représenter ce que le service fournit	-Décrit comment ce service fonctionne: les processus internes du service -Spécifie le protocole	-Fournit des spécifications d'information d'accès de service -Service Model+Grounding

-Usage Principal: +Publicité des capacités +Demande de Web service qui a donné un ensemble de sa capacité	d'interaction de service -Spécifie les messages abstraits: l'information transmis sur type d'ontologie -Faciliter l'invocation et la composition des services, la surveillance de l'interaction	donne tous les choses nécessaire pour utiliser ce service -Construire sur WSDL pour définir la structure de message et la couche physique «binding» -Spécifier le protocole communication, mécanismes de transport, langages de communication, etc...

Tableau 2.1 : Composants d'OWL-S.

2.4.3. Objectifs d'OWL-S

OWL-S a plusieurs objectifs [38]:

- Découverte automatique des services Web par des agents logiciels.
- Détection et interprétation automatique des paramètres d'entrée-sortie nécessaires à chaque service Web et d'appeler le service.
- Composition et interopération automatiques des services Web.
- Surveillance automatique: des agents surveilleront l'exécution des processus invoqués pour chaque transaction et informeront l'utilisateur de l'évolution de la transaction en temps réel.

2.4.4. Exemple

Prenant cet exemple qui montre un service web qui fait un rapport de comparaison entre le café et thé, l'exemple est tiré de [54]

Service Profile :

Ce profil "COFFEETEAREPORT_PROFILE" inclure des propriétés fonctionnelles (sorties : café, thé et rapport) et des propriétés non fonctionnelles (nom de service : TeaCoffeeDifferencesService, description des textes : Ce service informe pour certains thé et café et rapports de comparaison de leurs avantages et inconvénients, process: COFFEETEAREPORT_PROCESS").

```
<profile:Profile rdf:ID="_COFFEETEAREPORT_PROFILE">
        <service:isPresentedBy rdf:resource="#_COFFEETEAREPORT_SERVICE"/>
        <profile:serviceName xml:lang="en"> TeaCoffeeDifferencesService
        </profile:serviceName>
        <profile:textDescription xml:lang="en">
        Ce service informe pour certains thé et café et rapports de comparaison de leurs
        avantages et inconvénients
        </profile:textDescription>
        <profile:hasOutput rdf:resource="#_COFFEE"/>
        <profile:hasOutput rdf:resource="#_TEA"/>
        <profile:hasOutput rdf:resource="#_REPORT"/>
        <profile:has_process rdf:resource="_COFFEETEAREPORT_PROCESS"/>
</profile:Profile>
```

ProcessModel :

Ce ProcessModel "COFFEETEAREPORT_PROCESS_MODEL" a un process atomique "COFFEETEAREPORT_PROCESS". Ce dernier a des sorties (café, thé et rapport).

```
<process:ProcessModel rdf:ID="_COFFEETEAREPORT_PROCESS_MODEL">
    <service:describes rdf:resource="#_COFFEETEAREPORT_SERVICE"/>
```

<process:hasProcess rdf:resource="#_COFFEETEAREPORT_PROCESS"/>
</process:ProcessModel>
<process:AtomicProcess rdf:ID="_COFFEETEAREPORT_PROCESS">
<process:hasOutput rdf:resource="#_COFFEE"/>
<process:hasOutput rdf:resource="#_TEA"/>
<process:hasOutput rdf:resource="#_REPORT"/>
</process:AtomicProcess>
<process:Output rdf:ID="_COFFEE">
<process:parameterType>http://127.0.0.1/ontology/Mid-level-ontology.owl#Coffee
</process:parameterType>
</process:Output>

ServiceGrounding :

La localisation de la description WSDL de ce service est contenue dans :" http://www.webservicex.com/globalweather.asmx?wsdl#"

<grounding:wsdlDocument rdf :datatype=’&xsd ; #anyURI’>
http://www.webservicex.com/globalweather.asmx?wsdl#
</grounding:wsdlDocument>

Tableau 2.2 : Exemple d’OWL-S.

2.5. Conclusion

Le web sémantique n’est pas pour l’avenir puisque il existe déjà, son but est de rendre le web actuel aussi utilisable par les humains que par les machines. Et c’est ce que nous avons vu dans notre chapitre.

Nous avons présenté le web sémantique et ses langages et ontologies, et on a terminé par les services web sémantiques et l’ontologie OWL-S.

Dans le chapitre suivant, nous allons aborder les approches de transformation de modèles et nous allons commencer notre transformation des diagrammes d'activité UML vers OWL-S.

Chapitre 3

LA TRANSFORMATION DE MODÈLE

Au sommaire de ce chapitre :

1. *Introduction.*
2. *IDM*
3. *L'approche MDA.*
4. *La transformation de modèles*
5. *Conclusion.*

Chapitre 03 : La transformation de modèles

3.1. Introduction

L'architecture dirigée par les modèles MDA est une démarche de réalisation de logiciel. C'est une variante particulière de l'ingénierie dirigée par les modèles IDM. Tout comme les méta-modèles, les transformations de modèles sont un concept central de l'ingénierie dirigée par les modèles, qui sont des programmes permettant de générer automatiquement et de modifier des modèles. Il existe plusieurs standards de transformations de modèles comme *QVT* (Query, View, Transformation) ou MOFM2T ainsi que de nombreux langages de transformation de modèles comme ATL.

Dans ce chapitre, nous allons aborder les principes fondamentaux de IDM et MDA, en suite nous exposons les concepts de transformation des modèles, ses types, son objectif et on termine par la classification des approches de transformation de modèles.

3.2. Ingénierie Dirigée par les Modèle (IDM)

3.2.1. Présentation du l'IDM

Dans les années 80, il apparut le principe de l'approche objet est du "tout est objet". Mais aujourd'hui, l'ingénierie du logiciel s'oriente vers l'ingénierie dirigée par les modèles (*IDM*) avec le principe du "tout est modèle".

L'*IDM* (en Anglais *MDE* : Model Driven Engineering) est une approche de développement logiciel, se base sur l'utilisation des modèles comme des éléments centraux tout au long du cycle de vie du logiciel. Il est nécessaire de formaliser des modèles précis et riches pour les rendre exploitables par les machines des programmes permettant de traiter des modèles.

3.2.2 Concepts de base de l'IDM

- **Système** : C'est l'ensemble des éléments relient les uns aux autres selon certains principes ou règles [19]. Il faut faire une abstraction du système à l'aide d'un modèle.
- **Modèle** : Un modèle est une abstraction de la réalité ou bien une représentation simplifié, qui permet de mieux comprendre le processus de développement d'un système. Son but est de modéliser et faciliter le traitement du système à utiliser. La notion de modèle dans l'IDM fait explicitement référencer à la notion de langage de modélisation [20].
- **Langage de modélisation** : Un langage de modélisation est un langage artificiel qui peut être utilisé pour exprimer de l'information ou de la connaissance ou des systèmes dans une structure qui est définie par un ensemble cohérent de règles. Les règles sont utilisées pour l'interprétation de la signification des composants dans la structure [21].
- **Méta-modèle** : Un méta-modèle est un modèle qui décrit la structure de modèles. Il est une abstraction permettant de décrire des modèles. il permet la construction de langages de modélisation, la création de relations entre les modèles et la définition de règles de modélisation. On dit que le méta-modèle représente le modèle, tandis que le modèle instancie le méta-modèle [22]. Il y a Quatre aspects pour la spécification d'un méta-modèle :
  - ***La syntaxe abstraite*** : Elle décrit les constructions qui composent ses modèles.
  - ***La syntaxe concrète*** : Elle décrit la représentation des constructions définies par sa syntaxe abstraite, qui peut être graphique, textuelle ou mixte. À chaque syntaxe abstraite correspondent une ou plusieurs syntaxes concrètes.
  - ***La sémantique statique (ou structurel)*** : Elle décrit les critères et les règles de modélisation qui ne peuvent pas être représentés par la syntaxe abstraite.
  - ***La sémantique dynamique*** : C'est une description du sens qui lui est donné. Elle est exprimée à l'aide d'un langage naturel et rarement par le langage formel.
- **Méta-méta-modèle** : C'est un modèle qui décrit un langage de méta-modélisation, c.-à-d. les éléments de modélisation nécessaires à la définition des langages de modélisation. Il a de plus la capacité de se décrire lui-même [23].
- **Méta-modélisation** : La méta-modélisation est l'activité consistant à définir le méta-modèle d'un langage de modélisation. Elle vise donc à bien modéliser un langage, qui joue alors le rôle de système à modéliser [24].

- **Transformation de modèle** : C'est la génération automatique d'un modèle cible à partir d'un modèle source, suivant une définition de la transformation.
- **Plateformes** : Une plate-forme est une base de travail à partir de laquelle on peut écrire, lire, développer et utiliser un ensemble de logiciels. Elle peut être composée du : matériels, système d'exploitation, outils logiciels, gestion de projet, SGBD, serveur web, serveur d'applications, …

3.3. L'approche MDA (Model Driven Architecture)

3.3.1. Principe de MDA

En novembre 2000, l'OMG initie la démarche MDA. C'est une proposition à la fois d'une architecture et d'une démarche de développement.

MDA est venu comme solution de problème de portabilité, d'interopérabilité et ainsi l'évolutivité et la documentation pour augmenter la productivité dans le processus de développement logiciel. On peut conclure que la technologie MDA est entièrement basée sur les modèles et leurs transformations [26].

L'objectif majeur de MDA est l'élaboration de modèles pérennes, indépendants des détails techniques des plateformes d'exécution (J2EE, .Net, PHP ou autres), afin de permettre la génération automatique de la totalité du code des applications et d'obtenir un gain significatif de productivité [27].

3.3.2. Les standards du MDA

L'approche MDA basée sur un ensemble de standards de l'OMG, la figure 3.1 ci-dessous montre ces standards [28].

- ✓ Au centre, se trouve les standards : UML, MOF et CWM.
- ✓ Dans la couche suivante, se trouve aussi un standard XMI qui permet le dialogue entre les middlewares (Java, CORBA, .NET et web services).
- ✓ La troisième couche contient les services qui permettent de gérer les évènements, la sécurité, les répertoires et les transactions.

- ✓ Enfin, la dernière couche propose des frameworks spécifiques au domaine d'application (Finance, Télécommunication, Transport, Espace, médecine, commerce électronique, manufacture,…).

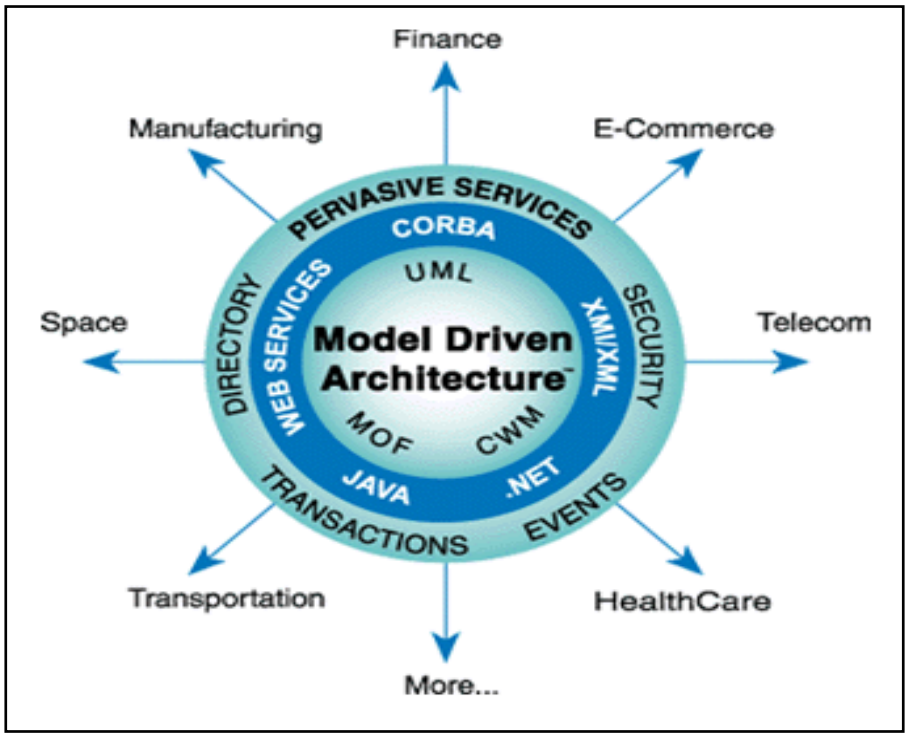

Figure 3.1 : Les standards du MDA.

- **MOF (Méta Object Facility)**

Se situe au sommet dans l'architecture à quatre niveaux de l'OMG (voir figure 3.5 ci-dessus). C'est un standard de l'OMG s'intéressant à la représentation des méta-modèles et leur manipulation. MOF est un ensemble des interfaces standards qui permet de définir la syntaxe et la sémantique d'un langage de modélisation.

- **UML (Unified Modeling Language)**

C'est un langage de modélisation normalisé semi-formel qui permet de modéliser les systèmes à travers des diagrammes et des textes. Il est couramment utilisé en développement logiciel et en conception orientée objet.

- **XMI (XML Meta data Interchange)**

C'est un standard pour l'échange d'informations de métadonnées UML basé sur XML, qui offre une représentation concrète des modèles sous forme de documents XML.

- **OCL (Object Constraint Language)**

C'est un langage informatique d'expression de contraintes permet de décrire n'importe quelle contrainte sur des modèles. Ce langage formel est volontairement simple d'accès et représente un juste milieu entre langage naturel et langage mathématique.

3.3.3. Les modèles de MDA

Le MDA offre un moyen d'utiliser des modèles au lieu du code source traditionnelle. Pour cela, le MDA classe les modèles en modèles indépendants des plates-formes appelés *PIM* (Platform-Independent Models), et en modèles spécifiques appelés *PSM* (Platform-Specific Models). L'approche MDA permet de déployer un même modèle de type PIM sur plusieurs plates-formes (modèles *PSM*) grâce à des projections standardisées [25].

3.3.3.1. Modèle d'exigence CIM (Computation Independent Model) : C'est le modèle métier ou le modèle du domaine d'application qui décrit les besoins fonctionnels de l'application, aussi bien les services qu'elle offre que les entités avec lesquelles elle interagit. Il est indépendant de tout système informatique [29].

3.3.3.2. Le PIM (Platform Independent Model) : C'est un modèle d'analyse et de conception de l'application. Il représente une vue partielle d'un CIM, le logique métier spécifique au système ou le modèle de conception, et aussi le fonctionnement des entités et des services. Il est indépendant de toute plate-forme technique [30].

3.3.3.3. Le PSM (Platform Specifique Model) : C'est le modèle qui se rapproche le plus du code final de l'application et il décrit l'implémentation d'une application sur une plateforme particulière et spécifiée par l'architecte, il est donc lié à une plateforme d'exécution [29].

3.3.3.4. Le PDM (Plateform Description Model) : Cette notion n'est pas encore bien définie par l'OMG, pour l'instant il s'agit plus d'une piste de recherche, c'est pour ça il est rarement utilisé. Un PDM contient des informations pour la transformation de modèles vers une plate-forme en particulier et il est spécifique de celle-ci [27].

3.3.4. Transformation de modèles

3.3.4.1. Définition de transformation

C'est un ensemble de règles de transformation qui décrivent ensemble comment un modèle dans un langage source est transformé en modèle dans un langage cible.

La transformation est exécutée à l'aide d'un moteur de transformation qui lit le modèle source qui conforme à un méta-modèle source et produit en sortie un modèle cible conforme à un méta-modèle cible [32].

Supposons que : ***T*** est la transformation de modèle par la fonction, ***S*** le modèle source, ***C*** le modèle cible, ***m*** et ***ḿ*** *sont* des méta-modèles pour les modèles ***S*** et ***C*** respectivement. On peut définit ***T*** par : ***T***(***S***) → ***C***, c.à.d. à partir d'un (ou un ensemble des) modèle(s) source(s) on peut créer un (ou un ensemble des) modèle(s) cible(s). Si ***m*** = ***ḿ*** alors la fonction ***T*** est endogène sinon elle est exogène [33].

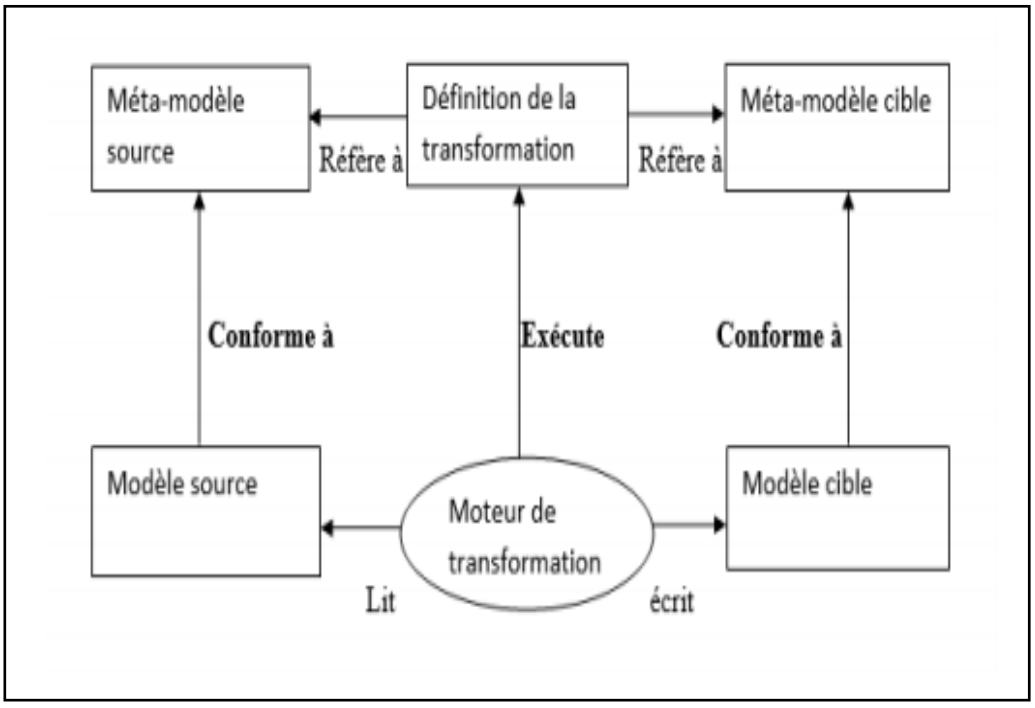

Figure 3.2 : Principe d'une transformation de modèles méta modèles [33].

3.3.4.2. Types de transformation

On peut distinguer plusieurs types de transformation de modèle [34] :

- **Selon la nature de méta-modèle**

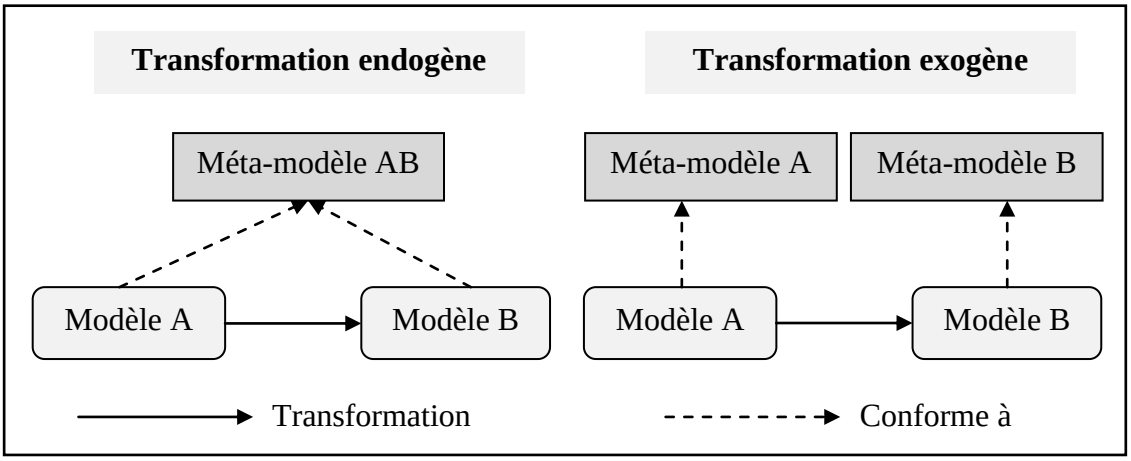

Figure 3.3 : La transformation endogène et exogène [35].

- ✓ ***Transformations endogènes* :** lorsque son modèle source et son modèle cible instancient le même méta-modèle.
- ✓ ***Transformations exogènes* :** une transformation est dite exogène lorsque son modèle source et son modèle cible n'instancient pas le même méta-modèle.

- **Selon la base de niveau d'abstraction**

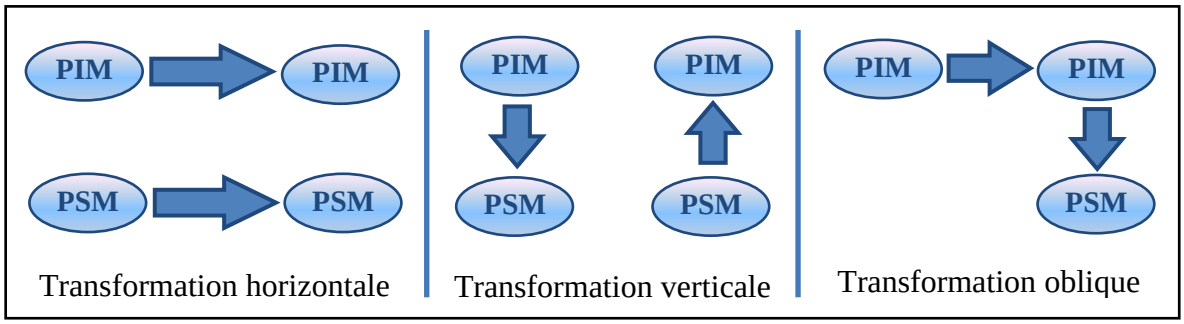

Figure 3.4 : La transformation horizontale et verticale.

- ✓ ***Transformations horizontales* :** Une transformation est dite horizontale lorsque son modèle source et son modèle cible se trouvent au même niveau d'abstraction.
- ✓ ***Transformations verticales* :** une transformation est dite verticale lorsqu'elle opère à des niveaux d'abstraction différents. Le raffinement est une transformation verticale.

- ✓ ***Transformations Oblique :*** Une transformation oblique combine une transformation horizontale et une verticale [33].

- **Selon la base de la structure**
  - ✓ ***Transformations syntaxiques*** **:** Une transformation est dite syntaxique lorsqu'elle n'utilise que la syntaxe du ou des modèles source.
  - ✓ ***Transformations sémantiques*** **:** La transformation est dite sémantique, lorsque des traitements plus complexes prenant en compte la sémantique des modèles sont effectués.
- **Selon la base de sens**
  - ✓ ***Transformations bidirectionnelles*** **:** Une transformation est dite bidirectionnelle si tous les modèles qu'elle comprend peuvent être à la fois des modèles source et cible.
  - ✓ ***Transformations unidirectionnelles*** **:** Dans le cas contraire, la transformation est dite unidirectionnelle.

3.3.5. L'architecture à quatre niveaux

La figure 3.5 ci-dessous représente l'architecture qui est hiérarchisée en quatre niveaux. En partant du bas [23] :

- ***Le niveau M0 (ou instance)*** **:** Ce niveau correspond au monde réel. Il définit les informations réelles de l'utilisateur, instance du modèle de M1.
- ***Le niveau M1 (ou modèle)*** **:** Ce niveau est composé de modèles d'information. Il représente toutes les instances d'un méta-modèle de M2.
- ***Le niveau M2 (ou méta-modèle)*** **:** Ce niveau représente toutes les instances d'un méta-méta-modèle. Il définit le langage de modélisation et la grammaire de représentation des modèles M1.
- ***Le niveau M3 (ou méta-méta-modèle)*** **:** Ce niveau définit un langage unique pour la spécification des Méta-modèles qui s'appelle le MOF. Ce dernier permet de décrire la structure des méta-modèles, d'étendre ou de modifier les méta-modèles existants. Le MOF est réflexif puisqu'il se décrit lui-même.

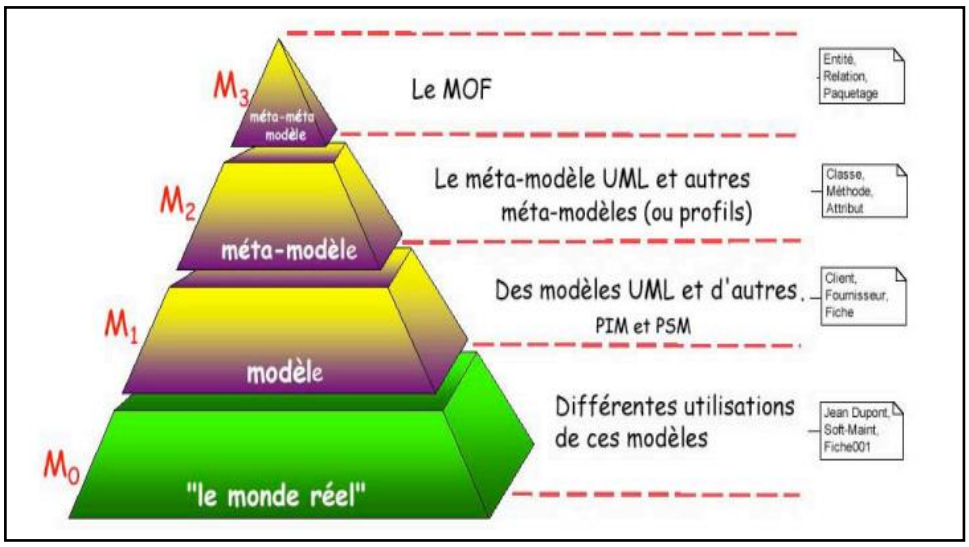

Figure 3.5 : Architecture à quatre niveaux [36].

3.3.6. Processus de transformation de modèles selon MDA

Il existe plusieurs types de transformation de modèles que l'on peut trouver dans l'approche MDA, nous allons les illustrer dans la figure 3.6 [37].

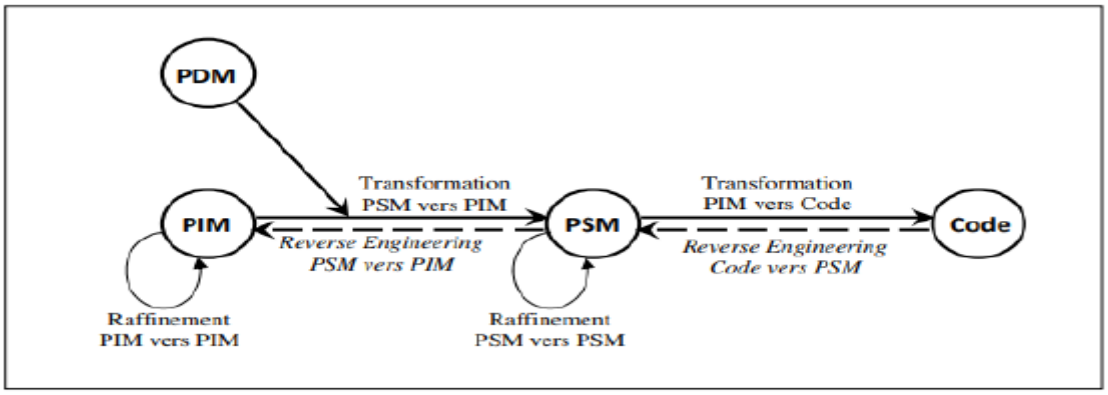

Figure 3.6 : les modèles et la transformation dans l'approche MDA.

- **Transformation PIM vers PIM et PSM vers PSM :** On utilise ce type de transformation pour enrichir, filtrer ou spécialiser les informations de modèle, ces transformations utilisées pour le raffinement de modèle [38].
- **Transformation PIM vers PSM :** On utilise ce type de transformation si le PIM est suffisamment raffiné. Cette transformation permet d'ajouter au PIM des informations propres à une plate-forme technique c'est le PDM [30].

- **Transformation PSM vers Code :** Cette transformation est de type modèle à code, tantôt le code est assimilé à un PSM exécutable généralement n'est pas possible d'obtenir la totalité du code à partir de modèle, alors il est besoin de la compléter manuellement [23].
- **Transformation PSM vers PIM et Code vers PSM :** Ces transformations sont des transformations les plus difficile complexe à réaliser, et qui s'appel des opérations de rétro-ingénierie. Il faut faire un appel aux techniques traditionnelles de rétro-ingénierie si le code n'a pas conçu dans la démarcha de MDA pour pouvoir effectuer telles opérations [26].

3.3.7. Propriétés de transformation de modèles

La transformation de modèles est caractérisée par des propriétés, nous allons illustrons dans le tableau 3.1 [30] [3] [40].

Propriétés	Significations
Traçabilité	C'est la propriété qui permet d'enregistrer des liens entre les éléments des deux modèles (source et cible), on utilise ces liens pour l'analyse de performances, la synchronisation…
Réversibilité	La transformation est bidirectionnelle c.-à-d. exécuter dans les deux sens, on peut déduit un modèle cible à partir de modèle source.
Ordonnancement des règles	Cette propriété détermine l'ordre d'exécution des règles individuelles. Il y a deux mécanismes d'ordonnancement. Soit implicite si l'utilisateur n'a pas un contrôle explicite dans l'algorithme d'ordonnancement, sinon explicite si l'ordre d'exécution des règles est spécifié.
Organisation des règles	C'est la propriété qui détermine la composition et la structuration des règles de transformation. On peut classer ces règles selon le mécanisme de modularité, réutilisation et la structure d'organisation.

Réutilisabilité	Autoriser la réutilisation des règles de transformation dans d'autres transformations de modèles.
Incrémentation	Lorsque les modèles sources changent, on a la possibilité de mise à jour des modèles cibles.

Tableau 3.1 : Les propriétés de transformation de modèles.

3.3.8. Objectifs de la transformation de modèles

L'objectif principal de la transformation des modèles dans le contexte de l'IDM est la génération automatique du code, mais on peut citer d'autre fonctionnalité comme suit [41]:

- **Extraction de modèle** : L'extraction est l'action inverse de la génération de code, elle permet de construire un modèle visuel de haut niveau d'abstraction.
- **Vérification de modèle** : La vérification divise en deux parties comme suit:
  - ***L'analyse syntaxique*** : L'analyse syntaxique vérifie si le modèle conforme à un Meta modèle et si les conditions sont respectées.
  - ***L'analyse sémantique*** : L'analyse sémantique vérifie les propriétés dynamiques d'un domaine.
- **Simulation et exécution de modèle** : La simulation donne une vision sur l'exécution d'un modèle sans l'avoir exécuté réellement afin de minimiser le cout, le temps et les ressources.
- **Validation de modèle** : On peut dire qu'un modèle valide ou bien fiable si elle répond aux besoins exprimés par les clients.
- **Traduction de modèle** : Son objectif est de transformer un modèle représenté par un langage de modélisation en un modèle d'un autre langage de modélisation.
- **La gestion d'incohérence de modèle** : Les incohérences dans les modèles proviennent des points de vue multiples, donc on ne peut pas être évité ces incohérences. Par conséquent, nous avons besoins de techniques basées sur la transformation de modèle pour réparer les incohérences.

3.3.9. Classification des approches de transformation de modèles

Selon Czarnecki et Helsenont, la transformation des modèles est décomposée en deux grandes catégories, la transformation modèle vers modèles et la transformation modèles vers code [18] [40].

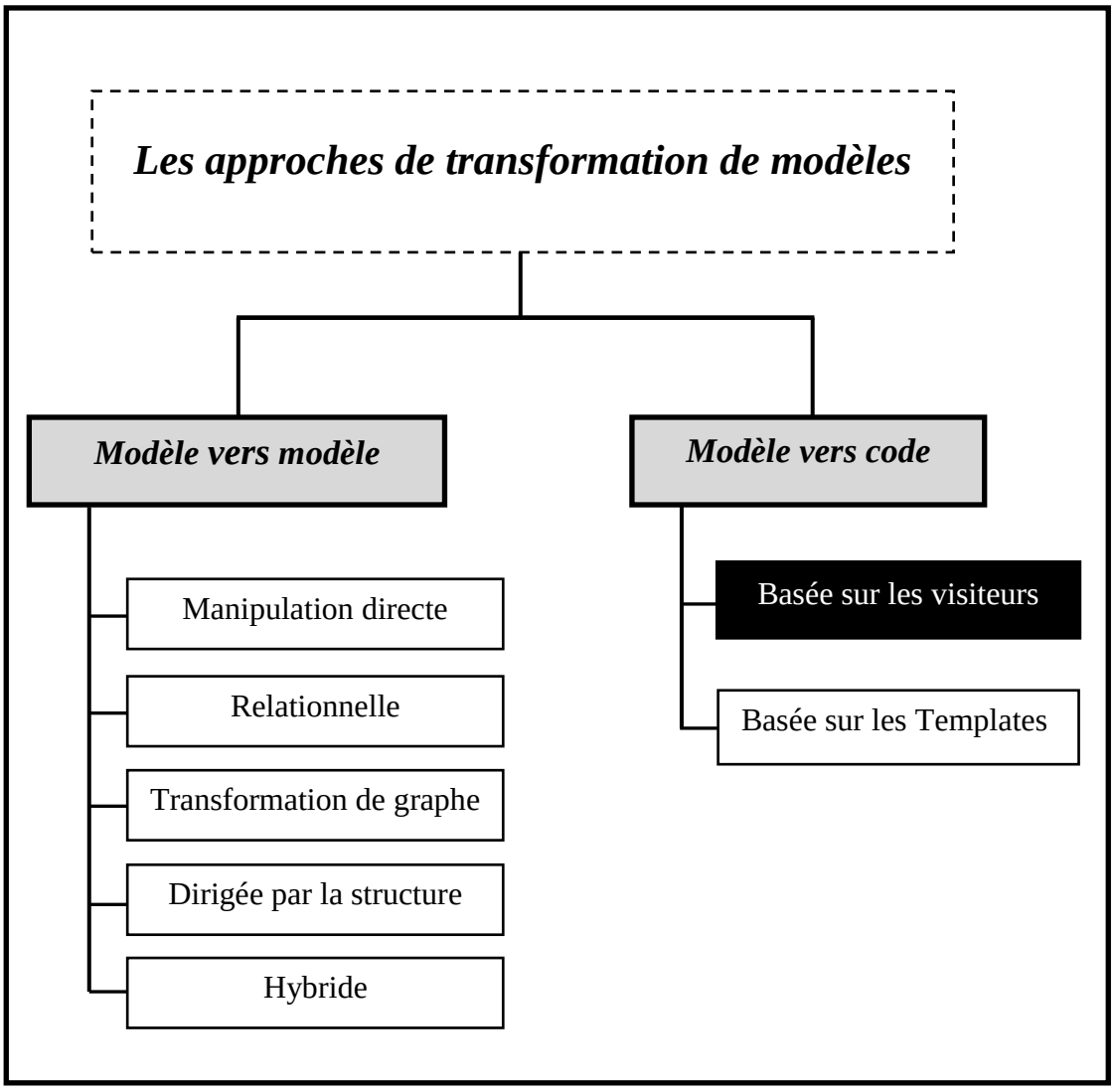

Figure 3.7 : Classification des approches de transformation.

- **Transformation de type modèle vers modèle :** Cette transformation permet de transforme des modèles sources vers des modèles cibles qui peuvent être des instances ou non d'un même méta-modèle, on peut définir plusieurs catégories dans ce type, parmi elles il y a :

- ***Les approches manipulant directement les modèles*** : Ces approches se basent sur des API qui permet de manipuler la représentation interne des modèles, on a comme exemple : EMF.
- ***Les approches relationnelles*** : Cette approche consiste à établir une relation entre les éléments des deux modèles à l'aide des contraintes pour spécifier ces relations, on a comme exemple : QVTP.
- ***Les approches basées sur la transformation de graphes*** : Ces approches se basent sur la théorie des transformations de graphe. Les règles de transformation sont définies pour des fragments de modèles, chaque règle est composée d'un graphe source (LHS : Left Hand Side) et d'un graphe cible (RHS : Right Hand Side), on a comme exemple : VIATRA.
- ***Les approches dirigées par la structure*** : La transformation de ces approches distinguent deux phases : la première crée la structure hiérarchique du modèle cible, la deuxième définit les valeurs des attributs et de références pour compléter le modèles, on a comme exemple : OptimalJ.
- ***Les approches hybrides*** : C'est la combinaison des approches précédentes. Le langage de transformation de règles est une combinaison d'approches déclarative et impérative, on a comme exemple : Atlas.

- **Transformation de type modèle vers code :** Cette transformation est un cas particulier de la transformation modèle vers modèle où le méta-modèle du modèle cible est la grammaire d'un langage de programmation. On distingue deux catégories dans ce type :
  - ***Approche basée sur l'utilisation des templates*** : un Template consiste généralement du texte cible contenant des épissures de méta-code pour accéder aux informations de la source et pour effectuer la sélection du code et l'expansion itérative.
  - ***Approche basée sur le visiteur*** : fournit un mécanisme de visiteur pour traverser la représentation interne d'un modèle et écrit le code, on a comme exemple : Jamda.

3.5. Conclusion

Nous avons présenté dans ce chapitre les approches IDM et MDA, particulièrement nous avons mis l'accent sur la transformation de modèles, les types de transformations, la classification des approches. Le but est de permettre la compréhension de notre travail qui consiste à la proposition d'une transformation de diagramme d'activité vers OWL-S.

Dans le dernier chapitre, nous présenterons les règles théoriques de transformation pour le passage entre le DAC et OWL-S, et l'automatisation de cette transformation.