Ce chapitre a pour objectif de représenter les systèmes cyber-physiques (CPS), avec leur définition et leurs caractéristiques aussi la modélisation .

1. Les systèmes cyber-physiques

1.1 -Définition :

Un système cyber-physique en anglais « cyber-physical system », en bref on l'appelle CPS : est un système ou où des éléments informatiques collaborant pour le contrôle et la commande d'entité physique [1].Une génération de précurseurs des systèmes cyber-physiques est très importante donc elle peut être trouvée dans plusieurs domaines comme l'automobile, aéronautique, les processus chimique, l'énergie, la santé, l'infrastructure civile, la fabrication, le transport, les divertissements et les appareils électroménagères ce qui peut être synthétisé par la (Figure 1.1).

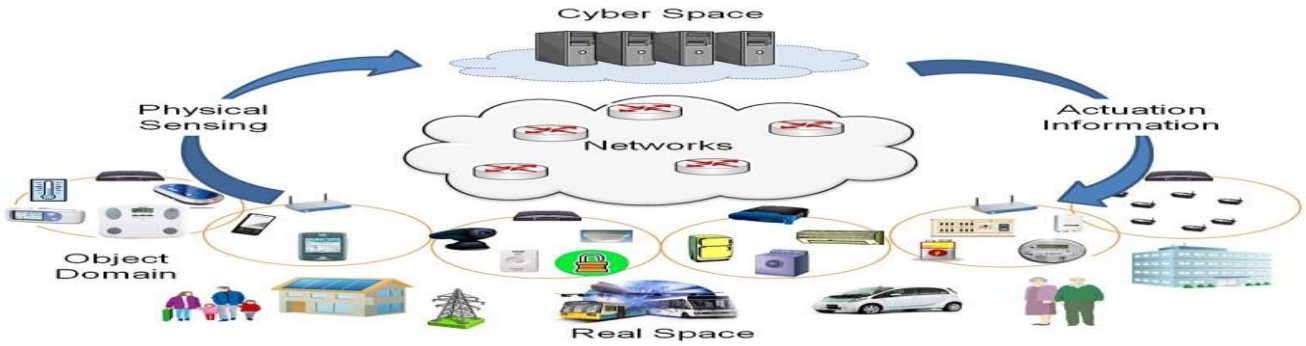

FIGURE 1.1 -domaine d'utilisation du CPS [1].

Cette génération est comme systèmes embarqués. Dans se derniers l'accent tend à être plus, sur les éléments informatiques et moins sur les liens entre les éléments informatiques et physiques.

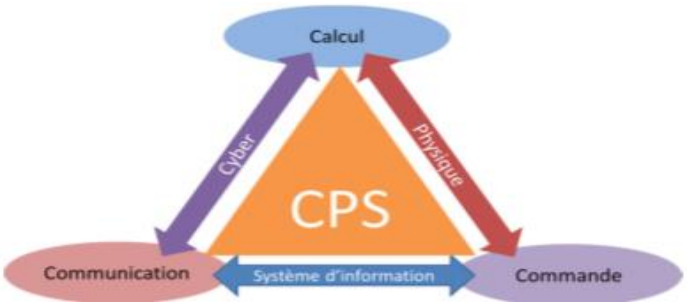

FIGURE 1.2 - Composantes d'un système cyber physique[1].

La déférence entre les systèmes cyber physiques et les systèmes embarqués traditionnelle c'est que un cps est généralement conçu comme un réseau d'éléments informatiques en interaction avec des entrées et des sorties physiques au lieu de dispositifs autonomes en interaction [2]. Cette notion est étroitement liée aux concepts de la robotique et des réseaux de capteurs.

Le développement dans les sciences et l'ingénierie permettront d'améliorer le lien entre les éléments de calcul et physiques, aussi faite une grande augmentation dans la capacité d'adaptation, l'autonomie, la fonctionnalité, l'efficacité, la fiabilité, la sécurité et finalement utiliser les systèmes cyber-physiques facilement. Ceci peut améliorer les capacités des systèmes cyber-physiques dans différentes dimensions on note [3] :

-**La réactivité** : par exemple pour l'évitement des collisions.

-**La précision** : par exemple de la chirurgie robotique

-**Travaille dans des environnements dangereux ou inaccessibles** : par exemple pour le sauvetage, la lutte contre les incendies.

-**La coordination** : par exemple pour le contrôle de la circulation aérienne dans le domaine militaire.

- **L'efficacité** : par exemple pour les bâtiments à énergie positive.

- **L'augmentation des capacités humaines** : par exemple pour les dispositifs de contrôle de santé personnels.

Les systèmes cyber-physiques sont un des deux piliers de l'Industrie 4.0 [5]. Pour cela la National Science Fondation (NSF) les a identifiés comme domaines de recherche prioritaires et très importants [6]. Depuis la fin de 2006, la NSF et d'autres agences gouvernementales ont financé des ateliers sur les systèmes cyber-physiques[7]Le 1er juillet 2019 l'Union européenne a lancé le projet CPS4EU[16] qui contient les grandes entreprises (VALEO, THALES, TRUMPF, RTE, LEONARDO et SCHNEIDER ELECTRIC).

1.1.1. Les caractéristiques les plus importantes [8] :

Les systèmes cyber-physique ont plusieurs caractéristiques comme Électronique, informatique, métrologie, automate, data fusion et mécanique en peut résumer tout ca dans la figure suivante :

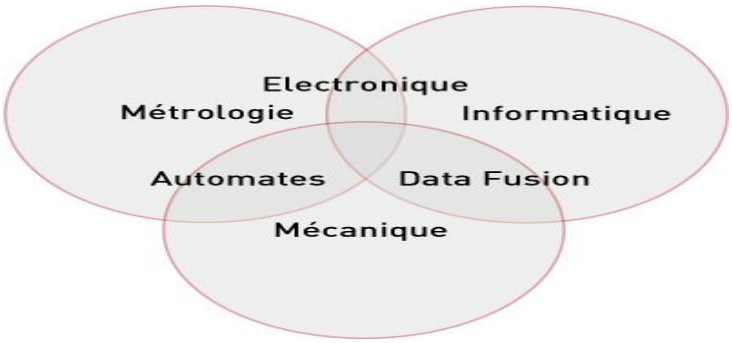

FIGURE 1.3 -Défférentes caractiristiques d'un CPS[8].

Chacune de ces caractéristiques est utilisée dans des cas différents :

Le domaine Électronique :

On cite :

-L'intelligence interne

- Circuits de protection (entre autres protection CEM)

- Utilisation sous conditions difficiles.

Le domaine Métrologie :

On cite :

-Haute précision de signaux

- Réduction des incertitudes de mesure.

Le domaine Informatique :

On cite :

-Connexion entre le monde physique et le monde numérique

- Interconnexion entre les systèmes et à l'extérieur

- Intégration dans le réseau de l'entreprise et aussi technologie de l'internet.

Le domaine Automates :

On cite :

- Connexion aux logiciels d'automates les plus courants

-Transfert seule des données appréciables et Transfert de données physique

Fusion de données :

On cite :

- La Combinaison et le calcul de données de différentes sources
- La Fusion de données multi-sensorielles.

Le domaine Mécanique :

On cite:

-Boîtier robuste contre les perturbations

-Résistance contre les chocs et l'intégration dans la machine

1.1.2 Quelques exemples de développements de CPS [9] :

Lorsque nous parlons de CPS nous devons parler tout d'abord de ses importants et utilisation dans les différents domaines (domaine d'automobile, aéronautique, l'énergie, la santé …ext) (parti précédente) . Donc comme une conséquence CPS ont fait l'objet de nombreux travaux de recherche lors de la dernière décennie, amenant assez régulièrement jusqu'au niveau du développement de prototypes fonctionnels. Dans ce qui suit nous essayons d'énumérer quelques exemples de développement.

Exemple 1:

Au début, les premières applications des CPS se sont basées sur les appareils de type « smartphone » pour déployer les applications. De ce fait, les applications d'assistance à la personne se sont développées, et particulièrement celles basées sur l'aide médicale. La vision de la « santé connectée » a pris de l'étalement ces dernières années, étalement grâce à l'essor de technologies connexes telles que les réseaux sans fil ou les capteurs. Ceci a permis le développement d'Equipements de Santé Personnels (Personal Health Devices (PHD)) qui visent à récupérer puis partager de l'information sur un réseau local ou Internet, tel que celui présenté sur la Figure 1.4 (Santos et al., 2015).

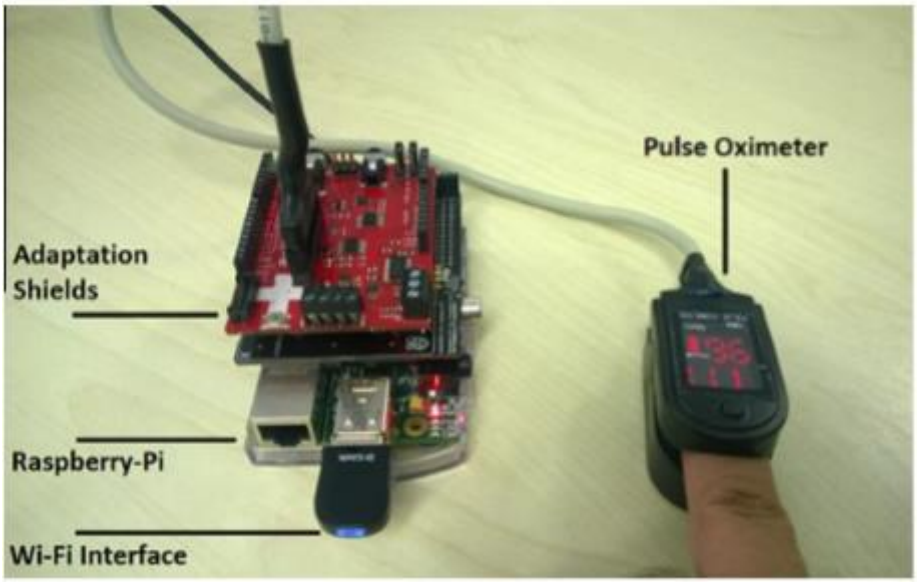

FIGURE 1.4-Prototype d'équipement personnel de santé (Santos et al., 2015)[9].

Exemple 2 :

Les CPS ont également de nombreuses applications dans le domaine de l'énergie (génération, distribution, consommation). Ces systèmes ont pour intérêt par exemple de diminuer les conséquences d'une défaillance.. Une ligne de 430km de long relie une turbine à vapeur d'eau à un bus infini. Le courant est mesuré de chaque côté de la ligne et est envoyé à un contrôleur central par le réseau. Ce contrôleur, par comparaison du courant sur les phases, détecte les éventuelles défaillances et agit sur les coupe-circuits, également via le réseau, ce qui constitue un CPS : {C2 ; Energy ; Full ; Ethernet}

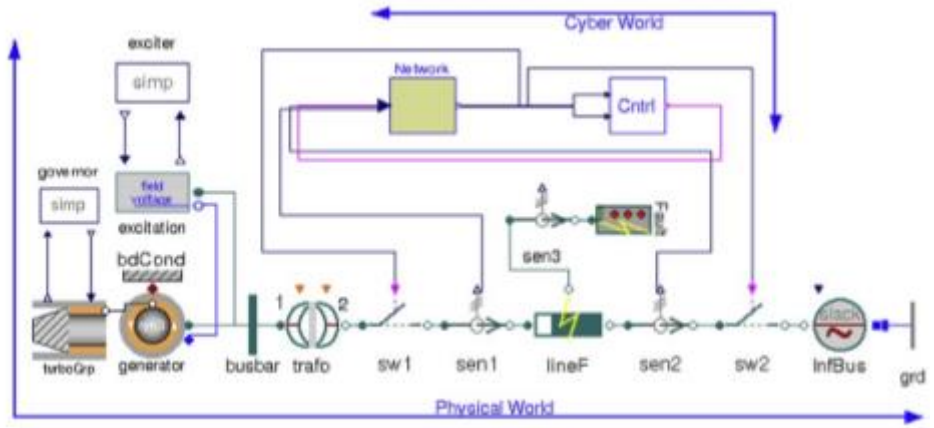

FIGURE 1.5 -Schéma de principe d'un système de transmission de puissance électrique (Al-Hammouri, 2012)[9]

1.2 La modélisation des systèmes cyber-physiques

1.2.1 Introduction :

Les systèmes cyber-physique ou embarqués sont des systèmes intégrant du logiciel et matériel, inclus dans des objets ou des systèmes qui ne sont pas perçus comme des ordinateurs par leurs utilisateurs.

La complexité croissante des systèmes informatiques entraîne des difficultés d'ingénierie pour les systèmes à base de composants, en particulier liées à l'optimisation, la validation et à l'analyse des performances et des exigences concernant la sûreté de fonctionnement. Des approches d'ingénierie guidée par des modèles sont de plus en plus utilisées dans l'industrie dans l'objectif de maîtriser cette complexité au niveau de la conception. Ces approches encouragent la réutilisation et l'automatisation du cycle de développement. Elles doivent être accompagnées de langages et outils capables d'assurer la conformité du système implémenté aux spécifications.

1.2.2 Architecture Dirigée par les Modèles [10] :

L'Architecture Dirigée par les Modèles (Le Model Driven Architecture, MDA)[11]est une démarche de développement proposée par l'Object Managment Group (OMG) [12]. Elle permet de séparer les spécifications fonctionnelles d'un système des spécifications de son implémentation sur une plate-forme donnée. A cette fin, le MDA définit une architecture de spécifications structurée en modèles indépendants des plates-formes (Platform Independent Model, PIM) et en modèles spécifiques (Plateform Specific Model, PSM).

L'approche MDA permet de réaliser le même modèle sur plusieurs plates formes grâce à des projections standardisées. Elle permet aux applications d'interopérer en reliant leurs modèles et supporte l'évolution des plates formes et des techniques. La mise en œuvre du MDA est entièrement basée sur les modèles et leurs transformations.

Cette approche consiste à manipuler différents modèles de l'application à produire, depuis une description très abstraite jusqu'à une représentation correspondant à l'implantation effective du système. Le processus MDA se décompose en trois étapes principales, représentées sur la figure (1.6) :

1. la définition d'un modèle de très haut niveau, indépendant des contraintes d'implantation : le PIM peut être raffiné afin d'ajouter différentes informations non fonctionnelles telles que la

gestion de la sécurité ; ces informations demeurent indépendantes de la plateforme d'exécution qui sera effectivement utilisée ;
2. le PIM est ensuite transformé pour prendre en compte les spécifications propres à la plateforme d'exécution ; le nouveau modèle est appelé PSM; le PSM peut être également raffiné afin de prendre en compte différents paramètres liés à l'environnement d'exécution.
3. Le PSM est ensuite utilisé pour générer une application exécutable basée sur les spécifications à partir desquelles le PIM a été construit

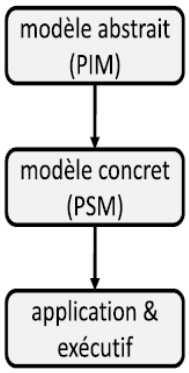

FIGURE 1.6 - L'approche MDA [12].

MDA propose une démarche intégrée permettant de rassembler tous les éléments pour la description d'une application. De cette façon, MDA vise la mise en place de différentes étapes de raffinement depuis la conception de l'application jusqu'à la production de code exécutable.

1.2.3 Modèles formels des applications :

L'approche MDA peut être instanciée en utilisant différents langages. C'est pourquoi un certain nombre de langages permettant cette description sont apparus et offrent à ce jour un certain nombre de fonctionnalités. La plupart des approches guidées par des modèles se basent soit sur :

• Le langage de modélisation unifié (Unified Modeling Language, UML), qui est un langage de modélisation à usage général et ses profiles comme MARTE (Modeling and Analysis of Real-Time and Embedded) . Si UML permet une grande expressivité, il ne transporte par en lui-même une sémantique précise pour décrire les architectures.

• Des langages de description d'architecture (Architecture Description Language, ADL), qui sont des langages propres à des domaines particuliers. Un ADL est un langage qui permet la modélisation d'une architecture conceptuelle d'un système logiciel et/ou matériel. Il fourni une syntaxe concrète et une structure générique (framework) conceptuelle pour caractériser

les architectures. Parmi les langages ADL on trouve le langage AADL qu'on va détailler dans la suite de ce chapitre.

• Des langages spécifiques, conçus pour répondre à des besoins particuliers de modélisation et d'analyse, par exemple Behavior Interaction Priority (BIP), Fractal et Ptolemy.

1.2.4 Le langage AADL [13] :

Parmi les ADL, le langage d'analyse et de description d'architectures (*Architecture Analysis and Design Language*, AADL), a fait l'objet d'un intérêt croissant dans l'industrie des systèmes embarqués critiques(comme Honeywell, Rockwell Collins, l'Agence Spatiale Européenne, Astrium, Airbus). AADL a été standardisé par la SAE (*International Society of Automotive Engineers*) en 2004, pour faciliter la conception et l'analyse de systèmes complexes, critiques, temps réel dans des domaines comme l'avionique, l'automobile et le spatial. AADL fournit une notation textuelle et graphique standardisée pour décrire des architectures matérielles et logicielles. Le succès d'AADL dans l'industrie est justifié par son support avancéà la fois pour la modélisation d'architectures reconfigurables et pour la conduite d'analyses. En particulier, le langage a été conçu pour être extensibleafin de permettre des analyses qui ne sont pas réalisables avec le langage de base.

AADL (Architecture Analysis & Design Language) est un langage de description d'architectures. Ce langage permet la conception et la description d'architectures temps-réel embarquées, en considérant simultanément tous les aspects logiciels et matériels sous-jascents. C'est la SAE (Society of Automotive Engineers) qui a préparé le standard international d'AADL depuis 2001, dont la version 1.0 est parue en 2004.

1.2.4.1 Principes du langage AADL :

AADL peut être exprimé selon différentes syntaxes. Le standard en définit trois : en texte, en XML, ainsi qu'une représentation graphique. Par cette multiplicité des syntaxes possibles, AADL peut être utilisé par de nombreux outils différents, graphiques ou non. Le développement de profils pour UML permet également d'envisager l'intégration d'AADL au sein d'outils de modélisation UML.

AADL a été créé avec le souci de faciliter l'interopérabilité des différents outils, c'est pourquoi la syntaxe de référence est textuelle. La représentation XML permet de faciliter la création de parseurs pour des applications existantes. La notation graphique se pose en complément de la notation textuelle, pour faciliter la description des architectures. Elle permet une représentation beaucoup plus claire que le texte ou XML, mais elle est moins expressive.

En tant que langage de description d'architecture, AADL permet de décrire des composants connectés entre eux pour former une architecture. Une description AADL consiste en un ensemble de déclarations de composants. Ces déclarations peuvent être instanciées pour former la modélisation d'une architecture.

1.2.4.2 Les composants AADL

Les composants AADL sont définis en deux parties : l'interface et les implantations. Un composant AADL possède une interface (component type) à laquelle correspondent zéro, une ou plusieurs implantations (component implémentation). Toutes les implantations d'un composant partagent la même interface.

a. Types et implantations

Le type d'un composant définit son interface tandis que l'implantation décrit les éléments (sous-clauses) de sa structure interne. Une implantation fait toujours référence à un type défini par ailleurs. AADL étant un langage déclaratif, l'ordre des déclarations n'importe pas : il est possible de déclarer une implantation avant le type correspondant. Voici un exemple illustratif.

```
processor myProcessor
end myProcessor;

processor implementation myProcessor.Intel
end myProcessor.Intel;

processor newProcessor
end newProcessor;

processor implementation newProcessor.newIntel
                         extends myProcessor.Intel
end newProcessor.newIntel;
```

FIGURE 1.7 -Types et implantations de composants AADL[13]

Il est possible d'étendre une déclaration de composant (type ou implantation) afin d'y ajouter ou d'en préciser des éléments. La figure 1.8 résume les mécanismes d'extensions possibles. Nous pouvons remarquer que l'implantation d'un type qui en étend un autre peut étendre une implantation du type initial.

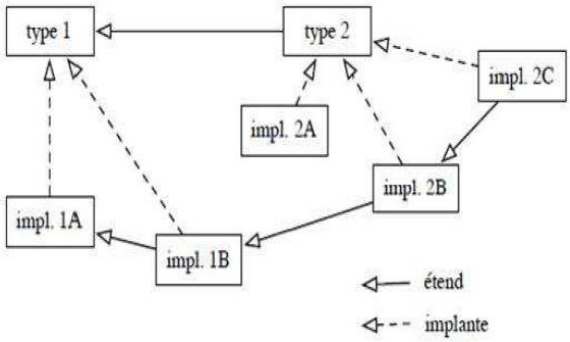

FIGURE 1.8 -Possibilités d'extension des composants [13]

L'extension de composant AADL ne correspond pas exactement à la notion d'héritage des langages objet : un composant et son extension constituent des composants distincts; il n'est pas possible d'utiliser l'un pour l'autre au sein d'une description architecturale. En revanche, les implantations d'un type de composant peuvent être substituées les unes aux autres dans la mesure où elles partagent la même interface aux autres composants.

b. Catégories de composants:

AADL définit plusieurs catégories de composants, réparties en trois grandes familles :

– **les composants logiciels** définissent les éléments applicatifs de l'architecture.

– **les composants de la plate-forme** d'exécution modélisent les éléments matériels.

– **les systèmes** permettent de regrouper différents composants en entités logiques pour structurer l'architecture.

1.Composants logiciels :

Il existe quatre catégories de composants logiciels :

– les données.

– les sous-programmes.

– les fils d'exécution (threads).

– les processus.

(a) : Données (b) : Sous-programmes (c) : Threads (d) : Processus

FIGURE 1.9 -Syntaxe graphique des composants logiciels [13].

2. Composants de plate-forme :

Il existe quatre catégories de composants de plate-forme :

– les processeurs.

– les mémoires.

– les bus.

– les devices (dispositifs).

(a) : Processeurs (b) : mémoires (c) : bus (d) : devices

FIGURE 1.10 -Syntaxe graphique des composants matériels[13].

1.3 Conclusion

Dans ce chapitre nous avons présenté système cyber physique avec ces caractéristique et Domain d'utilisation et en termine par la modélisation .

2.1 Introduction

Aujourd'hui, car l'augmentation de la taille et la complexité des logiciels développés de plus en plus rapidement on trouve des nouvelles techniques du génie logiciel qui sont apparues.

Dans ce contexte, Les grandes tendances actuelles sont basées sur l'utilisation de modèles pour permettre une montée en abstraction par rapport aux langages de programmation, et pour faciliter et simplifier notre travail. C'est Ingénierie Dirigée par les Modèles (IDM).

Dans ce chapitre, nous présentons une définition pour IDM et les concepts de base de la démarche IDM (modèle, Méta-Modèle, Méta-méta-modèle) en expliquant avec plus de détail l'approche MDA, ses principes, différents modèles et en termine par la transformation de ces modèles Ensuite, nous illustrons quelques notions fondamentaux autour de la transformation de modèles.

2.2 Ingénierie Dirigée par les Modèles

L'ingénierie dirigée par les modèles (IDM) ' Model Driven Engineering ' (MDE) en anglais, IDM c'est le résultat de l'évolution du processus de développement logiciel d'une utilisation "contemplative" à une utilisation "productive" des modèles, en outre c'est une approche parmi les approches qui présente la mécanisation du processus que les ingénieurs expérimentés suivent manuellement ,elle est plus globale et générale que l'architecture dirigée par les modèles , IDM est basée sur la transformation de modèle car ce dernier joue un rôle fondamental et considérée l'une des techniques prometteuses dans cette approche. les modèles étaient utilisés comme des éléments de conception, de discussion ou de documentation, l'idée de l'IDM est de les exploiter comme entrée du processus de développement.

Dans le coté pratique, cela nécessite de formaliser les modèles pour les rendre exploitables par la machine et de réaliser des programmes permettant de traiter des modèles.

Dans la terminologie de l'IDM, ces programmes sont regroupés sous le terme de transformations de modèles [14].

Dans 'IDM', les modèles sont utilisés de manière systématique tout le long du processus de développement d'un logiciel et ils doivent être suffisamment précis afin de pouvoir être transformés. Le processus de développement d'un logiciel est caractérisé par une séquence partiellement ordonnée de transformations de modèles. Chaque transformation prend des

modèles en entrée et produit des modèles en sortie, jusqu'à l'obtention d'artefacts exécutables [15].

2.2.1 Concepts fondamentaux

Cette partie définit les concepts de base de l'ingénierie des modèles et les relations qui existent entre ces concepts.

2.2.1.1 Modèle (m1)

Un modèle est une simplification de la réalité, Il décrit les informations de M0. De plus est une abstraction d'un système construite dans un but donné ,il contient un ensemble restreint d'informations sur un système et il est construit dans un but précis en ne contenant que les informations pertinentes vis-à-vis de l'utilisation qui sera faite du modèle [15].Ces modèles permettent de mieux comprendre le système que l'on développe.

- ✓ **Niveau M0 :** Le niveau M0 (ou **instance**) correspond au monde réel (réalité) . Ce niveau contient les informations réelles de l'utilisateur, instances du modèle du niveau M1.

2.2.1.2 Méta-Modèle (m2) : Langage de modélisation

Un méta-modèle (niveau M2) définit le langage de modélisation des modèles M1, il permet d'énoncer des modèles en définissant les concepts nécessaires et les relations entre eux. Donc, un modèle est une construction possible du méta-modèle dans lequel il est défini. Dans la littérature, un modèle est dit conforme à son méta-modèle.

2.2.1.3 Méta-méta-modèle (m3) : langage de méta-modélisation

Dans l'approche MDA, méta-méta-modèle (niveau M3) est composé d'une unique entité qui s'appelle le MOF. Ce dernier permet de décrire la structure des méta-modèles. Donc il est nécessaire d'avoir un méta-modèle pour interpréter un modèle, pour pouvoir interpréter un méta-modèle il faut disposer d'une description du langage dans lequel il est écrit : un méta-modèle pour les méta-modèles. C'est naturellement que l'on désigne ce méta-modèle particulier par le terme de méta-méta-modèle [14]. Pour limiter le nombre de niveaux d'abstraction, et éviter de définir un méta-méta-méta-modèle, les méta-méta-modèles sont généralement conçus pour être auto-descriptifs, c'est-à-dire capables de se décrire eux-mêmes.

Les figures 2.1 et 2.2 représentent les différentes relations qui existent entre les modèles, méta-modèles et méta-méta-modèles. On distingue sur ces figures les trois niveaux de modélisation M1, M2 et M3. Dans la littérature, et en particulier dans certains documents de l'OMG, le niveau M0 correspondant au système que représente un modèle du niveau M1. La nature de la relation qui existe entre le niveau M0 et M1 est donc différente de la nature des

relations existant entre les niveaux M1 et M2 et M2 et M3. Pour éviter ce problème d'homogénéité, le niveau M0 n'est généralement pas considéré.

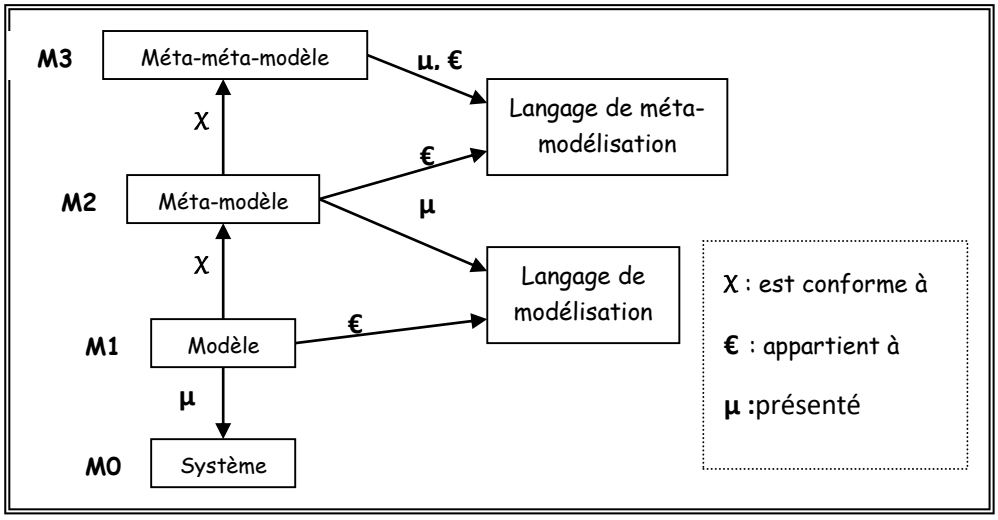

FIGURE 2.1 -Niveaux de modélisation[14].

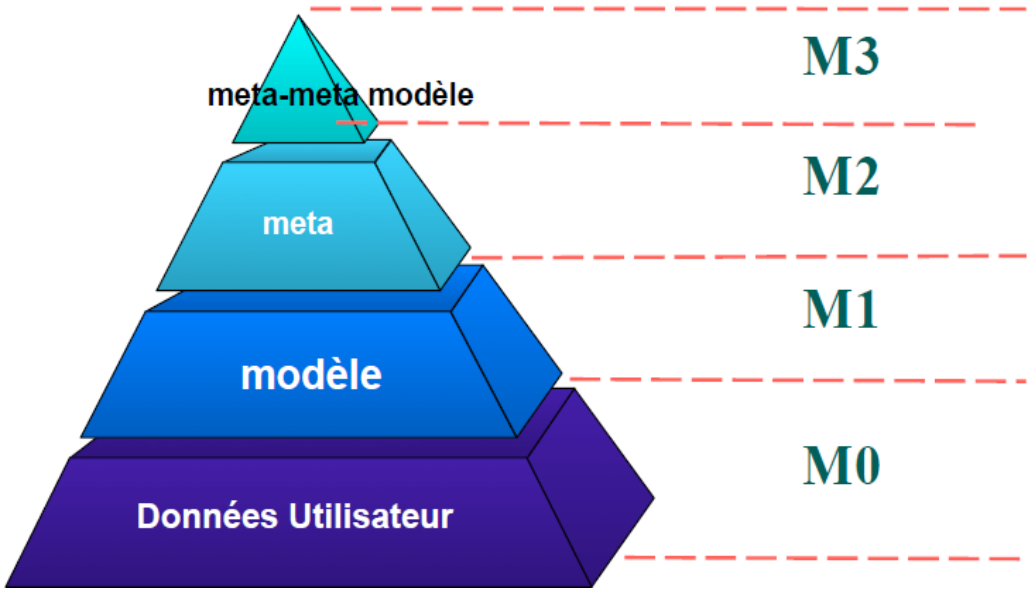

FIGURE 2.2 -autre image pour préciser les niveaux de modalisation [14].

On retrouve également dans ces figures, les trois relations de base de l'ingénierie des modèles: la conformité, la représentation et l'appartenance. Cela permet de mettre en évidence les relations qui existent entre modèles et langages.

2.2.2 Architecture Dirigée par les Modèles

MDA est une approche proposée et soutenue par l'Object Management Group (OMG) en novembre 2000.Cette approche peut être considérée comme une variante du MDE pour le génie logiciel. C'est un champ de développement, de maintenance et d'évolution de produits logiciels basé sur l'utilisation de modèles. Nous décrivons son principe et ses objets dans cette partie.

2.2.2.1 Principe du MDA

La démarche MDA a toutes les techniques de modélisation et de transformation qui réalise un changement notable dans la conception des applications. Elle est basée sur un principe qui sépare les spécifications fonctionnelles des spécifications de l'implantation sur une plate-forme donnée => interopérabilité des applications.

L'idée principale de MDA est d'aménager des modèles, d'abord d'analyse puis de conception, jusqu'au code, par transformations, dérivations et enrichissements successifs

L'OMG expose le langage déclaratif (à base de règles) "QVT" (Query/View/Transformation) pour exprimer les transformations de ces modèles.

Les principaux modèles sont :

- **CIM** modèle indépendant de calcul (computation independant model en anglais) : chargé de décrire les flux et les actions sur le système.
- **PIM** modèle indépendant des plates-formes (plateform independant model en anglais): chargé de décrire les traitements orientés métier.
- **PDM**modèle des plates-formes (plateform dependant model en anglais): chargé de décrire une architecture technique (plusieurs par projet)
- **PSM** modèle dépendant des plates-formes (plateforme specific model en anglais) : chargé de décrire les détails techniques liés à l'implantation pour une plate-forme

On peut les voire dans la figure 2.3 au-dessous.

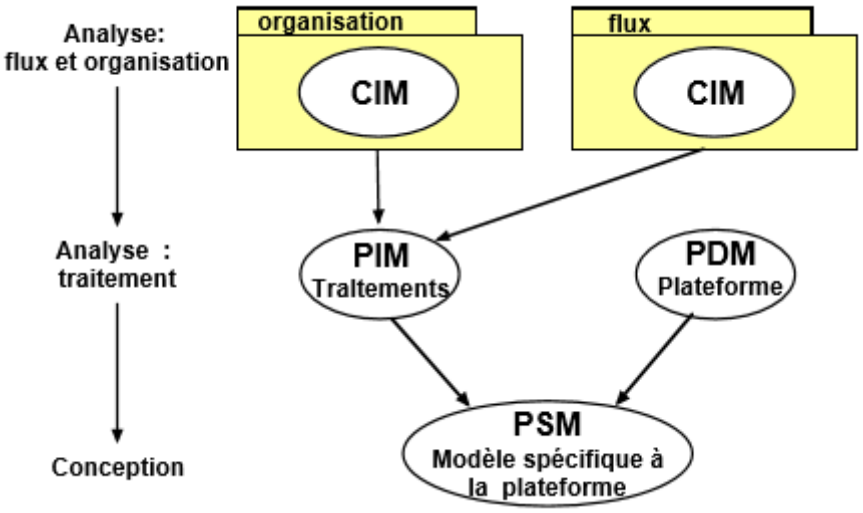

FIGURE 2.3 - Les principaux modèles[14].

2.2.2.2 Différents modèles du MDA

Comme nous avons déjà vue dans la section précédente l'Architecture Dirigée par les Modèles (Model Driven Architecture – MDA en anglais) : est une approche proposée par l'Object Management Group (OMG),il est composé de plusieurs modèles qui vont servir dans un premier temps à modéliser l'application, puis par transformations successives à générer du code ces modèles sont :CIM , PIM,PDM,PSM

Le CIM (Computation Independent Model) :

Le CIM est un modèle parmi les modèle MDA, en outre c'est le modèle d'analyse de base du métier ou du domaine d'application, la disposition du CIM consiste à décrire le produit indépendamment de tout système informatique mais en respectant les exigences des utilisateurs. Cette indépendance technique permet de capitaliser le savoir faire dans le CIM en faisant abstraction de la technologie d'implantation [16].Aussi il lui permet de garder tout son intérêt au cours du temps et il est modifié uniquement si les connaissances ou les besoins métier changent.

Les exigences modélisées dans le CIM seront prise en compte dans les constructions des PIM (Platform Independent Model) et des PSM (Platform Specific Model)

Le PIM (Platform Independent Model)

Le PIM supplément c'est un modèle de conception qui décrit le système indépendamment de toute plate-forme technique et de toute technologie utilisée pour déployer l'application, il est connexe avec CIM ,pour cela une fois le

CIM établi, il est possible de concevoir les modèles d'analyse et de conception (PIM) en respectant les conditions d'utilisation du produit spécifiées dans le CIM.

Le PIM intègre les aspects technologiques et architecturaux sans manifester les détails de l'utilisation du produit sur sa plate-forme technique et caractérise le fonctionnement des entités et des services. Pour coté pratique, il existe différentes méthodes qui permettent de construire des modèles d'analyse et de conception telles que Merise, Booch et OMT.

Toutefois, le MDA recommande d'utiliser UML pour représenter les PIM, a ce niveau, le formalisme utilisé est un diagramme de classes qui peut-être couplé avec un langage de contrainte comme OCL (Object Constraint Language) [17].

Le PDM (Plateform Description Model)

Le pdm supplément c'est un modèle de transformation pour permettre la circulation du PIM vers le PSM**,** car il contient des informations pour la transformation des modèles vers une plateforme donc Apres la définition du PIM, l'architecte doit choisir une plate-forme technique qui permet d'implanter le produit avec la qualité architecturale décrite dans le PIM. Cette plateforme doit être décrite dans le modèle de pdm afin de permettre son intégration dans la construction du produit. . L'OMG ne donne pas plus de précisions sur le PDM dans la description du MDA. Dans la littérature, on pense que chaque fournisseur de plate-forme devrait proposer le PDM [16].

Une plate forme est un ensemble de sous-système et de technologies qui fournissent un ensemble cohérent de fonctionnalités à travers des interfaces qui peuvent être employés par n'importe quelle application soutenue par cette plateforme sans souci les détails de la façon dont la fonctionnalité fournie par la plateforme est mise en application [18].

Le PSM (Platform Specifique Model)

Une fois le PIM et le PDM connus, on peut construire le modèle du code ou modèle de conception effective du produit : le PSM. Le PSM dépend de la plate-forme technique choisie par l'architecte, il indique comment le produit sera utilisé sur cette plateforme et il sert essentiellement de base à la génération du code exécutable vers la plate-forme choisie [19].

2.2.2.3 Transformation de modèles MDA

Pour préparer et faciliter la génération de code vers la plate-forme technique choisie, MDA consiste à s'écouler progressivement des PIM aux PSM. Cette circulation des PIM aux PSM est une transformation de modèles. Le MDA identifie différents types de transformations de modèles dans le cycle de vie et de développement d'un produit. La figure 2.4 ci-dessous synthétise les différentes transformations possibles des modèles.

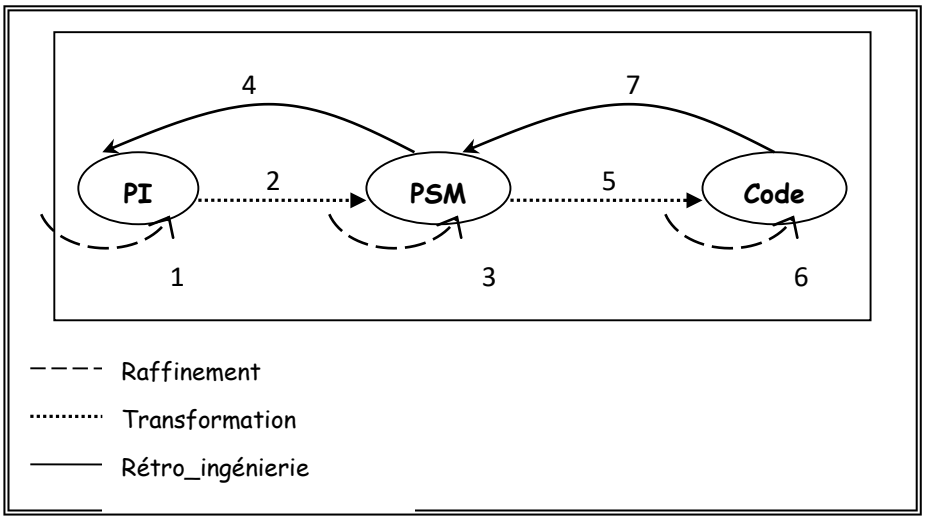

FIGURE 2.4 - Transformation des modèles MDA[19].

Le niveau de modèle le plus abstrait est représenté par Le PIM .Par passages consécutifs, les modèles deviennent de plus en plus concrets jusqu'à l'obtention du code. Ce code peut être assimilé à un PSM exécutable, de même que la génération de code n'est pas toujours considérée comme une transformation de modèle. La figure 2.4 montre aussi qu'il est possible à partir du code de recréer un PSM puis un PIM [17].

2.3 Transformation de modèles

L'ingénierie des modèles a deux principaux artefacts sont les modèles et les transformations de modèles. Généralement, on appelle *transformation de modèles* tout programme dont les entrées et les sorties sont des modèles. Maintenant plusieurs axes de recherche pour la transformation des modèles sont explorés. Cette section propose un rapide tour d'horizon des principes de base de transformation de modèles.

2.3.1 Définitions générales

Dans l'IDM, un ensemble de modèles cibles est génère par une transformation de modèles s'appuyer sur d'un ensemble de modèles sources conformément à une définition de

transformation. Les modèles sources et cibles sont décrites dans des méta-modèles précis. Ces méta-modèles peuvent être identiques ou différents. Les modèles sources et cibles peuvent appartenir à un même niveau d'abstraction ou bien différents [22].

2.3.1.1 Transformations endogènes et exogènes

On compte deux types de transformations de modèles par rapport à la différenciation des méta-modèles de définition des modèles sources et cibles, Transformations endogènes et exogènes :

1- Une transformation endogène : si les modèles sources et cibles sont définis dans le même méta-modèle.

2- Une transformation exogène : si les modèles sources et cibles sont définis dans déférent méta-modèle

2.3.1.2 Transformations horizontales et verticales

On compte deux types de transformations par rapport à la différenciation du niveau d'abstraction des modèles sources et cibles : les transformations horizontales et les transformations verticales. Une transformation est dite :

1- Horizontale : si les modèles sources et cibles appartiennent au même niveau d'abstraction. On cite comme un exemple typique de transformation horizontale: Le raffinement**.**

2- Verticale :si les modèles sources et cibles appartiennent à des niveaux d'abstractions différents. Par exemple la génération de code.

2.3.2 Concepts de base et principes

Une transformation est définie sur la base des méta-modèles de sa source et de sa cible. Les méta-modèles et les modèles (source, cible, y compris le modèle de la transformation) sont enregistrés dans un ensemble de référentiels. Cette transformation est exécutée à l'aide d'un moteur de transformation. On peut voir ca dans La figure 2.5 qui résume les concepts de base et le principe d'une transformation de modèles dans les différents outils.

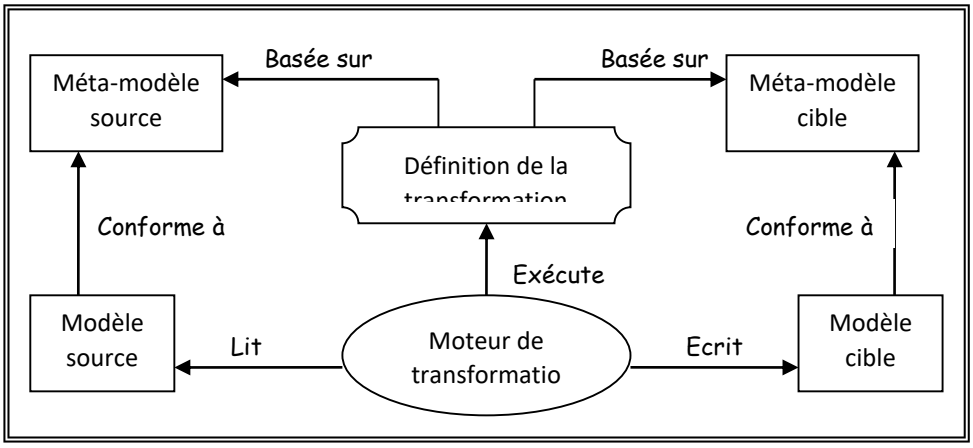

FIGURE 2.5 - Concepts de base et principe des transformations de modèles[19].

On rappelé que un moteur de transformation est un outil parmi les outils qui est susceptible de lire un modèle dans un référentiel donné et permet d'appliquer la transformation pour générer une solution conforme à la définition de la transformation. cette solution générée est soit un modèle, soit du code (texte). Pratiquement, on parle de la transformation de type « modèle vers modèle », dans le premier cas, et de la transformation de « modèle vers code » dans le deuxième cas [16].

2.3.2.1 Transformation de type modèle vers code :

Cette Transformation est un cas particulier de transformation de « modèle vers modèle » dans laquelle le méta-modèle de la cible correspond à la grammaire du langage de programmation cible ce type de transformation est utilisé pour générer du code dans un langage de programmation particulier (Java, C++ …etc) à partir d'un modèle source [16].
Puis en passe au deuxième type de transformation.

2.3.2.2 Transformation de type modèle vers modèle :

ce type de Transformation est basée sur les modèles dans les deux cotés ,il est utilisé pour réaliser une activité de génération automatique de modèles. Par exemple la dérivation de lignes de produits, qui est l'objectif du travail présent, le raffinement de modèles, la création d'un PSM à partir d'un PIM dans le MDA,…etc.

2.3.3 Caractéristiques d'une transformation de modèles

Une transformation de modèles est caractérisée par plusieurs éléments, tel que :Les règles de transformation , La portée des règles , La traçabilité , La directivité:

- ✓ **Les règles de transformation :** Une règle de transformation renferme deux parties : une partie gauche (LHS : Left-Hand Side) et une partie droite (RHS : Right-Hand Side).
  partie gauche : décrit les accès aux éléments du modèle source .
  partie droite : exprime les expansions (création, modification, suppression d'éléments) dans le modèle cible.
  Chaque partie peut être décrite en combinant des variables, des patterns et des expressions logiques décrivant des calculs et des contraintes sur les éléments de modèles.

- ✓ **La portée des règles :** Permet de limiter les parties d'un modèle (source ou cible) qui peuvent participer à la transformation.

- ✓ **La traçabilité :** Autorise de garder la trace des liens entre éléments source et cible, d'analyser l'influence d'un changement d'un modèle sur un autre et de synchroniser entre les modèles.

- ✓ **La directivité:** Les transformations peuvent être unidirectionnelles ou bidirectionnelles.
  Les transformations unidirectionnelles : peuvent s'exécuter dans un seul sens : le modèle cible est généré (crée ou modifie) à partir du modèle source, mais il est impossible de reconstituer le modèle source à partir du modèle cible. Par contre Les transformations bidirectionnelles s'exécutent dans les deux sens : le modèle cible est généré à partir du modèle source et vis versa.

2.3.4 Outils et langages de transformation

L'IDM réalise une grande réussite dans le domaine de l'ingénierie des logiciels grâce aux transformations de modèles. Cette réussite permet l'apparition de nombreuses techniques et outils de transformations de modèles talque : AGG, AToM3, MTL. Nous intéressons dans ce mémoire la par l'outil AToM3 qui sera présenté dans le dernier chapitre.

Dans ce domaine la Jean-Marc Jézéquel propose une classification intéressante dans laquelle, il regroupe les différents outils de transformations de modèles dans cinq catégories présentées ci-après :

1- Les outils génériques :

Ce type de outils faire le regroupent principalement les techniques de transformation d'arbres et de graphes. Ils permettent de transformer des modèles qui prennent la forme de graphes dirigés, attribués, à nœuds et arcs étiquetés. Il existe plusieurs systèmes de transformation de graphes qui permettent de réaliser des transformations de modèles, par exemple : ATOM3 et AGG.

2-Les outils intégrés aux ateliers de génie logiciel :

Les outils intégrés aux ateliers de génie logiciel sont majoritairement des outils propriétaires qui sont commercialisés par les vendeurs des ateliers de génie logiciel. On cite par exemple, l'outil Objecteering de Objecteering Software. Le profite de ces outils est leur intégration dans ces ateliers qui sont matures. Cette intégration poussée à un inconvénient : ces outils de transformation de modèles sont la plupart du temps propriétaires, non standardisés. De plus, ils ne sont que des fonctionnalités additionnelles pour leurs ateliers de génie logiciel. Ils n'ont pas été définis spécifiquement pour l'application de développements dirigés par les modèles [15].

3-Les outils spécifiques à la transformation des modèles :

On trouve de nombreux outils universitaires et industriels dans cette catégorie. Le standard QVT de l'OMG est l'outil de référence de cette catégorie. Dans le milieu universitaire on trouve par exemple les outils ATL (implante le standard QVT) et MTL de l'INRIA,…etc. comme des points forts de cet outil, on cite :

- la simplicité de développement et de maintenance des transformations
- la puissance d'expression et la composition des règles de transformation [15].

4-Les outils de méta-modélisation supportant les transformations :

Pour construire des modèles de transformations orientés objets et MDE pour exécuter ces modèles de transformations ce type de outil utilisent l'approche de programmation par objets. Dans ces outils, une transformation de modèles est tout simplement un méta-programme exécutable. Les outils de méta-modélisation sont moins nombreux que les outils spécifiques aux transformations de modèles [15].

5-Les outils associés aux langages de programmation :

Finalement en termine avec un autre type de outil c'est les outils associés aux langages de programmation,ce dernier est disponible sous forme d'APIs (Application Programming

Interface ou interface de programmation) dans des langages de programmation habituels tels que les langages Visual Basic, C++, C#....ext . L'atout majeur de ces outils est relatif au fait que les programmeurs de ces langages n'ont pas besoin d'apprendre de nouveaux langages pour définir des transformations de modèles. Mais, ces outils ne sont pas vraiment adaptés pour méta-modélisation et l'élaboration de transformations complexes.

2.3.5 Classification des outils de transformation

Les travaux réalisés dans le domaine de la transformation de modèles ne sont pas récents et peuvent être chronologiquement classés selon plusieurs générations en fonction de la structure de donnée utilisée pour représenter le modèle

- **Génération 1 : Transformation de structures séquentielles d'enregistrement**

  Dans ce cas un script spécifie la manière de récréation d'un fichier d'entrée en un fichier de sortie. Bien que ces systèmes soient plus lisibles et maintenables que d'autres systèmes de transformation, ils nécessitent une analyse grammaticale du texte d'entrée et une adaptation du texte de sortie.

- **Génération 2 : Transformation d'arbres**

  Ces méthodes permettent de faire un parcours d'un arbre d'entrée au cours duquel sont générés les fragments de l'arbre de sortie.

- **Génération 3 : Transformation de graphes**

  Ces méthodes permettent des transformations du modèle, c'est a dire un modèle en entrée (graphe orienté étiqueté) est transformé en un modèle en sortie. Ces approches visent à considérer l'opération de transformation comme un autre modèle conforme à son propre méta-modèle (lui-même défini à l'aide d'un langage de méta-modélisation).

2.3.6 La validation

Les transformations de modèles est très considérable dans le modèle d'ingénierie dirigée par les modèles. Elles permettent l'automatisation de certaines activités de développement aussi et la réutilisation de savoir faire. Pour assurer la qualité du processus de développement, il est nécessaire d'assurer la qualité des transformations de modèles. En effet, il existe peu de techniques de validation spécifiques à ce domaine. Pourtant, le besoin de telles techniques est clairement identifié dans la littérature: une condition pour que l'ingénierie des modèles tienne ses promesses en terme de qualité est d'assurer la validation des modèles et des transformations utilisées [14].

- **Validation de modèles**

Dans le contexte du développement basé sur les modèles, il existe deux approches pour la validation des modèles. La première approche est la validation directe des modèles qui s'appuyer à vérifier ou à tester les modèles. Mais cette technique a un inconvénient est qu'elle impose de disposer d'un environnement permettant de simuler ou d'exécuter des modèles. Pour la deuxième approche tire avantage du fait que les modèles sont utilisés pour produire une partie de l'application développée. L'idée est de tester le code produit à partir des modèles plutôt que de tester directement les modèles. Cette technique a des avantages aussi des inconvénients pour avantage est qu'il n'est pas nécessaire de disposer d'un environnement modélisation exécutable. Son inconvénient est que la distance entre les modèles et le code produit est potentiellement importante et que le passage de l'un à l'autre peut introduire des erreurs [14].

- **Validation de transformations de modèles**

Dans le domaine de l'ingénierie des modèles, beaucoup travaux s'intéressent aux langages et techniques pour le développement de transformations de modèles mais parmi ces travaux très peu s'intéressent aux problèmes de la validation. Pratiquement, les transformations de modèles sont implantées comme des programmes et peuvent donc être validées comme n'importe quel autre programme. Cependant, les transformations de modèles ont la particularité de manipuler des modèles qui sont eux-mêmes décrits par des méta-modèles.

2.4 Conclusion :

Dans ce chapitre nous avons passé en vue l'ingénierie dirigée par les modèles ainsi que les techniques de transformation de modèles.