+

**République Algérienne Démocratiques Populaire**

**Ministère de l'Enseignement Supérieure de la Recherche Scientifique Université IBN KHALDOUN – Tiaret**

**Faculté des sciences et des sciences de l'Ingénieur**

**Département Informatique**

MDA

Rapport

**Module:** MOM

**Prof:** Ms Hattab

**Présenter par :**

SENOUCI FATIMA

MOSTEFAOUI Ismahen

SI MERABET Sabrina Hanan

2020

Première année Master Génie Informatique

1/10/2020

**Plant de travaille :**

1. *Introduction*
2. *Définition de le MDA*
3. LES 3 AXES DU MDA
4. Principe:
5. L'ingénierie dirigée par les modèles – IDM
6. *5. Les modèles*
7. *Architecture MDA*
8. Méta modèle
9. Infrastructure

10)UML dans MDA

11)Object Contrainte Langage

12)XMI

13)Synthèse sur la productivité

14)Conclusion

***Introduction :***

Les technologies sont en constante évolution. Afin de bénéficier des avancées technologiques, il est nécessaire d'adapter les applications à ces technologies. Or cette opération coûte cher aux entreprises, car il est souvent nécessaire de réécrire le code entièrement. Lorsqu'il n'existe pas de capitalisation des fonctions de l'application et que le développement repose généralement sur le code source, la séparation des préoccupations apparaît comme la solution nécessaire au problème. Ainsi, spécifications fonctionnelles et spécifications techniques sont prises en compte séparément par l'approche MDA.

* *« Modéliser est le futur, et je pense que les sociétés qui travaillent dans ce domaine ont raison »*

B. Gates

* *« Obtenir du code à partir d’un modèle stable est une capacité qui s’inscrit dans la durée »* R. Soley

Donc Qu est ce que l’approche MDA ?

**Definition:**

* 1. MDA est une abbreviation de :M= Model, D= Driven, A= Architecture

1-2)MDA est une nouvelle proposition de l’OMG (*Object Management Group*) en 2000 dont l’objectif est la conception de systèmes basée sur la seule modélisation du domaine, en faisant abstraction des aspects technologiques. À partir de cette modélisation, MDA propose d’obtenir par transformation les éléments techniques capables de fonctionner au sein d’une plateforme logicielle comme Java ou .NET.

Dans MDA, le modèle des objets du domaine s’appelle PIM, c’est-à-dire *Platform Independent Model* ou modèle indépendant de la plateforme. Le PIM est constitué d’un ensemble d’éléments dont la conception doit se faire indépendamment de tout langage de programmation ou de technologie. Ce modèle est ensuite transformé manuellement ou automatiquement en un modèle spécifique à une plateforme et à un langage de programmation. Un tel modèle spécifique s’appelle PSM, c’est-à-dire *Platform Specific Model* ou modèle spécifique à la plateforme.

Le lien avec UML réside au niveau du PIM. En effet, UML est un très bon candidat comme langage à ce niveau. UML possède l’avantage de décrire finement des objets tout en restant indépendant des technologies. Au niveau des traitements, l’extension Action Semantics d’UML devrait lui permettre de répondre à tous les besoins de description. ...

**LES 3 AXES DU MDA :**

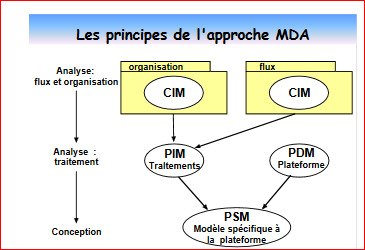
**Donc pour quoi utilisons on le MDA ?**

« Un bon schéma vaut mieux qu’un long discours … sauf qu’à un schéma (UML) correspond plus d’un long discours ! »

**Principe:**

Le principe de MDA est de séparer les spécifications fonctionnelles des spécifications de l'implantation sur une plate-forme donnée => interopérabilité des applications L'idée centrale de MDA est d'élaborer des modèles, d'abord d'analyse puis de conception, jusqu'au code, par transformations, dérivations et enrichissements successifs L'OMG propose le langage déclaratif (à base de règles) "QVT" (Query/View/Transformation) pour exprimer les transformations de ces modèle .

**Besoin de bonnes pratiques et d’objectifs précis :**



**Objectifs :**

-Élaboration de nouvelles applications

-Évolution d’applications existantes

-Maîtriser l’impact des nouvelles technologies

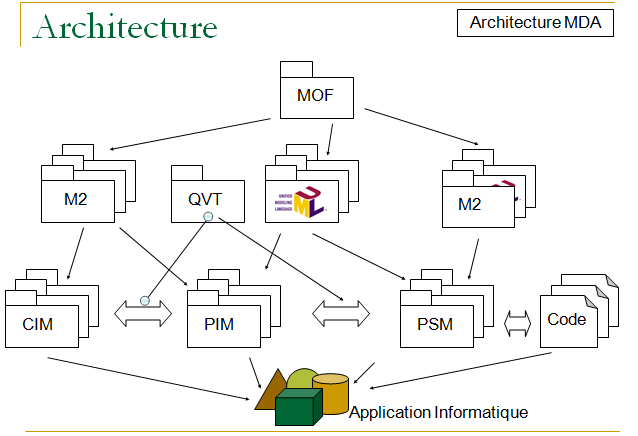
**L'ingénierie dirigée par les modèles – IDM :**

Avant de parler plus en détail du sujet qui nous intéresse ici, à savoir l'approche MDA, énumérons les concepts sous-jacents sur lesquels est bâtie cette approche.

La modélisation est la représentation de l'information à différents niveaux d'abstraction. Dans le cas d'une application informatique, un modèle permet de représenter, à chaque étape du développement, certaines caractéristiques de l'application, car développeurs, analystes et architectes n'ont pas les mêmes besoins de connaissance de l'application et de son environnement. L'IDM est la mise en œuvre de la modélisation dans le domaine du développement logiciel.

Sur la base de la modélisation et des méthodologies de conception basées sur les modèles, l'OMG propose l'approche MDA, dérivée de l'IDM. De par ce fait, l'approche MDA est construite sur les mêmes bases que l'IDM, à savoir : le métamodèle, le modèle et la transformation de modèle. Chacun de ces concepts est détaillé dans les sections qui suivent.

***Architecture MDA :***

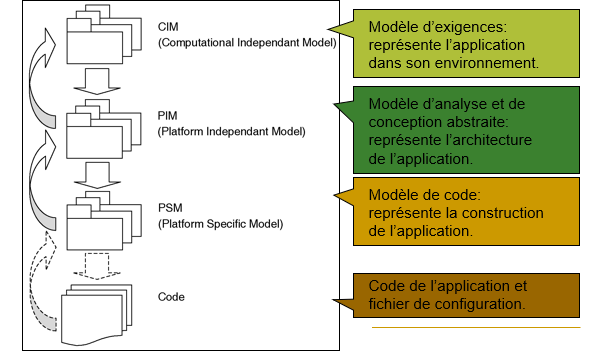
******

***5.***

***Les modèles***

L'approche MDA distingue deux aspects principaux dans le processus de développement d'une application, l'aspect métier qui représente les fonctions de l'application, et l'aspect technique qui représente la technologie de mise en œuvre de l'application. Chaque aspect est exprimé par un ensemble de modèles, qui véhiculent l'information nécessaire à la génération du code source de l'application. On passe d'une vue contemplative des modèles à une vue productive. MDA définit trois niveaux de modèles représentant les niveaux d'abstraction de l'application, le CIM, le PIM et le PSM :

***Architecture MDA :***



**Le CIM :**

!est le modèle d'analyse de base du métier ou du domaine d’application

!est indépendant de tout système informatique

!décrit les concepts de l'activité métier, le savoir faire les processus, la terminologie et les règles de gestion (de haut niveau)

!décrit la situation dans lequel le système est utilisé !n'est modifié uniquement que si les connaissances ou les besoins métier changent (très longue durée de vie) 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 :***

!est un modèle de conception

!décrit le système indépendamment de toute plate-forme technique et de toute technologie utilisée pour déployer l’application

!représente la logique métier spécifique au système (fonctionnement des entités et des services) !est pérenne dans le temps

!consiste en des diagrammes UML de classes (avec des contraintes en OCL) Les différents niveaux de PIM précisent les choix de persistance, de gestion des transactions, de sécurité...

***Un PDM :***

! contient des informations pour la transformation des modèles vers une plateforme ! est spécifique à une plateforme ! est un modèle de transformation pour permettre le passage du PIM vers le PSM

***Un PSM :***

! sert à la génération du code exécutable pour les plates-formes techniques particulières

! décrit comment le système utilisera la plate-forme

! est dépendant de la plate-forme

**Niveaux de PSM :**

Le premier niveau, issu de la transformation d’un PIM par l'adaptation des modèles UML aux spécificités la plate-forme Les autres niveaux PSM sont obtenus par transformations successives en prenant en compte le langage (Java, C#, PHP...), les choix de conception... Le dernier niveau, ou PSM d’implantation, décrit, en autres, le code du programme, les schémas des tables, les bibliothèques utilisées, les descripteurs de déploiement...

**Code source :**

Représente le résultat final du processus MDA, le code source est obtenu par génération automatique (partielle ou totale) du code de l'application à partir du PSM. Le code source obtenu peut toujours être enrichi ou modifié manuellement.

**Transformations CIM vers PIM:**

Les modèles CIM expriment les besoins des utilisateurs. Cette étape consiste à construire, partiellement, des modèles PIM à partir des CIM. Le but est de retranscrire les informations contenues dans les CIM vers les modèles PIM. C'est ce qui va permettre de s'assurer que les besoins de l'utilisateur sont véhiculés et respectés tout au long du processus MDA.

**Transformations PIM vers PIM :**

Les modèles PIM modélisent l'aspect structurel et dynamique d'une application. Cette étape s'exprime par l'enrichissement des modèles PIM. Enrichir des PIM consiste à leur rajouter de l'information utile et à spécifier leur contenu. Les PIM sont ainsi raffinés et les informations qu'ils contiennent, précisées.

.**Transformations PIM vers PSM :**

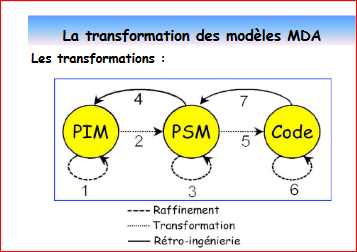
Cette étape consiste à créer des modèles PSM à partir des informations fournies par les modèles PIM, en y ajoutant des informations techniques relatives à la plateforme d'exécution cible. C'est ainsi que le lien avec la plateforme d'exécution se forme. Un PSM fournit les informations utiles à la génération du code de l'application et est dépendant de la plateforme d'exécution. Il est possible de créer autant de PSM qu'il y a de plateformes cibles.

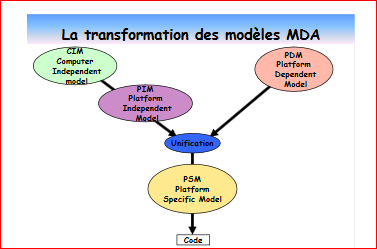
**Transformations PSM vers code :**

La transformation des modèles PSM en code source consiste à générer le code source de l'application, de façon totale ou partielle, à partir des modèles PSM de l'application. Cette étape n'est pas à proprement dit considérée comme une transformation par MDA. En effet, une transformation selon MDA est définie par la transformation d'un modèle vers un autre modèle, chacun d'eux étant structuré par leur métamodèle. Or le code source n'a pas de métamodèle, la transformation des PSM vers le code est plutôt considérée comme une retranscription textuelle du modèle PSM.

**Transformations inverses**

Transformations inverses ou rétro-ingénierie, signifie la construction de modèles à partir d'applications existantes. Dans ce cadre, MDA identifie aussi les transformations inverses : Code vers PSM, PSM vers PIM et PIM vers CIM.





**Métamodèle**

Un métamodèle est essentiellement la définition d’un ensemble de concepts et de leurs relations à l’aide d’un diagramme de classes.

Un métamodèle ne définit que la structure et pas la sémantique.

Un modèle est une instance d’un métamodèle s’il respecte la structuration définie par le métamodèle.

Le métamodèle UML définit les concepts UML et leurs relations. Un modèle UML doit respecter la définition du métamodèle.



**Métamétamodèle :**

-Le MOF définit le langage permettant de définir des métamodèles

-Les concepts du MOF sont les concepts de métaclasse, méta-attribut, méta-association, etc.

-MOF peut être défini à l’aide d’un diagramme de classe. Ce diagramme est le métamétamodèle

-Le métamétamodèle s’auto-définit.



**Niveaux Méta :**

-Les relations entre les niveaux méta sont des relations de définition de structure

-Les relations entre les niveaux méta ne sont pas des relations d’abstraction.

-Les relations entre les niveaux méta sont semblables aux relations entre les grammaires (BNF, ou XML Schema)

**Infrastructure 2.0 :**

-UML définit les concepts nécessaires à l’expression des diagrammes de classe

-MOF définit les concepts nécessaires à l’expression des diagrammes de classe

=> Capitaliser sur les concepts nécessaires à l’expression des diagrammes de classe

**Infrastructure :**

**UML dans MDA :**

-UML permet principalement de construire des modèles d’applications informatiques indépendants des plates-formes d’exécution (phase d’analyse et de conception abstraite)

-UML est donc le métamodèle naturel pour les PIM (Platform Independant Model)

-UML permet aussi de représenter une application dans son environnement afin d’en faire ressortir les exigences (cas d’utilisation)

-UML peut être utilisé pour les CIM (Computational Independant Model)

-UML peut être profilé afin de cibler des plates-formes d’exécution (ex: profil pour CORBA, profil pour EJB)

-UML peut être utilisé pour les PSM (Platform Specific Model)

Il serait donc possible d’appliquer MDA en utilisant uniquement UML

**Object Contrainte Langage :**

-OCL définit la structuration des modèles représentant des contraintes sur les applications informatiques

-Invariant, Pré-post conditions

-OCL est un métamodèle instance de MOF

-OCL est fortement couplé au métamodèle UML

-OCL définit sa sémantique à l’aide d’un métamodèle (opération sans effet de bord)



**Action Sémantiques :**

-AS définit la structuration des modèles représentant des séquences d’instructions

**XMI :**

**XMI et UML :**

-XMI se décline en 6 versions et UML se décline en 4 versions

-Cela explique pourquoi l’échange de modèle UML en XMI pose quelques problèmes

**Transformation de modèles :**

-Les transformations de modèles sont le cœur des aspects de production de MDA

CIM vers PIM, PIM vers PSM, PSM vers code (sens inverse).

-Les transformations de modèles sont basées sur les métamodèles

-Tout composant UML se transforme en une classe PHP.

-Les constructeurs de plate-forme devraient produire les transformateurs permettant d’exploiter les plates-formes

-UML vers EJB

-Les sociétés devraient pouvoir adapter les transformateurs selon leurs propres expériences et leurs propres besoins

-Ex: ne pas utiliser de composant EJB Entity!

-Actuellement les transformations de modèles peuvent s’écrire selon trois approches

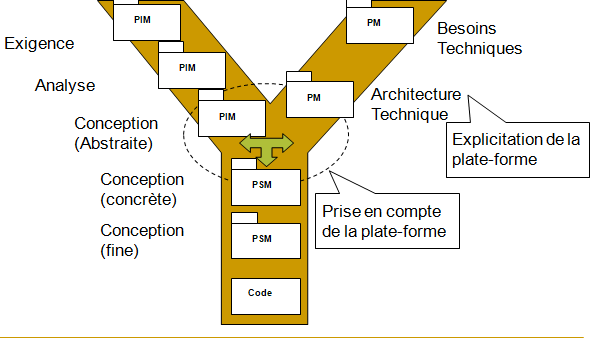
-Programmation, Template, Modélisation

**Synthèse sur la productivité :**

MDA fait passer les modèles du contemplatif au productif

-Les transformations de modèles sont réalisables même si actuellement seule l’approche par programmation est pleinement exploitable.

**Cycle en Y et plate-forme :**



**Conclusion :**

-MDA entre dans une phase d’industrialisation

-La pérennité est aujourd’hui totalement atteinte grâce aux standards MOF, UML, OCL, AS et XMI

-La productivité des modèles est une réalité. Les progrès annoncés laissent entrevoir un essor considérable (MOF2.0 QVT)

-La prise en compte des plates-formes reste encore trop à la charge de celui qui met en place l’approche MDA