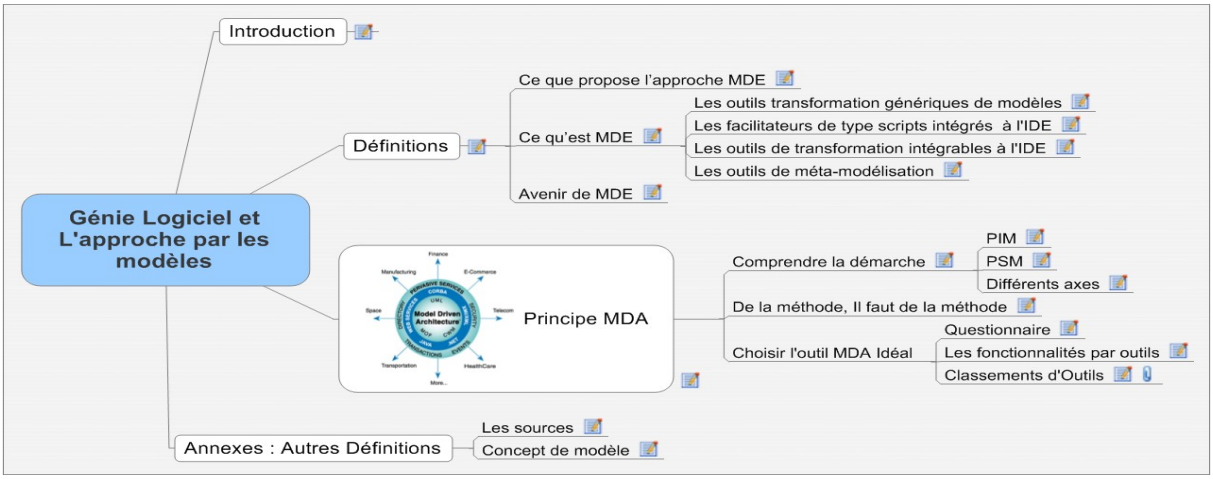
Génie Logiciel et L'approche par les modèles



1Introduction

Durant la dernière décennie, la complexité des systèmes informatiques s'est beaucoup accrue, notamment, en matière d'exigences, de sécurité, de qualité, ou encore avec la multiplication des canaux de communication largement portés par l'avènement du web 2.0, mais surtout des besoins en mobilité.

Paradoxalement, malgré cette complexité, la part croissante des acteurs de tout bord, dans leur implication au système d'information de leur entreprise, donne l'impression réelle ou non, d'une intégration globale des savoirs faires au service d'un même projet d'entreprise. Face à ce challenge, la seule arme véritable mais pas des moindres : La montée en puissance des Nouvelles Technologies de l'Information et de la Communication (NTIC) et leurs interactions avec le monde du génie logiciel. Cette interaction est le chaînon qu'il manquait, à tous les managers informatiques, soucieux d'investir dans de bonnes pratiques de partages d'une information polymorphes. Il s'agit de rendre à l'homme son pouvoir de formalisation (ou d'expression de son intelligence métier), indépendamment de la technologie finale, qui a trop longtemps polluée la communication entre population d'horizon différents.

> « Le génie logiciel » se résume en l'industrialisation de la production logicielle.

Sans révolutionner le contenu de cette science dans ses préceptes fondamentaux, il est néanmoins acquis, la nécessité d'en améliorer la mise en oeuvre opérationnelle. C'est au domaine de la méthodologie que revient cette difficile tâche. MDA serait-elle une méthode ?

> ***Objectif des DSI :*** Maîtriser / Minimiser les coûts de développement. Comment en faire plus avec moins ?

Comme dans les autres sciences, l'observation, la formalisation, l'enrichissement, et la transformation, guide la démarche. Ce processus se nomme : « La modélisation » ou plus largement « La formalisation ». La modélisation facilite la maîtrise de la complexité, tant en conception qu'en validation.

Mais cette démarche, qui se veut aussi rigoureuse, qu'efficace, permet elle de définir en une seule fois, un concept, de façon précise, concise, simple, répétitive, et reproductible. Une simple coquille de noix est elle suffisante pour accueillir toute la substantifique moelle informationnelle nécessaire à l'édition d'un logiciel moderne ?

C'est le défi relevé par l'approche MDE, et plus particulièrement, par sa variante MDA.

> « MDA est au Système d'information, ce que l'épistémologie est à la science : Un pont entre les îlots de savoir »

Après avoir expliqué ces deux démarches complémentaires, nous instruirons une grille, résumant les critères qu'il faut vérifier, pour trouver en théorie l'outil idéal de développement poussé par les modèles, en fonction des besoins exprimés et de l'aptitude à en suivre les préceptes.

2Définitions

MDE ou "Ingénierie dirigée par les modèles" est le domaine de l'informatique mettant à disposition des outils, concepts et langages pour créer et transformer des modèles.

MDA (Model Driven Architecture) est une variante concrète de MDE, appliqué particulièrement au génie logiciel du monde ouvert. C'est une démarche de réalisation de

logiciel, proposée et soutenue par l'OMG. C'est une variante particulière de MDE... D'autres variantes MDE ont été développées, par exemple par Microsoft (DSL Tools).

Langage : UML, MOF, QVT

Mots-Clés : Modèle, Métamodèle, Métamétamodèle, Diagramme de classe…

2.1Ce que propose l'approche MDE

MDE propose simplement de mécaniser le processus que les ingénieurs expérimentés suivent à la main. L'intérêt pour MDE a été fortement amplifié par l'initiative MDA que l'OMG (Object Modeling Group) a rendu publique, et qui peut être considéré comme une spécialisation de MDE dans la gestion des dépendances d'un logiciel à une plateforme d'exécution.

MDE est une forme d'ingénierie générative, qui se singularise par le fait que tout ou partie d'une application informatique est générée à partir de modèles, comme la programmation générative, la programmation orientée aspects, les usines à logiciel (Software Factories) où encore le MIC (Model Integrated Computing).

Pour cela il faut bien sûr que, les modèles et les processus de tissage de la conception soient suffisamment précis et formels, pour être interprétés ou transformés par des machines.

Le processus de conception peut alors être vu comme un ensemble de transformations de modèles partiellement ordonné, chaque transformation prenant des modèles en entrée et produisant des modèles en sortie, jusqu'à obtention d'artéfacts exécutables. Ainsi, quand on doit dériver un nouveau produit, qu'il soit une simple évolution d'un produit existant ou une nouvelle variante, on peut se contenter de « rejouer » automatiquement la plus grande partie du processus de conception, en changeant simplement quelques détails ici et là.

2.2Ce qu'est MDE

Dans le contexte MDE, la notion de transformation de modèle joue un rôle fondamental ; aussi de nombreux outils, tant commerciaux que dans le monde de l'open source sont aujourd'hui disponibles pour faire la transformation de modèles. On peut grossièrement distinguer quatre catégories d'outils :

Les outils de transformation génériques de modèles.

Les facilitateurs de type scripts intégrés à l'IDE.

Les outils de transformation intégrables à l'IDE.

Les outils de méta-modélisation.

2.2.1Les outils transformation génériques de modèles

Dans la première catégorie on trouve notamment d'une part les outils de la famille XML, comme XSLT ou Xquery, et d'autre part les outils de transformation de graphes (la plupart du temps issus du monde académique). Les premiers ont l'avantage d'être déjà largement utilisés dans le monde XML, ce qui leur a permis d'atteindre un certain niveau de maturité.

En revanche, l'expérience montre que ce type de langage est assez mal adapté pour des transformations de modèles complexes (c'est-à-dire allant au-delà des problématiques de transcodage syntaxique), car ils ne permettent pas de travailler au niveau de la sémantique des modèles manipulés mais simplement à celui d'un arbre couvrant le graphe de la syntaxe abstraite du modèle ce qui impose de nombreuses contorsions qui rendent rapidement ce type de transformation de modèles complexes à élaborer, à valider et surtout à maintenir sur de longues périodes.

2.2.2Les facilitateurs de type scripts intégrés à l'IDE

Dans la seconde catégorie, on va trouver une famille d'outils de transformation de modèles proposés par des éditeurs d'ateliers de génie logiciel (AGL).

L'intérêt de cette catégorie d'outils de transformation de modèles est d'une part leur relative maturité et d'autre part leur excellente intégration dans l'atelier de génie logiciel qui les héberge. Leur principal inconvénient est le revers de la médaille de cette intégration poussée : il s'agit la plupart du temps de langages de transformation de modèles propriétaires sur lesquels il peut être risqué de miser sur le long terme. On choisira un éditeur stable, porteur d'une vision d'avenir sur les langages de modélisation, reconnu par ces pairs, ayant pignon sur rue, et présent dans les grandes institutions de la profession.

2.2.3Les outils de transformation intégrables à l'IDE

Cette catégorie regroupe les outils conçus spécifiquement pour faire de la transformation de modèles et prévus pour être plus ou moins intégrables dans les environnements de développement standard. Ces langages spécifiquement conçus pour la transformation de modèle ont l'avantage de permettre d'exprimer très simplement les transformations de modèles simples (comme par exemple des transcodages syntaxiques), mais ils trouvent rapidement leurs limites lorsque les transformations de modèles deviennent complexes du point de vue algorithmique, ou bien lorsqu'il faut gérer de nombreuses variantes (contexte des lignes de produits) et les faire évoluer et les maintenir sur de longues périodes.

2.2.4Les outils de méta-modélisation

La dernière catégorie d'outils de transformation de modèles est celle des outils de méta-modélisation « pur jus » dans lesquels la transformation de modèles revient à l'exécution d'un méta-programme. La transformation de modèles revient alors à l'exécution d'un méta-programme orienté objet, pour lequel il est relativement aisé de construire un Framework facilitant la gestion et la maintenance de nombreuses variantes de transformations nécessaires au contexte des lignes de produits.

2.3Avenir de MDE

Même s'il existe une longue expérience de l'utilisation de l'ingénierie des modèles dans certains domaines comme les télécoms, sa généralisation à l'ensemble de l'industrie n'en est qu'à ses débuts.

Solution d'avenir pour une gestion du processus de génie logiciel contrôlé, elle requiert, en contrepartie, un effort d'abstraction plus important de la part des développeurs mais aussi de toute personne intégrant le projet.

Elle permet de conserver/capitaliser le savoir faire de conception proche des centres de décision, grâce aux économies d'échelle dues à l'automatisation.

3Principe MDA

Le principe de base de MDA est de pouvoir élaborer des modèles de conception logicielle, par transformations, enrichissements, et générations successives, dans le sens d'une technicité et d'un formalisme croissant. Ainsi pour décrire son métier, ses processus, et ses données associées, il ne serait pas nécessaire dans l'absolue, de connaître un formalisme technique particulier. L'homme de métier devra seulement s'aligner sur un formalisme suffisant pour exprimer clairement les contraintes et nuances de son métier. L'élaboration d'un diagramme des concepts saillants et/ou de cas d'utilisations en UML, ou d'un diagramme de description d'un processus en BPMN ou encore l'approche MAP, doit permettre de faciliter la tache de validation des besoins par la MOA et l'AMOA de façon quasi indépendante de la MOE, et aussi formellement que possible, sans la moindre pollution liée à la technicité de telle ou telle plate- forme de production cible.

Proche des considérations et des contraintes métier d'une MOA le PIM (Platform Independent Model) se résume à l'élaboration d'un modèle des concepts saillants de l'activité métier. Le vocabulaire précis y est fixé, et les règles de gestion de haut niveau y sont mentionnées.

La seconde étape est l'enrichissement technique des PIM, afin d'obtenir par génération le niveau PSM (Platform Specific Model, PSM). C'est dans ce modèle que l'on va créer de l'adhérence avec tel ou tel langage de développement.

Les techniques employées sont donc principalement:

- De l'enrichissement spécifique des PIM par paramétrage de la cible technique.

- De la transformation de modèles pour importer / exporter vers une tiers partie.

- Puis la dernière étape est de générer tout ou partie du code de l'application, en présence de tous les modèles intermédiaires précédemment conçus.

De cette démarche, naît la notion de données projets intégrées de bout en bout, source d'économie en terme de documentation, par l'élimination naturelle de la redondance polluante d'informations, et par la réutilisation de toute brique informative, d'une étape projet à une autre, d'un profil de population à un autre.

3.1Comprendre la démarche

Pour être efficace, la démarche MDA doit être appréhendée au plus tôt par chacun des membres de l'équipe projet. Dès la note de cadrage d'un projet, à réception de la lettre de nomination du chef de projet souhaitant appliquer cette démarche dans son équipe, MDA doit/ peut entrer en action. MDA est avant tout une démarche, un état d'esprit et une boite à outil, au service d'une méthodologie, d'une gestion de livrable, d'une conduite de projet...

Par exemple au moment ou j'écris ces quelques lignes, j'ai moi même une démarche MDE puisque j'utilise un outil permettant d'organiser mes idées, mon raisonnement, ma démarche, sans avoir à me préoccuper du format final dans lequel vous lirez ces mêmes lignes. Vous les lirez probablement au format WORD, PPT, PDF... que je choisirai le moment venu.

Le demandeur de mes travaux, à lui-même sans forcement le savoir une démarche MDE, puisqu'il ne m'impose pas d'autres contraintes que celle du résultat, d'un livrable, dans un

format qu'il sait lire. Son intérêt est d'avoir le meilleur résultat possible. Pour cela il favorise mon confort et l'expression de mes idées.

On comprend bien dès lors que tous les acteurs d'un projet qui apportent une valeur ajoutée tangible, sur leur axe d'expertise métier, sont concernés par le mouvement.

MDE/A va leur permettre a chacun de :

- Contribuer au projet en livrant des informations pertinentes au format de son choix.

- Puiser dans le référentiel projet, les informations administrées et régulées, traduites le cas échéant, dans son format favori, pour permettre de construire itérativement la dimension sur laquelle on est mandaté.

3.1.1PIM

Concrètement, un modèle PIM (Plateform-Independent Model) est un modèle conceptuel indépendant de toute considération liée à la plate forme d'implémentation cible, à un langage particulier ou plus généralement, à une technologie.

En UML, on peut rapprocher ce modèle à ceux des concepts saillants, des diagrammes de cas d'utilisation, et des diagrammes de collaboration pour la description des processus. (*Précisons que les cas d'utilisation sont peu formelle est apporte un confort de compréhension, plus utile à l'homme qu'à sa machine.*). On y retrouvera aussi le dictionnaire, et la définition des règles métiers.

L'on s'attache à préciser à grosses mailles les entités du système que l'on cherche à formaliser, expliquer, concevoir.

Dans l'approche Merisienne, Le modèle entité-relation (MCD) est à cheval entre un PIM et un PSM, dans la mesure ou la seule finalité physique de cette représentation, et le Modèle physique de données (MPD). Il n'y a pas dans MERISE, la notion de stéréotype (métaphore), permettant d'élargir la cible sémantique de telle ou telle représentation graphique. En UML, un classeur générique peut être une classe, un acteur, un cas,...

Aujourd'hui, il est acté qu'à partir d'un diagramme de classe UML, niveau PIM, il est possible de déduire avantageusement le modèle physique de données (niveau PSM). En effet le diagramme de classe (stéréotypé Entité) remplace avantageusement un diagramme entité-relation en fournissant un support de communication plus riche, et plus humain. Néanmoins et pour une question de culture, la philosophie MDA encouragera l'usage des artéfacts fossiles comme le MCD, aussi longtemps qu'il y aura une majorité qui le plébiscitera. MDA veut réunir tout le monde autour de la table.

3.1.2PSM

Concrètement, un Modèle Spécifique Plateforme est lié à un système, un langage, ou une technologie particulière, contrairement au PIM, vu précédemment.

Paramétré à partir de l'outil L4G choisit, Cette étape est indispensable à l'implémentation du système cible en mode MDA.

En fonction des choix spécifiques que l'on sera amené à faire pour la couverture applicative du système, comme par exemple : Le langage java, la Base Oracle, l'OS linux, et la Machine AS400...), le modèle PSM sera déduit du modèle PIM, et prendra en compte ce niveau d'information pour spécifier la solution.

A partir du PIM des entités métiers saillantes du système, par exemple l'entité "Person", on est en mesure d'obtenir:

- Person.java contenant les attributs et leurs accesseurs, les méthodes equals() et hashCode(), un identifiant d'instance dans sa base de persistance. En général, cette classe java d'entité, est abstraite, c'est à dire qu'elle ne peut pas être instanciée. La classe fille concrète est alors générée par MDA (PersonImpl.java). Celle ci est instanciée par son entité mère Person à l'aide d'une méthode assurant sa fabrication, implémentant pour l'occasion le fameux design pattern "Abstract Factory" d'Erich Gamma

- PersonImpl.java: la classe concrète héritant du concept, permettant au developpeurs d'apporter toutes les spécifications souhaitées au concept sous-jascent.

- PersonDao.java: C'est la classe "Interface" spécifique d'accès aux données contenant les primitives principales de type CRUD.

- PersonDaoBase.java: implémente concrètement les méthodes CRUD spécifiques pour l'interface PersonDao.java.

- PersonDaoImpl.java: pour une extension concrète de la classe PersonDaoBase.java, afin de permettre toute spécification jugée utile, surchargeant le comportement commun de PersonDao.java.

- Enfin Person.hbm.xml: C'est le fichier Hibernate qui mappe l'entité Person avec sa représentation de persistance en base de données par exemple.

- Le fichier de création de la table dans la base cible PERSON au format DDL.

A la lecture des lignes précédentes, le béotien comprendra mieux l'intérêt du PIM qui lui permettrait de s'affranchir de toute considération qu'il jugerait "Barbare"

A partir de cet exemple, il faut se demander comment l'on compte s'organiser pour produire les spécifications attendues dans les squelettes générés:

- Le Développeur va spécifier ses besoins à l'intérieur de balises générées dont le contenu sera scrupuleusement conservé sous forme de TEXTE sans sémantique particulière.On écrit directement dans les classes générées, mais quid en cas de re-génération ?

- On écrit les spécifications sous forme d'artéfacts proposé par l'outil, dans un langage de haut niveau par le truchement des spécifications dynamiques. (Diagramme de séquences, Diagramme d'état, ou encore diagramme d'activité).

3.1.3Différents axes

De tout temps, les projets les plus complexes ont requis une quantité importante d'information, apportées par des personnes différentes, ayant des compétences différentes, sur des phases et des moments différents. Aujourd'hui le management moderne sait que pour être efficace, il faut pouvoir gérer conjointement toutes ces compétences. Le pilotage de Projet de type MDE est là pour organiser l'approche multi-axiale, des compétences, de façon fluide, en favorisant au maximum les formalismes de chacun. Les uns feront de la spécification de contrainte en OCL, d'autres établiront le schéma d'un processus en BPML, certains préféreront utiliser UML, d'autres coderont des spécifications en java. Enfin les uns et les autres s'attacheront à éditer des notes en langage humain, avec l'éditeur de leurs choix pourvu

qu'elles soient rattachées à des concepts formels préalablement modélisés, et eux même multi-dimensionnellement documentés.

Au final on obtient un creuset d'informations d'horizon bien différentes, toutes mues au service d'un même projet d'entreprise, formalisées dans un format standard unique: Celui du projet, compréhensible et modifiable, par tous. Le projet a son cycle de vie, et avance par itérations et générations successives. A tout moment l'état d'avancement peut avancer ou reculer en fonction du degré de maturité du projet et de la compréhension qu'en ont chacun des contributeurs.

Pour cela, il faut aussi choisir un langage de communication humain (Le Français et/ou l'Anglais).

Il faut choisir aussi un format d'échange automatique. On choisit naturellement le dialecte XMI (C'est un XML) permettant une expression textuelle formelle du méta-modèle d'UML, et des concepts métiers spécifiques au présent projet. XMI est à UML, ce que BPMN est à BPEL : Un format XML d'échange auto documenté par son modèle, sous forme de balise, et formalisé par son Métamodèle : Le schéma XSD, lui même un document XML, auto documenté par son modèle : Le schéma de schéma. On se situe alors au niveau du meta meta-modèle, par rapport à notre besoin d'origine, décrit en XMI.

Énumérons pèle-mêle, les axes de contributions suivant :

- Un axe Chef de Projet, pour la mise en place de l'organisation, la gestion du planning, des indicateurs de performance...

- Un axe analyse métier

- Un axe Développeur (*"Model Centric" et non plus "Code Centric"*)

- Un axe DBA, pour les problématiques de persistances, et d'optimisation des accès aux données.

- Un axe de savoir-faire Recette & Homologation

- et pourquoi pas un axe Ressources Humaines...

MDA permet une gestion multi-axiale d'un projet aussi confortablement que possible

3.2De la méthode, Il faut de la méthode

En s'appuyant sur les préceptes méthodologiques (UP / Praxeme, par exemple), il est intéressant de comprendre qu'à la croisée des phases et des discipline, s'enracinent des portions de processus de fabrication dont le résultat tangible peut être obtenu par MDA.

Chacune des disciplines classiques (Expression des besoins, Spécifications, implémentation, mise en oeuvre, Test, déploiement, Gestion de la configuration, Direction de projet...), constituent autant d'axes de travail, occupés par un ou plusieurs acteurs. Ces axes ont leurs intérêts et leur légitimité, dans la méthode. L'approche MDA dans cet exemple va faciliter le travail collaboratif des différentes disciplines, en organisant un référentiel projet universel, une sorte de creuset concentrant tous les savoir-faires des contributeurs dans une organisation en étoile, un peu à l'image d'un concentrateur solaire, captant les rayons lumineux des 4 coins cardinaux.

Tout naturellement, et parce qu'elle est la boite à outils privilégiée des méthodes agiles, on utilisera UML, pour modéliser, maintenir et standardiser, dans un langage commun, les différents artéfacts, et organiser leurs livraisons documentées. La langue humaine (française, anglaise) doit également faire l'objet d'un choix afin de décrire au mieux les artéfacts successivement formalisés. Traditionnellement, est choisie la langue maternelle de la majorité des contributeurs, et on choisit l'anglais pour l'expression du vocabulaire technique. Les notes et autres documents humain et informels, doivent être rattachés à l'objet métier formalisé dans le référentiel partagé. Sa masse devrait être limitée au minimum, selon une métrique 1/3 2/3. Trop souvent les projets sont formalisés en phases amont à hauteur d'1/10 et les 9/10 sont des informations humaines, interprétables, tronquées, incomplètes et parfois même, sujettes à caution.

Quand on décrit un concept en langage humain, les plus bavards d'entre nous, se laisserons bon gré mal gré, déborder par des considérations passionnées (C'est le cas des quelques lignes ci dessus). Ce risque de bavardage en pilotage par les modèles n'existe plus. Le nécessaire et suffisant devient la règle. Le formalisme supprime toute forme d'interprétation possible.

En revanche il est admis par le corps professoral, que la bonne compréhension des messages humains, passe par la multiplication des définitions qu'un enseignant serait capable d'en faire. La machine, contrairement à nous n'a besoin que d'une et une seule définition mais formelle. Ce pose alors en aval, la problématique de validation...

Le formalisme au plus tôt est le facteur de réussite de la démarche MDA.

3.3Choisir l'outil MDA Idéal

3.3.1Questionnaire

Je compte utiliser quels diagrammes UML ?

Quelques uns seulement pour une description de haut niveau?

Le plus possible pour une description jusqu'au bout des ongles ?

Est ce que je veux générer du code ? Si oui, lequel ?

- Du code spécifique (Java, .net, C++, Cobol)
- Plusieurs à la fois

Est ce que je veux pouvoir générer en partie mes documents de spécifications à partir de modèle ? Si, oui sous quel format ?

Est-ce que je souhaite pouvoir personnaliser la génération ?

Est ce que d'autres projets utiliseront ensuite UML ? (Pour permettre d'amortir le prix)

Dois-je capitaliser mes développements (approches par services & composants) ?

Aurais-je besoin d'assistance technique ?

- Outillage gratuit / ouvert (granulaire modulable adaptables) ?
- Outillage pérenne avec assistances & accompagnements garantis ?

Suis-je soumis à l'application d'une méthode (XP, UP) ou d'un standard (CMMI, ITIL) ?

Est ce que je veux couvrir tout ou partie du cycle de vie d'un projet par le processus d'industrialisation ?

- Conception
- Implémentation / Test
- Qualification / Fonctionnelle
- Vérification du respect de l'architecture imposée
- Transition vers de nouveau besoin / Maintenance applicative

Suis-je dans une logique Outsourcing ?

Suis-je sensible très sensible au confort / Visualisation / Exploration de mes modèles ?

Nombre de personnes sur le projet ? Plusieurs modélisateurs ?

Dois-je favoriser le travail collaboratif (Référentiel projet partagé par l'équipe) ?

Suis-je dans une approche Top-down vs. Approche Bottom-up ?

3.3.2Les fonctionnalités par outils

L'étude a portée plus particulièrement sur les outils suivants:

Il en existe d'autre, mais nous avons souhaité cibler les principaux ("Les pures players") capable d'accompagner avantageusement l'acteur dans sa démarche MDA.

name	version	éditeur
Enterprise Architect	6.1	Sparx System
MEGA Designer	2007	MEGA
Objecteering	6.1	Softeam
Visual Paradigm for UML	6.1	Visual Paradigm
RSA	7.0.0.2	IBM /Rational
Aris Business Architect / Designer	2007	IDS SCHEER

3.3.3Classements d'Outils

Voir la pièce jointe: Comparatif Outils MDA -YYO-JBS V3.ppt

Il est relativement difficile d'établir une liste d'outil en fonction de réponses hypothétiques, basées sur un questionnaire probablement incomplet et non exhaustif.

Néanmoins, nous retiendrons les 2 outils suivants qui ont particulièrement attirés notre attention.

Rang	name	version	editeur
1	Objecteering	6.1	Softeam
2	Enterprise Architect	6.1	Sparx System

Le grand champion est Objecteering car il propose une démarche entièrement industrialisée orientée MDA/SOA. Une grande variété d'utilitaire couvrant les scopes de la modélisation, de l'urbanisme SOA et de la conduite de projet. Objecteering a l'avantage ***exclusif*** de savoir contrôler de bout en bout, la cohérence fonctionnelle de tous les modèles, à travers quelques 250 règles de consistance. ce qui lui confère sa place de leader dans l'approche MDA / UML.

4Annexes : Autres Définitions

4.1Les sources

En particuliers les sources d'information utilisées sont les suivantes:

tool	url
Enterprise Architect	http://www.dotnetguru.org/modules.php?op=modload&name=News&file=article&sid=733
Enterprise Architect	http://www.objetdirect.com/html/produitsEtPartenaires/ea.html
Enterprise Architect	http://www.sparxsystems.com.au/
MEGA Designer	http://www.bpms.info/outil_detail.asp?ref=93
MEGA Designer	http://www.mega.com/
MEGA Designer	http://www.mega.com/index.asp/l/en/c/product/p/mega-modeling-suite/p2/mega-designer
Objecteering	http://www.softeam.fr/pdf/fr/Objecteering_EA-support_de_SEA.pdf
Objecteering	http://www.softeam.fr/references_publications_whitepapers.php
Objecteering	http://www.omg.org/mda/
Objecteering	http://www.softeam.fr/produit_objecteering6.php
Poseidon	http://gentleware.com/index.php?id=30
PowerAMC	http://www.infos-du-net.com/telecharger/PowerAMC,0301-4133.html
PowerAMC	http://www.sybase.fr/products/modelingmetadata/poweramc.shtml
RSA	http://www-306.ibm.com/software/awdtools/developer/datamodeler/
Together	http://www.borland.com/us/products/together/index.html
Win'Design	http://www.win-design.com/fr/
Win'Design	http://www.win-design.com/fr/NEW_dl_WD.htm

4.2Concept de modèle

Bien que possédant un grand nombre de définitions, un modèle en MDE est l'abstraction d'un objet sujet de l'acte de modélisation.

La modélisation, est l'abstraction, la simplification, ou encore le grossissement d'un aspect de la réalité. La modélisation en informatique s'attache à décrire de façon formelle, l'allocation optimum des ressources matérielles, nécessaire pour la gestion

d'information, et cela dans le but de créer de la valeur. Celle décrite dans le business plan.

De plus la modélisation a une vertu toute particulière ; celle de fixer le vocabulaire dès le début en phase de définition du modèle métier.

Par exemple : Les objets du monde réel comme une personne, un client, une voiture, un contrat, une commande, une pièce détachée, n'auront pas le même sens, la même sémantique, selon le métier qu'ils sont censés décrire/expliquer. Le concept "Voiture" par exemple, n'aura pas la même définition selon qu'il est destiné à décrire une voiture dans une agence de location, ou une voiture dans un garage de réparation, ou encore une voiture chez un concessionnaire qui vendrait aussi des pièces détachées.

- Pour l'agence de location, les propriétés importantes de la voiture seront la couleur, l'immatriculation et le prix de location.

- Pour le garagiste, la couleur a peu d'importance, il conçoit la voiture sûrement comme une somme de prestation de réparation à apporter.

- Le concessionnaire, lui vend des voitures et les pièces détachées. Une voiture va être composée (en fait agrégées) d'une multitude de pièces différentes, ...