وزارة التعليم العالي و البحث العلمي

BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختار - عنابة

Faculté des sciences de l'ingénieur

Département d'informatique

Année : 2010

MEMOIRE

Présenté en vue de l'obtention du diplôme de **MAGISTER**

Une Architecture Fondée Sur Les Services Pour La Prise En Compte Du Contexte Dans Les Systèmes D'information Ubiquitaires

Option

Genie Logiciel

Par

Ismail BOUACHA

DIRECTEUR DE MEMOIRE : Dr Hassina SERIDI MCA Université de Annaba

DEVANT LE JURY

PRESIDENT :	Dr Djamel MESLATI	MC	Université de Annaba
EXAMINATEURS:	Dr Halima ABIDAT-BAHI	MC	Université de Annaba
	Dr Farid MOKHATI	MC	Oum-El-Bouaki

Dédicace

Remerciements

Je souhaite, avant toute chose, remercier Dieu pour m’avoir soutenu et permis la réalisation de ce travail.

Je remercie mes parents, leur soutien et leurs conseils m’ont été d’une grande aide.

Je tiens à remercier mon encadreur Mme Seridi pour avoir dirigé ce travail. Son expérience et ses grandes compétences ont permis l'accomplissement de ce travail.

Un merci particulier pour la qualité de sa collaboration, ses nombreux conseils, et son aide constante et pour la façon efficace avec laquelle elle a suivi ce travail. J'ai beaucoup appris à son contact, pendant les nombreuses heures passées ensemble.

Pour ses précieux conseils de tout ordre, sa disponibilité et sa confiance, elle a su mettre en œuvre toute son expérience pour apporter une vision particulière et non moins importante au travail présenté dans ce mémoire. Qu'elle trouve ici les marques de ma reconnaissance et de mon respect.

Un grand merci à l’ensemble du personnel de BIG. Informatique pour leur soutien.

Je tiens à exprimer ma plus sincère gratitude à l’université Badji Mokhtar Annaba.

Enfin, un grand merci à tous mes amis et tout ceux qui sont dans mon cœurs.

ملخص

الهدف من الأنظمة واسعة الإستعمال هو جعل المعلومة متوفرة في كل وقت و في كل زمان، يجب على هذه الأنظمة أن تتوفر على القدرة على العمل حسب السياق وذلك حسب محيط المستعمل، المكان الذي يتواجد فيه

الهندسة المركزة على الخدمات تسمح للأفراد، المنضمات و الأنظمة بتوفير أو استغلال خدمات هذه الأخيرة لها وصف مجرد كما أنها منفصلة مما يعطي فوائد في الصيانة و التطوير

في الواقع هذه الهندسة يمكن أن ترتكز على خدمات الواب، مكونات، إلخ كما أنها تعتمد على ثلاث محاور رئيسية المعلومات المتبادلة، الوصف، السلوك

في السنوات الأخيرة الأبحاث المرتكزة على السياق تميل إلى استعمال الطريقة الموجهة بالنماذج من أجل استغلال السياق

لكن هذه الأبحاث ترتكز على معاني و طرق تمثيل مختلفة للمعلومات المتعلقة بالسياق إلى غاية اليوم لا يوجد نموذج لنموذج السياق

في الواقع أغلب أدوات التحويل التي تساند الطريقة الموجهة بالنماذج تابعة إلى أرضية ما و لا تأخذ بعين الاعتبار المعلومات المتعلقة بالسياق

نحن نتواجد في ميدان تمثيل الخدمات من أجل استغلال السياق باستعمال الطريقة الموجهة بالنماذج، بالتالي يمكننا استغلال المعلومات المتعلقة بالسياق من الراحل الأولى

نحن نرتكز على اليو م ل في التمثيل كما نعتمد على النموذج المقترح في من أجل تمثيل الخدمات كما نقترح التحويل الالي للنماذج المتحصل عليها خلال المرحلة السابقة

كلمات
أنظمة واسعة الاستعمال، ، خدمات الواب، السياق، التحسس من السياق، الطريقة الموجهة بالنماذج، التمثيل، التحويل

Abstract

The gool of ubiquitous system is to provide information any where, and at any time.

These systems must have the ability to be used in different contexts, switch the user environment, profile, location…

SOA is a paradigme witch allows to personnes, organistions, or systems to provide or to exploit services. These services have an abstract description witch is independent to its implementation and they are loosely coupled. This allows advantages in maintenance and evolution.

En pratique, ces architectures peuvent s'appuyer sur des services web, CORBA, RPC…, et elles sont décrites en définissant 3 axes principaux : les données d'échanges utilisées par les services (Schémas XML), les interfaces des services (WSDL) et l'orchestration de services qui est une implémentation des comportements de services (BPEL).

Ces dernières années, les travaux existant dans le domaine de la sensibilité au contexte d'utilisation se penchent vers l'utilisation de l'approche dirigée par les modèles pour prendre en compte le contexte. Cependant, ces travaux se basent sur des significations et représentations différentes des informations contextuelles, (jusqu'à présent il n'existe pas un méta modèle standard pour le contexte). Sur le plan pratique, la plupart des outils de transformations qui supportent l'approche MDA sont spécifiques à des plateformes et ne permettent pas de prendre en compte les informations contextuelles.

Nous nous positionnons dans le domaine de la modélisation des services web pour la prise en compte du contexte par une approche dirigée par les modèles. Ainsi il est possible d'exploiter le contexte dès les premières étapes du cycle de développement des services.

Nous nous basons sur la modélisation du contexte dans le langage UML, et nous s'appuyons sur le méta modèle ContextUML proposé dans [Shen05] pour la modélisation des services web. De plus nous proposons une transformation automatique des modèles obtenus à partir de l'étape de la modélisation vers la plateforme service web.

Mots clefs :

Systèmes parvasifs, SOA, Web service, Contexte, Sensibilité au contexte, Approche dirigée par les modèles, Modélisation, Transformation.

Résumé

Les systèmes pervasifs ont pour objectif de rendre l'information disponible partout et à tout moment. Ces systèmes doivent pouvoir être utilisés dans différents contextes selon l'environnement de l'utilisateur, son profil, sa localisation ...etc.

Les architectures SOA (Service Oriented Architecture) sont un paradigme permettant à des individus, organisations et systèmes de fournir ou de consommer des services. Ces derniers ont une description abstraite indépendante de leurs implémentations et ils sont faiblement couplés, ce qui offre des avantages de maintenance et d'évolution.

En pratique, ces architectures peuvent s'appuyer sur des services web, CORBA, RPC…, et elles sont décrites en définissant 3 axes principaux : les données d'échanges utilisées par les services (Schémas XML), les interfaces des services (WSDL) et l'orchestration de services qui est une implémentation des comportements de services (BPEL).

Ces dernières années, les travaux existant dans le domaine de la sensibilité au contexte d'utilisation se penchent vers l'utilisation de l'approche dirigée par les modèles pour prendre en compte le contexte. Cependant, ces travaux se basent sur des significations et représentations différentes des informations contextuelles, (jusqu'à présent il n'existe pas un méta modèle standard pour le contexte). Sur le plan pratique, la plupart des outils de transformations qui supportent l'approche MDA sont spécifiques à des plateformes et ne permettent pas de prendre en compte les informations contextuelles.

Nous nous positionnons dans le domaine de la modélisation des services web pour la prise en compte du contexte par une approche dirigée par les modèles. Ainsi il est possible d'exploiter le contexte dès les premières étapes du cycle de développement des services.

Nous nous basons sur la modélisation du contexte dans le langage UML, et nous s'appuyons sur le méta modèle ContextUML proposé dans [Shen05] pour la modélisation des services web. De plus nous proposons une transformation automatique des modèles obtenus à partir de l'étape de la modélisation vers la plateforme service web.

Mots clefs :

Systèmes parvasifs, SOA, Web service, Contexte, Sensibilité au contexte, Approche dirigée par les modèles, Modélisation, Transformation.

Sommaire

LISTE DES FIGURES 7
LISTE DES TABLEAUX 8
ABREVIATIONS ET ACRONYMES ..... **Erreur ! Signet non défini.**
GLOSSAIRE ..... **Erreur ! Signet non défini.**
Introduction générale ..... **Erreur ! Signet non défini.**7
I. Les systèmes d'information ubiquitaires 188
II. La sensibilité au contexte 188
III. Problématique 199
IV. Cas d'étude 20
V. Organisation du mémoire 20
Partie 1 - Etat de l'art 22
Chapitre 1 : Web service et Context 233
I. Introduction 233
II. Principes de SOA 23
III. Utilisation de l'approche Model Driven Architecture (MDA) pour SOA 24
IV. Services web 24
V. Les langages et protocoles utilisés par les services web 24
VI. Composition de services 30
VII. Contexte et sensibilité au contexte 32
1. Définition du contexte 32
2. Sensibilité au contexte 33
3. Modélisation du contexte 35
4. Qualité du contexte 355
5. Systèmes existants et plateformes 36
6. Travaux connexes 37
VIII. Conclusion 378
Chapitre 2 : MDA (Model Driven Architecture) 40
I. Introduction 40
II. Le cycle de vie du développement MDA 40
III. Automatisation des étapes de transformation 422
IV. Application de l'approche MDA 433
V. Méta modèles et architecture à 4 couches 444
VI. MOF et QVT 455
VII. UML (Unified Modeling Language) 455
VIII.XML Meta Data Interchage (XMI) 48
IX. Types de Transformation ..... **Erreur ! Signet non défini.**8
IX. Transformations et standards utilisés ..... **Erreur ! Signet non défini.**
XII. Conclusion 56
Partie 2 Contribution 58
Chapitre 3 : Services web sensibles au contexte 59
I. Introduction 59

II. Prolème 59
III. Objectifs 60
IV. Processus de développement des services sensibles au contexte 60
1. Modélisation des services sensibles au contexte 61
1. Transformation 63
V. Conclusion 78
Chapitre 5 Implémentation et cas d'étude 788
I. Introduction 788
II. Cas d'étude 788
III. Contexte Aware Transportation 789
IV. Processus de développement 80
1. Modélisation 80
1. Transformation 81
V. Présentation de l'application CATransportation 88
IV. Conclusion 91
Conclusion et perspectives 92
I. Conclusion 93
II. Perspectives 93
Bibliographie 944

Liste des figures

Figure 1 - Modèle fonctionnel de l'architecture de publication et d'invocation

Figure 2 - L'architecture d'un fichier WSDL

Figure 3 - Le schéma d'un fichier SOAP

Figure 4 - Le cycle de vie du développement MDA

Figure 5 - Transformations de l'approche MDA

Figure 6 - Application de l'MDA

Figure 7 - Architectures à 4 couches

Figure 8 - Transformation par annotation ou marquage

Figure 9 - Transformation par méta modèle

Figure 10 - Cycle de transformation en double Y

Figure 11 – Le diagramme de caractéristiques de langages de transformation de modèles

Figure 12 - Processus de développement des services sensibles au contexte

Figure 13 – Le méta modèle ContextUML

Figure 14 - Exemple de service sensible au contexte

Figure 15 – Transformation Horizontale et Verticale

Figure 16 – Transformation Endogène et Exogène

Figure 17 – Transformation en utilisant XSLT

Figure 18 – Service

Figure 19 - Transformation de la partie service

Figure 20 – Exemple de la transformation de la partie service

Figure 21 - Transformation de la partie contexte

Figure 22 - Modélisation du contexte

Figure 23 - Exemple de la transformation de la partie contexte

Figure 24 - Binding

Figure 25 - Transformation de la partie sensibilité au contexte

Figure 26 - Modélisation de la sensibilité au contexte

Figure 27 - Exemple de la transformation de la partie sensibilité au contexte

Figure 28 - Services de l'application CATransportation

Figure 29 - L'outil ArgoUML

Figure 30 - ContextUML

Figure 31 - Modélisation d'un service sensible au contexte

Figure 32 - Format XMI

Figure 33 - Processus de transformation

Figure 34 -Importation d'une représentation XMI

Figure 35 - Représentation XMI d'un diagramme UML

Figure 36 – Transformation avec CATransformer

Figure 37 - Résultat de la transformation

Figure 38 - CATransportation

Figure 39 - L'outil MyEclipse

Figure 40 - Service ListeDesEngins (situation contextuelle 1)

Figure 41 - Service ListeDesEngins (situation contextuelle 2)

Liste des tableaux

Tableau 1 Tableau comparatif des travaux connexes

Tableau 2 - Correspondance entre les éléments du ContextUML et les éléments du WSDL

Abréviations et acronymes

API Application Programming Interface

BPEL4WS Business Process Execution Language for Web Services

BPWS4J Business Process Execution Language for Web Services Java Run Time

CORBA Common Object Request Broker Architecture

DCOM Distributed Component Object Model

HTTP HyperText Transfer Protocol

OWL Web Ontology Language

OWL-S OWL for Services

RDF Resource Description Framework

RMI Remote Method Invocation

SOAP Simple Object Access Protocol

TCP/IP Transmission Control Protocol/Internet Protocol

UDDI Universal Description, Discovery and Integration

URL Uniform Resource Locator

CC/PP Composite Capabilities / Preference Profiles

OMG Object Management Group

MDA Model Driven Architecture

CIM Computation Independent Model

PDM Platform Description Model

PIM Platform Independent Model

PSM Platform Specific Model

HTML HyperText Markup Language

PDA Personal Digital Assistant

PUMAS Peer Ubiquitous Multi-Agent System

J2EE Java Enterprise Edition

ATL Atlas Transformation Language

BOTL Bi-Directional transformation Language

OCL Object Constraint Language

CWM Common Warehouse Meta-model

DTD Document Type Definition

EJB Enterprise Java Bean

EMF Eclipse Modeling Framework

GMT Generative Model Transformer

IDL Interface Definition Language

JSP Java Server Pages

MOF Meta-Object Facility

MTL Meta Transformation Language

QVT Query, View, Transformation

UML Unified Model Language

UMT UML Model Transformation Tool

WSDL Web Services Description Language

XML eXtensible Markup Language

XMI XML Metadata Interchange

XSL eXtensible Style sheet Language

XSLT XSL for Transformation

Glossaire

OMG (Object Management Group) est une organisation internationale créé en 1989 avec l'objectif de promouvoir la théorie et la pratique de la technologie orientée objet dans le développement de logiciels (http://www.omg.org).

MDA (Model Driven Architecture) est l'approche dirigée par les modèles proposée par l'OMG. MDA est une démarche de développement basée sur les modèles et un ensemble de standards de l'OMG. Cette démarche permet de séparer les spécifications fonctionnelles d'un système (PIM) des spécifications de son implémentation (PSM).

Modèle est « une simplification de quelque chose qui nous permet de voir, manipuler, et raisonner sur le sujet étudié, et qui nous aide à en comprendre la complexité inhérente ».

MOF (Meta-Object Facility) est un langage abstrait et un framework pour la spécification, la construction, et la gestion de métamodèles neutres de technologie.

Abstraction est « une description de quelque chose qui omet certains détails non pertinents pour l'objectif de l'abstraction » .

ATL (Atlas Transformation Language) est un langage pour la réalisation de transformations de modèles dans le contexte de MDA.

PIM (Platform-Independent Model) est un modèle indépendant d'une plate-forme technologique telles que CORBA, Services Web, J2EE et dotNET.

Processus MDA processus spécifique pour la mise en ouvre d'un projet MDA.

PSM (Platform-Specific Model) est un modèle dépendant d'une plate-forme technologique telles que CORBA, Services Web, J2EE et dotNET.

Service est une ressource abstraite qui représente des possibilités d'accomplir des tâches qui assurent une fonctionnalité cohérente du point de vue des entités fournisseur et demandeur .

Services Web « est une manière standardisée d'intégration des applications basées sur le Web en utilisant les standards ouverts XML, SOAP, WSDL, UDDI et les protocoles de transport de l'Internet.

XML est utilisé pour représenter les données, SOAP pour transporter les données, WSDL pour décrire les services disponibles, et UDDI pour lister les fournisseurs de services et les services disponibles » (Cf. Webopedia http://www.webopedia.com).

SOA (Service Oriented Architecture) est un ensemble de composants qui peuvent être appelés, et dont les descriptions d'interfaces peuvent être éditées et découvertes.

SOAP (Simple Object Access Protocol) fournit une définition des informations représentées en XML qui peuvent être utilisées pour échanger des informations structurées et typées entre les participants dans un environnement distribué et décentralisé.

UDDI (Universal, Description, Discovery and Integration) fournit la définition d'un ensemble de services qui permettent la description et la découverte (1) des entreprises, des organismes, et d'autres fournisseurs de ServicesWeb, (2) des ServicesWeb qu'ils rendent disponibles, et (3) des interfaces techniques qui peuvent être utilisées pour accéder à ces services.

Langage de modélisation est une spécification formelle bien définie qui contient les éléments de base pour construire des modèles.

UML (Unified Modeling Language) est le langage unifié de modélisation. UML est basé sur les fondements de l'orientation objets : la notion de classe, l'objet, les attributs, les méthodes et les relations entre les classes ou les objets.

W3C (World Wide Web Consortium) est un consortium international où les organismes membres, le personnel à temps plein, et le public travaillent ensemble pour développer des normes de la Web. Ce consortium est responsable par la normalisation de XML, HTTP, WSDL, SOAP, SemanticWeb, etc (http://www.w3.org).

WSDL (Web Services Description Language) fournit un modèle et un format XML pour décrire des Services Web.

XMI (XML Metadata Interchange) permet l'échange de modèles sérialisés en XML.

XML (eXtensible Markup Language) est un métalangage de représentation de données. Il définit un ensemble de règles de formatage pour composer des données valides. XML est une mise en forme de texte simple et très flexible dérivée de SGML (Standard Generalized Markup Language) (ISO 8879).

BPEL4WS (Business Process Execution Language for Web Services) est un langage de composition de services.

Composition des Services Web est un processus par lequel un service Web est créé selon un arrangement d'autres services Web.

CORBA (Common Object Request Broker Architecture) est une spécification et une architecture pour la création et la gestion d'applications orientées objet distribuées sur un réseau. La spécification de CORBA est indépendante d'une implémentation, du langage de programmation et du système d'exploitation.

DSL (Domain Specific Langage) DSLs sont languages that instead of being focused on a particular technological problem such as programming, data interchange or configuration, are designed so that they can more directly represent the problem domain which is being addressed.

Intergiciel (middleware) est un logiciel de connexion consistant en un groupe de services qui permettent l'exécution de plusieurs applications sur une ou plusieurs machines connectées en réseau. Cette technologie fournit une couche d'abstraction qui permet l'interopérabilité entre différents langages de programmation, systèmes d'exploitation et architectures d'ordinateurs.

J2EE (Java 2 Platform, Enterprise Edition) « est un framework pour le langage de programmation Java de Sun plus particulièrement destiné aux applications d'entreprise. Dans ce but, il contient un ensemble d'extension au framework standard afin de faciliter la création d'applications réparties. Voici une liste des API contenues dans J2EE : Servlets, JSP, JSF, EJB, JNDI, JDBC, JMS, JAXP, JAXM, JAX-RPC, JAXB » (Cf. Wikipédia http://fr.wikipedia.org/wiki). JWSDP utilise également les API : JAXP, JAXM, JAX-RPC, JAXB et JAXR. Java RMI (Java Remote Methode Invocation) est un intergiciel spécifique à la plate-forme Java.

Introduction générale

I. Les systèmes d'information pervasifs

Ces dernières années, l'évolution technologique dans le domaine de la communication et notamment l'utilisation des outils mobiles (tels que ordinateurs portables, téléphones mobiles, PDA…) a exigé de nouvelles préoccupations dans l'ingénierie software, et depuis l'année 1990 on parlait des systèmes pervasifs. Cette notion introduite par Mark Weiser [Weiser95] et [Weiser99] vise à rendre les ordinateurs des compagnons intelligents dont le but est de faciliter notre vie quotidienne. Par définition : « l'informatique pervasive rend l'information disponible partout et à tout moment » [Agoston00]. En conséquence, ces systèmes doivent pouvoir détecter toutes les circonstances qui entourent l'utilisateur, et fournir les meilleures réponses en fonction de ces circonstances.

Autrement dit, ces systèmes doivent pouvoir traiter des informations non fonctionnelles pour satisfaire les utilisateurs qui se trouvent dans diverses situations contextuelles, et ce traitement doit se faire sans l'intervention explicite des utilisateurs. Cette nouvelle préoccupation ouvre un nouvel axe de recherche appelé la sensibilité au contexte.

II. La sensibilité au contexte

Les recherches sur le développement d'applications sensibles au contexte furent apparues depuis les années 1990s. Ces applications sont présentes dans plusieurs domaines tels que guidage des touristes, jeux…ayant comme fonctions de : fournir les informations adéquates, assistance à la navigation dans des environnements non connues…

Dans un système sensible au contexte, nous devons prendre en compte des informations non fonctionnelles telles que : le type du terminal, le profile utilisateur connecté pour garantir une utilisation confortable des applications. Pour réaliser une adaptation au contexte, on peut distinguer plusieurs informations contextuelles [Chen00] :

- Informations sur le terminal utilisé : la conception de ces systèmes est influencée par la diversité des terminaux. Un système sensible au contexte doit s'adapter aux capacités matérielles et logiciel de ces appareils.

- Informations sur l'utilisateur : l'utilisateur est devenu le point central de la conception des systèmes sensibles au contexte. Il faut donc prendre en considération ses préférences, son emplacement géographique, son profil…

- Autres informations: tels que les paramètres reliés à l'environnement (par exemple le climat, la température,…etc), les paramètres réseaux (la capacité de la bande passante, les connexions…ect)…etc.

III. Problématique

Les travaux existants dans le domaine de la sensibilité au contexte, se focalisent sur la création incrémentale ou sur le prototypage d'applications sensibles au contexte en incorporant le code d'adaptation dans le code métier de l'application. Ceci limite les capacités à prendre en compte de nouveaux contextes qui n'auraient pas été prévus lors de son développement. La majorité de ces travaux intègrent des règles si-sinon pour décrire comment les systèmes sensibles au contexte doivent réagir aux changements contextuels. Ces règles sont statiques et ne couvrent pas la totalité des informations contextuelles, par exemple si de nouvelles informations contextuelles apparaissent il faut ajouter d'autres règles pour les prendre en charge. Il est à noter aussi que les informations contextuelles se diffèrent d'une application à une autre. Donc, la représentation, le stockage et le raisonnement sur ces informations est une tache difficile, les informaticiens se trouvent généralement obligés de reprendre tout le cycle de vie de l'application afin de fournir une nouvelle version qui supporte les nouveaux contextes d'utilisation.

Des travaux récents dans le domaine de la sensibilité au contexte se penchent vers l'utilisation de l'approche MDA, ceci grâce aux bénéfices qu'elle apporte à savoir la séparation des préoccupations, l'abstraction et les techniques de transformation. Parmi ces travaux :

- ✓ ContextUML de Sheng et Benatallah [Sheng05] est un langage UML pour la spécification des services web adaptées au contexte d'utilisation.
- ✓ Shumao Ou et al [Shumao06] ont appliqué l'approche MDA pour le développement d'applications sensibles au contexte en utilisant les ontologies.
- ✓ Ceri S et Al [Ceri07] ont appliqué l'approche MDA pour le développement d'applications web sensibles au contexte.
- ✓ Vale S et Hammoudi S [Vale08] ont présenté un méta modèle du contexte et ils ont développé une architecture orientée service sensible au contexte en utilisant l'approche MDA.

L'utilisation des modèles assure une meilleure compréhension du fonctionnement des services (chose manquante lorsqu'on manipule des fichiers XML et surtout lorsque la taille de ces fichiers est importante) et facilite considérablement la maintenance et l'évolution de nos applications (Il suffit de revenir au modèle pour introduire les modifications nécessaires au lieu de modifier un fichier XML généralement complexe et n'est pas lisible). En plus, des gains important au niveau temps de développement seront apportés grâce à l'utilisation des techniques de transformation de modèles.

La transformation est l'une des bases de l'approche MDA, cependant nous avons constaté que cette phase est généralement négligée dans les travaux cités ci-dessus. En plus, la plupart des outils de transformations qui existent sont spécifiques à des plateformes et ne permettent pas de prendre en compte les informations contextuelles.

Nous proposons dans ce mémoire de s'appuyer sur le méta modèle ContextUML [Sheng05], pour la prise en compte du contexte dans le développement des services, et nous essayerons de réaliser un transformateur permettant de prendre en compte les informations

contextuelles obtenues à partir de l'étape de la modélisation (avec le méta modèle ContextUML) et faire la transformation vers la plateforme web service.

IV. Cas d'étude

Pour mieux illustrer l'objectif de ce travail de thèse, nous utilisons tout le long de ce mémoire, un exemple d'application orientée service sensibles au contexte. Nous avons choisi cet exemple car il exprime un besoin d'actualité dans les systèmes d'information pervasifs. Nous nous appuyons sur une application de gestion du transport et des livraisons qui permet la consultation des prestations offerte par une entreprise de transport, le suivi des clients, des engins …etc. Cette application, offre un ensemble de services aux professionnels de l'entreprise qui assurent le suivi et la visualisation des dossiers des engins, des de dossiers des clients, des factures…etc. La diversité des utilisateurs de cette application (personnels, partenaires et clients) nécessite la prise en compte du contexte pour assurer la pertinence des résultats.

L'utilisation d'une approche dirigée par les modèles permet prendre en compte du contexte dès les premières étapes du cycle de vie, et permet aussi de gagner un temps considérable de conception et de réalisation grâce aux techniques de transformation. Ainsi, la maintenance et l'évolution seront ou plus simple, puisque il suffit de revenir les modèles qui ont un haut niveau d'abstraction (donc une meilleure compréhension du fonctionnement), et faire des transformations, ceci aide les développeurs à ne pas reprendre tout le cycle de développement des applications.

V. Organisation du mémoire

Après cette introduction générale, ce mémoire se compose de deux parties principales : un état de l'art, et une partie contributions contenant la proposition de notre travail et l'implémentation.

La partie état de l'art est composée de deux chapitres : « chapitre 1 : Web services et contexte» et « chapitre 2 : Model Driven Architecture (MDA)».

Dans le premier chapitre, nous allons présenter un état de l'art sur les web services, la notion de contexte, et de la sensibilité au contexte, nous ferons aussi un tour d'horizon sur les systèmes sensibles au contexte. Le deuxième chapitre est dédié à la présentation de l'approche MDA. La deuxième partie, qui regroupe l'ensemble de notre contribution, comporte deux chapitres : « chapitre 4 : Services web sensibles au contexte » et « chapitre 5 : Cas d'étude ». Le chapitre 4 présente notre problématique, nos objectifs, et le processus de développement que nous avons adopté pour développer des services sensibles au contexte. Nous présenterons aussi le méta modèle que nous avons adopté et nous expliquerons le principe du transformateur que nous avons développé. Le chapitre 5 présente notre cas d'étude et l'implémentation des services. Nous y présentons aussi les outils que nous avons utilisés.

Le document se termine par une conclusion générale qui présente une synthèse de notre contribution ainsi que les perspectives que nous avons tracées pour la suite de ce travail.

Partie 1

Etat de l’art

Chapitre 1

Web service et Contexte

I. Introduction

L'architecture orientée services (SOA) [Anand05] est un paradigme qui utilise des services pour soutenir le développement des applications distribuées d'une façon rapide, peu coûteux, interopérables et évolutif. Les services sont des entités autonomes faiblement couplées et indépendantes de la plateforme qui peuvent être décrites, publiées, et découvertes. Elles remplissent des fonctions qui vont répondre aux demandes des clients ou consommateurs. SOA est basé sur l'idée de composer des applications en découvrant et en invoquant des services disponibles sur le réseau pour accomplir certaines tâches. Cette approche est indépendante des langages de programmation spécifiques ou d'exploitation systèmes. Les services Web sont actuellement les plus prometteuses pour l'implémentation des SOAs, Ils utilisent l'Internet comme un moyen de communication et des normes basées sur Internet, y compris le Simple Object Access Protocol (SOAP) pour la transmission de données, le Web Services Description Language (WSDL) pour la définition des services, et les processus d'affaires Execution Language pour les services Web (BPEL4WS) pour l'orchestration des services.

II. Principes de SOA

SOA est un paradigme promettant dans le monde de l'industrie des logiciels, ceci grâce d'un coté à l'utilisation des standards (tels que XML, WSDL, BPEL, SOAP, HTTP…) pour l'échange d'informations et la spécification du comportement des services, et d'autre coté au principe de décomposition en services faiblement couplés. Ces derniers peuvent avoirs plusieurs consommateurs (Avec des dispositifs divers et des préférences particulières), et plusieurs contextes d'exécution.

L'entreprise est composée de différents départements (ventes, marketing, support, informatique...) remplissant chacun une ou plusieurs fonctions importantes. L'idée d'architecturer l'informatique d'entreprise au travers de ces différentes fonctions sous la forme de services n'est pas nouvelle. La SOA (Service Oriented Architecture) est une évolution de la notion d'informatique distribuée qui est apparue dans les années 90 avec des technologies comme Corba ou COM. Il s'agit de prolonger le concept de l'époque en y intégrant les apports d'Internet. La SOA est un concept général et peut s'appuyer en pratique sur de simples RPC (Remote Procedure Call), sur Corba ou plus classiquement sur des Services Web.

III. Utilisation de l'approche MDA (Model Driven Architecture) pour SOA

MDA (Model Driven Architecture, [Gervais02]) définit des méta modèles permettant de la transformation automatique des modèles, et cela en utilisant des standard tels que Le MOF (MetaObject Facility), standard OMG (Object Management Group), qui définit un ensemble de règles permettant de faire la correspondance entre les différentes structures utilisées par les méta-modèles. Utiliser MDA pour développer une application apporte plusieurs avantages :

- ✓ Obtention d'un modèle indépendant de la plateforme : le modèle indépendant de la plateforme (PIM - Plateforme Independant Model) représente la logique métier, les données et les exigences fonctionnelles de chaque application.
- ✓ Génération automatique du code à partir du modèle : à partir du modèle indépendant, des outils peuvent générer du code dépendant d'une architecture cible (plus ou moins raffiné suivant le niveau de raffinement du modèle).
- ✓ Simplification de la migration de l'application : la migration de l'application d'une plateforme à une autre permet de conserver la logique contenue dans le modèle ; tout n'est donc plus à réécrire.

IV. Services web

Différents livres donnent plusieurs définitions complémentaires du Web Services[Gottshalk02]. Certains d'entre eux sont donnés ici: "Un service Web est un logiciel qui se rend disponible sur Internet et utilise un format XML. Ce dernier est utilisé pour coder toutes les communications à un service Web. Parce que toutes les la communication sont en XML, les services Web ne sont pas liée à aucun système d'exploitation ou un langage de programmation. "Un service Web est un ensemble de protocoles ouverts et standards utilisés pour l'échange de données entre les applications ou systèmes. Les applications logicielles écrites en diverses langages de programmation et fonctionnant sur diverses plates-formes peuvent utiliser les services Web pour l'échange de données sur des réseaux informatiques comme l'Internet d'une manière similaire à la communication interprocess sur un seul ordinateur. Cette interopérabilité (par exemple, entre Java et Python, ou les applications Windows et Linux) est due à l'utilisation des logiciels libres des normes »[Wha09]. Ainsi, les services web sont indépendants de la plateforme, basée sur les messages XML. L'idée est de distribuer des services sur Internet et à les rendre disponibles pour les clients. Ces services peuvent être mis en œuvre avec tous les langues et peuvent être invoqués ainsi que composé.

La figure 1 présente les trois opérations principales d'une architecture service web : la description du service, la découverte et l'invocation.

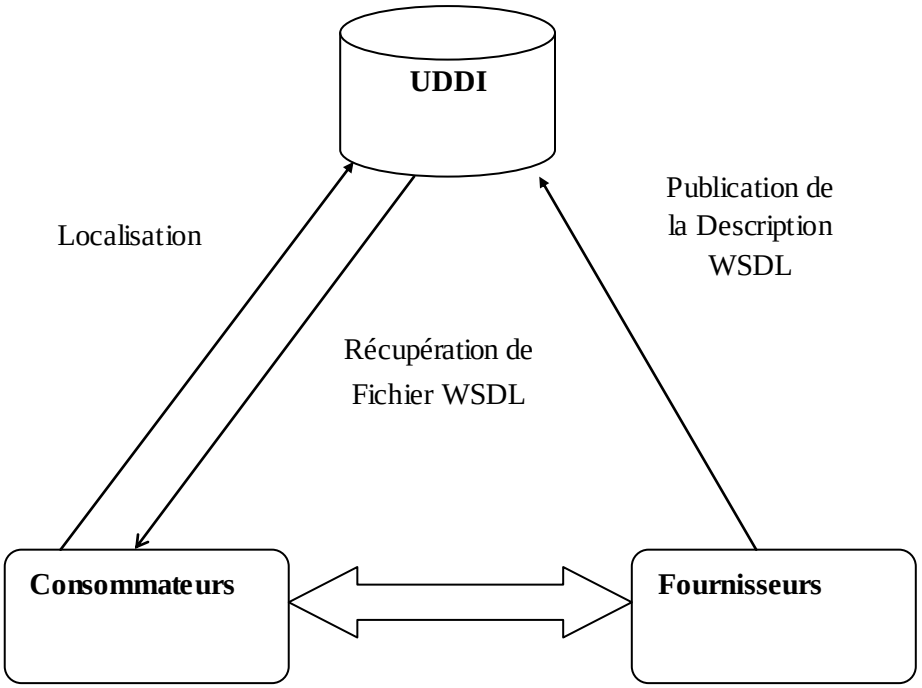

Figure 1 - Modèle fonctionnel de l'architecture de publication et d'invocation d'un service web.

Les services web se basent sur l'utilisation du protocole d'échange de données SOAP (lui même basé sur XML) [Kadima03]. Ce protocole se situe dans une couche supérieure des protocoles de niveaux applicatifs de l'Internet comme HTTP, FTP, SMTP, etc. Ces derniers encapsulent donc les messages XML issues du protocole SOAP dans leurs propres messages.

Le programmeur construit un service web en utilisant un langage de programmation spécifique. Ce service est publié en utilisant une interface WSDL. Ce service peut être invoqué par un des consommateurs "client" en utilisant cette interface. Les services Web sont présentés aux clients comme un ensemble d'opérations qui fournissent la logique métier. les services Web doivent être déployés sur un conteneur de serveur à la disposition des consommateurs. Sur côté le client, un objet distant qui représente le service à distance doit être généré. Cela permet aux clients d'invoquer les opérations définies sur le côté serveur. Le développeur n'a pas à se soucier de la création ou de l'analyse des messages SOAP. Cette tâche est effectuée par le système d'exécution du service Web. Toute langue peut également être utilisés pour réaliser de nouveaux services web. Ces services web sont invoquées partir de n'importe quel application qui est mis en œuvre en utilisant n'importe quel autre langage de programmation et fonctionne sur tout système d'exploitation.

V. Les langages et protocoles utilisés par les services web

V.1. HTTP (Hyper Text Transfer Protocol)

HTTP [HTTP97] est un protocole de connexion de données entre les explorateurs web et les serveurs. Il est le standard actuel de transfert des documents HTML. Il est conçu pour être extensible à d'autre format tel que XML.

Le but du protocole HTTP est de permettre un transfert de fichiers (essentiellement au format HTML) localisés grâce à une chaîne de caractères appelée URL entre un navigateur (le client) et un serveur Web

V.2. XML (eXtensible Markup Language)

Contrairement à HTML, qui est à considérer comme un langage défini et figé (avec un nombre de balises limité), XML [XML96] peut être considéré comme un métalangage permettant de définir d'autres langages, c'est-à-dire définir de nouvelles balises permettant de décrire la présentation d'un texte (Qui n'a jamais désiré une balise qui n'existait pas ?). La force de XML réside dans sa capacité à pouvoir décrire n'importe quel domaine de données grâce à son extensibilité. Il va permettre de structurer, poser le vocabulaire et la syntaxe des données qu'il va contenir.

Les éléments d'un document XML sont définit par une description spécifique à l'aide de DTD (Document Type Definition) ou avec XML Schéma.

Certaines applications utilisent XML pour le stockage d'informations grâce aux avantages suivantes :

- ✓ La nature textuelle des documents XML, leur rend indépendants des plateformes.
- ✓ L'encapsulation des données par des balises rend les documents XML lisibles par les humains.
- ✓ La validation des documents permet une meilleure communication.

V.3. Le langage WSDL

Le langage de description WSDL (Web Services Description Language, [Erik01]) est un langage de description de services Web basée sur XML. Son objectif principal est de séparer la description abstraite du service de son implémentation. Il permet de décrire les services web d'une façon unifiée. Chaque service est vu sous forme d'un ensemble de points d'entrée dans le réseau, capables d'échanger des messages

Le langage WSDL permet de décrire l'interface visible (ou publiée) nommé contrat du service web. Il décritles différents éléments du service (voir figure 2) , à l'aide du langage de balises XML :

- ✓ Les messages (intervenant lors des échanges) et leurs types associés (pour gérer l'interopérabilité entre les différents intervenants de l'échange)

- ✓ Les opérations (composées d'un ou plusieurs messages)
- ✓ Les liaisons et les ports de communication (permettant de lier les opérations à un protocole de transport sous-jacent)
- ✓ la description du service lui-même.

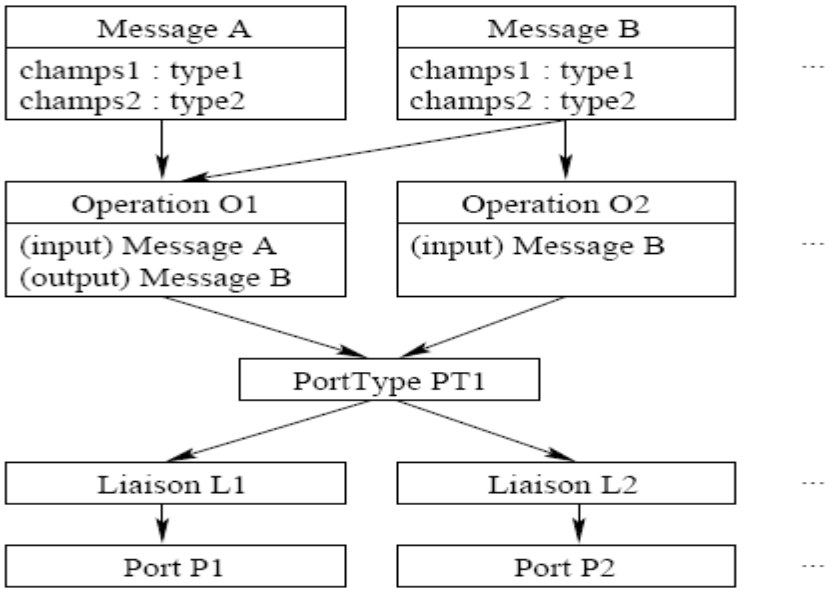

Figure 2 - L'architecture d'un fichier WSDL

Messages : est un ensemble de données qui sont décrits en XML et un type leur est associé. Les types de base XML peuvent être utilisés (entier, chaîne de caractères, etc.), mais également des types complexes définis dans un fichier WSDL (fichier identique à celui de la description du service ou externe).

Opérations : Une action prposée par un service web, décrite par ses messages (proche d'une méthode au sens java)

Plusieurs opérations sont associées pour former un PortType, utilisé par la suite.

Les liaisons : Un protocole et un format de données associé à un type de port. Ainsi, dans le cas d'un service web utilisant le protocole de communication SOAP :

- ✓ On définit pour chaque opération le type d'échange utilisé (mode document ou mode XMLRPC).
- ✓ On décrit pour chaque message (composant les opérations) l'espace de nom associé au type de message à transporter.

Service : Enfin, le service lui-même est une collection de ports (port ou endpoints)

Le mot clé port permet d'associer des adresses Internet (URL) à chaque PortType : ces adresses seront utilisées par le client pour l'invocation du service. Il est également possible

d'ajouter des informations permettant de trouver une documentation pour le service. C'est également cette partie qui peut être étendue par un autre langage que WSDL.

V.4. Protocole UDDI

UDDI (Universal Description, Discovery and Integration, [TOM02]) est une spécification définissant la manière de publier et de retrouver des informations concernant des services Web. Elle est issue d'une initiative de plus de 200 entreprises dont IBM, Microsoft et Ariba qui avaient pour but de définir une spécification commune pour l'enregistrement, l'identification et la découverte des services en ligne, conçus comme des API exploitables depuis un URL. Un annuaire UDDI contient des informations classées par catégories sur des entreprises, et les services qu'elles offrent, en associant `a ces services des spécifications techniques correspondantes. Ces spécifications techniques sont généralement définies en WSDL, ce qui permet d'avoir des informations sur ce que fait le service, comment il communique, et o`u il est hébergé.

UDDI fournit trois services de bases :

- ✓ **Publish :** ce service gère comment le fournisseur de service web s'enregistre lui-même ainsi que ses services (en utilisant UDDI).
- ✓ **Find :** ce service gère comment un client peut localiser le service web désiré (cela peut passer par des invocations de services web pour une utilisation automatique par un programme ou par une consultation d'annuaire en utilisant des mots clés).
- ✓ **Bind :** ce service gère également comment un client peut se connecter et utiliser le service web une fois celui-ci localisé.

Les entreprises qui fournissent ce service et hébergent un annuaire global UDDI sont appelées des opérateurs UDDI. Ils sont responsables de la synchronisation de l'information des annuaires : cette synchronisation d'annuaires s'appelle la réplication.

Il existe de nombreux annuaires UDDI dont les principaux sont :

- ✓ Celui de Microsoft : http://uddi.microsoft.com ;
- ✓ Celui d'IBM : http://uddi.ibm.com.

Un des enjeux de UDDI est d'éviter le monopole d'une entreprise qui, par un annuaire quelconque, donnerait comme réponse systématiquement ses services web plutôt que celle des autres : ce genre de pratique étant fortement limité car les annuaires UDDI se doivent d'avoir un contenu identique aux autres annuaires. Ainsi, il permet de donner des solutions industriellement viables pour la localisation de services web.

V.5. Le protocole SOAP

Les communications entre les différentes entités impliquées dans le dialogue avec le service web se font par l'intermédiaire du protocole SOAP (Simple Object Access Protocol).

Ce protocole est normalisé par le W3C [Mitra02, Gudgin02a, Gudgin02b]. La description qui va suivre est basée sur la version 1.2 de ce protocole.

Le protocole SOAP est un protocole léger destiné à l'échange d'informations structurées dans un environnement distribué. Il utilise des technologies XML pour définir une structure de messages pouvant être échangés sur divers protocoles sous-jacents(p.ex. HTTP, SMTP). Cette structure a été conçue pour être indépendante de tout modèle de programmation et autre sémantique spécifique d'implémentation. Un message SOAP est une transmission unidirectionnelle entre des nœuds SOAP (d'un ´émetteur vers un récepteur SOAP), mais les messages SOAP peuvent être combinés par les applications pour implémenter les séquences plus complexes d'interactions (p.ex. requête-réponse, échanges multiples conversationnels).

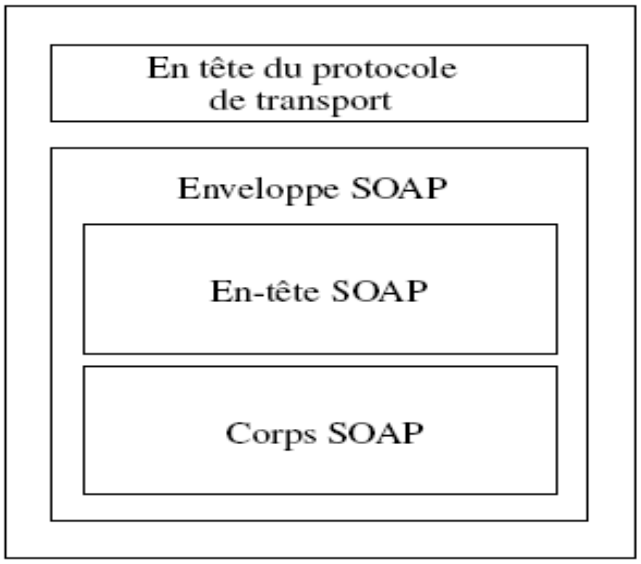

Figure 3 - Schéma d'un message SOAP

Un message SOAP contient deux sous-éléments `a l'intérieur de l'élément Envelope externe: un élément Header (en-tête) et un élément Body (corps). Les contenus de ces éléments sont définis par l'application et ne font pas partie de la spécification de SOAP.

L'en-tête du protocole de transport Cette partie dépend du protocole de transport utilisé. Par exemple, si le protocole HTTP est utilisé, cette en-tête contient :

- ✓ la version de HTTP utilisée,
- ✓ la date de génération de la page (qui est ici le message SOAP lui-même),
- ✓ le type d'encodage du contenu (ici, l'encodage est généralement le type text/xml), etc.

Les messages SOAP (l'enveloppe) La partie principale d'un message SOAP est l'enveloppe (symbolisée par la balise envelope). Cette enveloppe (SOAP Envelope) est elle même subdivisée en deux sous-parties : la partie en-tête et la partie corps du message. Elle permet également de spécifier des environnements de noms XML utilisés dans la suite du message.

L'en-tête du message SOAP L'en-tête SOAP (SOAP Header) est optionnelle et extensible.

Les balises XML permettant de symboliser cette partie sont <env:Header> et </env:Header>.

Ces balises peuvent être complétées par des attributs permettant de définir le domaine de noms du service web.

En fait, l'en-tête permet principalement d'ajouter des informations sur le comportement que doivent avoir les différents noeuds intermédiaires, lors du traitement du message.

Un noeud étant un intermédiaire SOAP, incluant le récepteur et l'émetteur SOAP, désignable depuis un message SOAP. Son rôle est de traiter l'en-tête (et d'effectuer les actions qui y sont décrites) et ensuite de transférer le résultat (le message SOAP modifié) à un autre intermédiaire (qui peut être le récepteur final).

Par extension, la description du comportement des différents noeuds permet également de réaliser une "composition de services", car le message peut être routé entre différents noeuds, chacun étant capable de réaliser une action précise et décrite dans le bloc d'en-tête.

Les principaux attributs des éléments formant le bloc sont :

- ✓ Env:role : permet d'indiquer à quel noeud la fonction décrite est destinée. Et donc par extension, cela permet le routage d'un message.
- ✓ Env:mustUnderstand : c'est une valeur booléenne, elle permet de préciser que le traitement devient obligatoire pour un noeud intermédiaire. Par exemple, pour un calcul très long, il peut être utile d'envoyer un e-mail à chaque étape.
- ✓ Env:relay : cet attribut permet de relayer un message à un autre noeud si le premier nœud n'est pas capable de le traiter.

Le corps du message SOAP Les données spécifiques à l'application se trouvent dans le corps du message SOAP (SOAP Body). Tout ce qui est présent dans cette section est défini lors de la conception de l'application. Enfin, les balises symbolisant cette partie sont <env:Body> et </env:Body>. Les données doivent donc être sérialisées selon l'encodage choisi. L'utilisation du XML 1.0 à l'intérieur des blocs XML <element> permet d'envoyer absolument tous les types de documents comme par exemple des feuilles de styles, des documents XML, des images au format binaire, etc...

En plus des données, cette partie peut transporter un type spécial : les messages d'erreurs (SOAP Fault).

Comme précédemment on peut ajouter un attribut dans la balise <env:Fault> permettant d'indiquer un type d'encodage des données, mais également d'autres attributs permettant la gestion de l'erreur, comme Code, Reason, Node, Role et Detail.

VI. Composition de services

VI.1. Définitions

1. Service web complexe

Un service complexe est un service web dont les opérations eux-mêmes sont des services web.

2. Orchestration

L'orchestration de services Web est la spécification d'un processus métier exécutable qui peut interagir avec l'ensemble des services qu'il orchestre (considérés comme des services internes), ainsi qu'avec d'autres services (considérés comme des services externes). La description de ce processus est elle même privée, et ne peut donc pas être accédée de l'extérieur. L'orchestration diffère de la chorégraphie car elle décrit un processus contrôlé par une unique entité centrale, alors que la chorégraphie décrit un modèle plus collaboratif (pair-à-pair), engendrant une échange de messages entre plusieurs entités, sans qu'aucune de ces entités ne contrôle cet échange.

3. Chorégraphie

La chorégraphie ne repose pas sur un service web principal. Chacun des services intervenant dans la composition sait exactement ce qu'il doit faire, quand il doit le faire et avec qui. Donc ils ont tous une connaissance plus ou moins globale du processus métier dans lequel ils se retrouvent. Contrairement à la méthode basée sur l'orchestration, la chorégraphie demande nettement plus de développement et de test : il faut développer chaque service dans le but de la composition qu'ils formeront.

Cela dit, en utilisant des méthodes de développement et des stratégies bien pensées, l'intérêt de la méthode apparaît : la non-centralisation du traitement.

L'approche la plus simple dans le passage à SOA est l'orchestration. En effet, le but principal de la composition est la réutilisation de services sans modifier ceux-ci (car ils peuvent très bien être hébergés par une autre compagnie et nous n'avons alors aucun moyen de contrôle sur ceux-ci). Le processus principal doit alors pouvoir s'adapter aux différentes erreurs possibles (non disponibilité d'un service, annulation, etc.).

SOA permet également d'utiliser l'approche chorégraphie : la preuve en est de l'existence de nombreux standards comme WSCI (Web Services Choreography Interface [Assaf02b]), ou encore WS-CDL (Web Services Choreography Description Language [Kavantzas05]).

VI.2. Langages de composition

Pour palier le manque d'information lié à l'utilisation du langage de description WSDL, on utilise les langages de programmation ou de composition de services web. un service web utilisant ces langages sont alors appelés services web complexes.

Parmi les langeges de compositions, on peut citer :

– ***WSFL :*** le langage Web Services Flow Language développé par IBM [Leymann01] dans le but de rendre possible la description de la composition d'un ensemble de services web en se basant sur la composition des flots de manière hiérarchique.

– *BPML :* le langage Business Process Modeling Language [Assaf02a] développé par l'organisation BPMI.org (Business Process Management Initiative), regroupant des acteurs comme Abode, Corel, BEA, IBM, Sun, etc... Il présente de nombreux points communs avec

les langages que nous allons développer par la suite, comme la notion de processus et d'activité.

– **XLANG :** le langage XLANG [Thatte01] a été développé par Microsoft en 2001, pour les besoins de sa plateforme de gestion de processus BizTalk. Ce langage permet de représenter les éléments clés, d'un point de vue algorithmique, des processus métier et se classe ainsi dans les langages de description comportementale.

– ***BPEL4WS*** : le langage BPEL4WS [Andrews03] est développé en 2003, par un regroupement d'industriels, parmi ceux-ci, on retrouve IBM, BEA et Microsoft. Ce langage est une fusion des langages, dit de première génération : XLANG et WSFL. Il reprend donc les avantages de ces deux langages en tirant parti de l'apprentissage lié à leur mise en place. Il se focalise sur la représentation du processus métier en se rapprochant des structures algorithmiques.

VII. Contexte et sensibilité au contexte

Avec l'apparence et la prolifération des dispositifs mobiles tels que les noetbooks, PDAs, SmartPhones, les systèmes ubiquitaires sont devenu rapidement populaire. Les utilisateurs de tels systèmes peuvent se retrouver partout (à la maison, dans le bus, à l'université, à l'hôpital, Aéroport …) et leurs accèdent à tout moment (le matin, le soir, en hivers, en été…). La sensibilité au contexte représente l'un des champs de l'informatique ubiquitaire.

Les systèmes sensibles au contexte ont la possibilité d'adapter leurs opérations à la situation contextuelle courante, sans l'intervention explicite de l'utilisateur. Ceci, améliore considérablement l'interaction utilisateur et augmente la pertinence des résultats fournies

VII.1. Définition du contexte

Il est très difficile de définir le mot contexte, et beaucoup de chercheurs ont essayé de formaliser sa signification. Cependant, jusqu'à présent ils n'ont pas arrivée à une définition standard et universelle.

Dans la littérature, les chercheurs ont commencé par fixer quelques informations contextuelles. Par exemple Shilit et Theimer [Schilit94] ont défini le contexte comme étant : la localisation, l'identité, le voisinage, les objets et les changements sur ces objets. Après Shilit et al [Schilit94] ont définit 3 classes d'informations contextuelles qui sont :

- ✓ Informations techniques (liées à l'informatique): telles que la connectivité du réseau, coûts de communication, ressources.
- ✓ Paramètres liés à l'utilisateur : tels que son profile, sa localisation, sa situation sociale.
- ✓ Propriétés physiques (sur l'environnement): telles que : la luminosité, bruit, et la température.

Par la suite Chen et Kotz [Chen00] ont introduit la classe temps (Time) représentant les paramètres tels que le temps du jour, semaine, mois, et la saison de l'année.

Dey et al [Dey98] définit le contexte comme étant : l'état émotionnel de l'utilisateur, son points d'attention, sa localisation et son orientation, le temps et la date, les objets et personnes qui se trouvent dans l'environnement.

Toutes ces initiatives étaient limitées et incomplètes, car elles ne permettent de représenter que des informations liées à des domaines spécifiques et la liste de ces informations n'est pas exhaustive. Pour cette raison, d'autres définitions ont vus le jour. Brown [Brown96] a définit le contexte comme étant : les informations sur les éléments de l'environnement de l'utilisateur. Ward et al [Ward97] voyaient le contexte comme étant l'état des paramètres dans lesquels l'application s'exécute.

Plu tard, Dey et al [Dey99] ont dit que le contexte se diffère d'une situation à une autre et dépend de l'application à développer. Pour cela ils ont proposé une définition formelle :

" ... any information that can be used to characterise the situation of an entity. An entity is a person, place or object that is considered relevant for the interaction between a user and an application, including the user and applications themselves."

Nous traduisons cette définition comme suit : '' Le contexte couvre toutes les informations pouvant être utilisées pour caractériser la situation d'une entité. Une entité est une personne, un lieu, ou un objet qui peut être pertinent pour l'interaction entre l'utilisateur et l'application y compris l'utilisateur et l'application eux-mêmes''.

Dans cette thèse, nous adoptant cette définition et nous considérons le contexte comme étant la collection de toutes les entités et leurs propriétés qui peuvent influencer sur l'interaction entre l'utilisateur et l'application (y compris l'utilisateur et l'application eux-mêmes).

VII.2. Sensibilité au contexte

Cette notion est une traduction de l'expression anglaise "context-awareness". Elle caractérise la capacité d'un système à s'adapter aux changements du contexte. Selon Dey et Abowd, un système est sensible au contexte s'il utilise le contexte pour fournir des informations et des services pertinents pour l'utilisateur, où la pertinence dépend de la tâche demandée par l'utilisateur [Dey00a].

Cette définition d'un système sensible au contexte a été adoptée par tous les chercheurs dans ce domaine.

Schilit et al [Schilit94] catégorise la sensibilité au contexte comme suit :

1. Faciliter le choix des objets qui entourent l'utilisateur (Proximate Selection).
2. Ajout de nouveaux ou suppression des composants existants en se basant sur le contexte (Automatic Contextual Reconfiguration).

3. Selon le contexte produire des résultats différents (Contextual Information and Commands).
4. Des regles si-sinon simple utilisés pour spécifier comment une application doit s'adapter (Context-triggered Actions).

Selon cette catégorisation, les applications sensibles au contexte peuvent proposer des choix d'actions appropriées à l'utilisateur. Il y a plusieurs exemples de ce type de travaux dans la littérature et dans quelques systèmes commercialisés. Par exemple, nous pouvons trouver des applications pour montrer la localisation de l'utilisateur ou de son véhicule sur une carte. Elles peuvent aussi proposer des icones (ou des alertes) des centres d'intérêts voisins de l'utilisateur [Abowd97], [Bederson95], [Davies98], [Feiner97], [Fels98], [McCarthy99], [McCarthy00]).

Nous pouvons aussi citer les travaux de [Schilit94] qui présentent la liste des imprimantes proches de l'utilisateur. Nous pouvons aussi trouver d'autres études sur la présentation d'informations dans des systèmes ambiants [Heiner99], [Ishii97], [Mynatt98], [Weiser97]….

Une autre catégorie, est l'exécution automatique de services, elle décrit les applications qui déclenchent une commande, ou reconfigurent le système à la place de l'utilisateur selon les changements de contexte. Dans cette catégorie nous pouvons citer : le système Teleport qui assure le transport automatique de profil utilisateur lorsqu'il passe d'une machine à une autre [Bennett94], un système d'enregistrement automatique de son quand une réunion ou un rassemblement non planifié se passe dans un certain lieu [Brotherton99], des téléphones mobiles qui changent leurs comportement et leurs configurations (vibreur/sonnerie) selon l'environnement de l'utilisateur [Harrison98], un système de sécurité portable qui détecte si l'utilisateur est effrayé en utilisant des capteurs biométriques [Healey98], et des dispositifs qui fournissent des signaux de rappel quand les utilisateurs sont à un lieu précis ([Beigl00] et [Marmasse00]).

Dans une autre categorie, les applications associent des données au contexte de leur utilisation. Par exemple, dans une conférence, une application étiquette des notes prises par l'utilisateur avec le lieu et le temps de l'observation [Pascoe98]. Dans des domaines similaires, Time-Machine Computing [Rekimoto99] et Placeless Documents [Dourish00] sont deux systèmes qui attachent l'identité des utilisateurs, leurs lieux et le temps de création et d'utilisation de ressources logicielles afin de pouvoir les utiliser d'une façon plus facile et rapide ultérieurement. D'autres exemples plus complexes dans cette catégorie sont des applications d'aide-mémoire telles que Forget-Me- Not [Lamming94] et Rememberance Agent [Rhodes97].

Malgré tous ces travaux, le domaine de la sensibilité de contexte est loin d'être au point. En effet, plusieurs éléments sont encore à approfondir et à étudier, par exemple : jusqu'à présent la notion de contexte n'est pas encore bien définie

VII.3. Modélisation du contexte

Pour pouvoir utiliser les informations contextuelles, les stocker et les partager, il faut tout d'abord les modéliser. La modélisation du contexte était le sujet de plusieurs travaux récents dont le but est de développer des applications sensibles au contexte, le Context Toolkit est un exemple de ces travaux [Dey01]. Pour les applications mobiles les travaux se basent sur la modélisation de la localisation. Cependant, cette dernière est juste l'une des plusieurs types d'informations contextuelles.

Certains travaux utilisent des méthodes standards pour modéliser le contexte, telles que les couples Attribut/Valeur ou les ontologies. Ces approches se diffèrent par le niveau d'abstraction, leurs formalismes, la richesse sémantique, et la facilité d'implémentation.

Strang et al [Strang04] ont étendu le travail de Chen et Kotz [Chen00] qui classifiait les applications selon les structures de données utilisées pour modéliser le contexte. Les structures de données les plus utilisées sont :

1. *Couples Attribut/Valeur* stockent le type du contexte par un attribut et la valeur actuelle par une valeur. Le context Toolkit [Dey01] utilise cette approche, Schilit et al [SAW 94] modélise la localisation de cette manière.
2. *Les approches basées sur la logique* utilisent un système formel pour décrire le contexte. Chen Harry et al [Chen03] utilisait CORBA-ONT pour décrire les places, les agents, les évènements, les propriétés associées dans le domaine de meeting-room en utilisant la logique des prédicats.
3. *Les Méthodes orientées objets* encapsulent les informations contextuelles comme étant des états des objet, l'accès à ces informations se fait à l'aide de méthodes spécifiques. Les projets TEA [Schmidt99] et GUIDE [Cheverst99] sont des exemples de telles applications.
4. *Les Schémas à balisage* sont basé sur l'utilisation du concept de structure hiérarchique ou chaque information contextuelle est annotée par une description du rôle joué par sa valeur. Stick-e note de Pascoe [Pascoe97] et ConteXtml de Ryan [Ryan99] sont des exemples de telles applications.
5. *Modèles graphiques* utilisation des langages graphiques tels que UML. ContextUML [Sheng05] est un exemple de telles applications.
6. *Utilisation des ontologies* pour modéliser le contexte. CoOL [Strang03] est un exemple de telles applications

VII.4. Qualité du contexte

L'une des caractéristiques du contexte est son imperfection. Cela se présente sous formes d'incomplétude des informations contextuelles, d'inconsistance ou d'erreur. La qualité du contexte dépend de la source d'information. Donc une application sensible au contexte doit exiger certaine qualité des informations contextuelles.

Dey [Dey00b] propose quelques critères de qualité :

- ✓ L'exactitude (Accuracy).
- ✓ La tolérance du système vis-à-vis des fautes du senseur (Reliabilite).
- ✓ L'ensemble des valeurs possibles d'une information contextuelle (Coverage).
- ✓ Le changement exigé d'une information contextuelle (Resolution).
- ✓ Le temps nécessaire pour qu'une information contextuelle soit mise à jour (Frequency).
- ✓ Le temps entre la notification d'une information contextuelle et le changement de sa valeur (Timeliness).

Ebling et al [EHL01] définit deux critères de qualité :

- ✓ La date de la dernière mise à jour (Freshness).
- ✓ Le taux d'exactitude (Confidence).

VII.5. Systèmes existants et plateformes

Dans ce paragraphe nous présentons quelques exemples d'applications sensibles au contexte.

VII.5.1. Systèmes sensibles au contexte

A. Applications sensibles à la localisation

Elles représentent un cas particulier des applications sensibles au contexte. Comme exemple de ces applications, on peut citer les projets de guidage de touristes où la localisation joue un rôle très important. D'autres exemples peuvent être trouvés dans Espinoza et al. [Espinoza01], Priyantha et al. [Priyantha00], Burrell and Gay [Burrell02] and Kerer et al [Kerer04].

Ces applications utilisent les satellites GPS, les tours des Téléphones portables, les détecteurs des badges, les caméras, les lecteurs des cartes magnétiques, les lecteurs des code-barres…

B. Applications sensibles au contexte

Les applications présentées dans la section précédentes utilisent un seul aspect du contexte qui est la localisation. L'utilisation d'autres aspects (tels que le profile utilisateur, le temps, les personnes voisines …) permet d'avoir plus de connaissances sur les conditions d'exécution d'une application. Par conséquent nous pouvant construire des applications pouvant s'adapter au contexte, et mieux satisfaire les utilisateurs. Un exemple de ces applications est présent dans Munõz et al [Munõz03].

VII.5.2. Plateformes

Elles permettent de développer des applications sensibles au contexte. Des exemples de plateformes sont présentés dans les projets The Portolano [Elser99], Oxygen [Dertouzos], Aura [Garlan02], and Gaia [Roman02].

VII.6. Travaux connexes

Dans la littérature, peu de travaux s'intéressent à la prise en compte du contexte dans le développement des services. Keild et indulska [Keild04] ont proposé un Framework pour le développement et le déploiement des services web sensibles au contexte. Dans ce framework les informations contextuelles sont limitées aux informations reliées au consommateur du service, et elles sont embarquées dans des messages SOAP.

Ces dernières années, certains travaux proposent l'utilisation de l'approche MDA pour le développement des services web. Skogan et al [Skogan04] proposent une approche MDA basée sur UML pour la composition des services. Baina et al [Baina04] proposent un framework qui supporte l'approche MDA pour le développement des services web et présentent comment générer une spécification exécutable complète d'un service web à partir de sa spécification externe. Cependant, ces travaux ne prennent pas en compte le contexte dans le développement des services.

Ceri et al [Ceri03] proposent des solutions pour la modélisation multicanaux des applications web sensibles au contexte. Leur travail est basé sur un langage de modélisation nommé WebML qui était initialement conçu pour le développement des pages web, en plus leur modèle du contexte ne couvre pas l'hétérogénéité des informations contextuelles [Sheng05].

Shumao Ou et al [Shumao06] ont appliqué l'approche MDA pour le développement d'applications sensibles au contexte en utilisant les ontologies. Dans [Ceri07] les auteurs ont appliqué l'approche MDA pour le développement d'applications web sensibles au contexte. Vale S et Hammoudi S [Vale08] ont présenté un méta modèle du contexte et ils ont développé une architecture orientée service sensible au contexte en utilisant l'approche MDA.

Ces travaux se basent sur des significations et représentations différentes des informations contextuelles, (jusqu'à présent il n'existe pas un méta modèle standard pour le contexte).

Sheng et Benatallah [Sheng05] proposent ContextUML qui est un méta modèle permettant la spécification des services web adaptées au contexte d'utilisation. Ce travail permet de séparer la modélisation du contexte et la sensibilité au contexte des composants du service. Il offre deux mécanismes d'adaptation au contexte. Cependant, comme la plupart des autres travaux la phase de transformation est négligée. Le tableau ci-dessous présente une comparaison entre tous ces travaux :

Travail / Critère	[Ceri03]	[Ceri07]	[Shumao06]	[Vale08]	[Sheng05]
Modélisation	Oui	Oui	Oui	Oui	Oui
Transformation	Non	Non	Non	Non	Non
Informations contextuelles	limitées	limitées	limitées	limitées	Tout type d'information
Langage	WebML	WebML	Ontologie	Edoc	UML

Tableau 1 Tableau comparatif des travaux connexes

Selon le tableau 1, la plupart des travaux se basent sur des significations et représentations différentes des informations contextuelles, et négligent l'étape de la transformation,

Nous nous basons sur la modélisation du contexte dans le langage UML, et nous s'appuyons sur le méta modèle ContextUML proposé dans [Shen05] pour la modélisation des services web. De plus nous proposons une transformation automatique des modèles obtenus à partir de l'étape de la modélisation vers la plateforme service web. Le problème est que les outils existant sont spécifiques à des plateformes et ne permettent pas de prendre en compte les informations contextuelles, pour cela nous allons essayer de développer un transformateur capable de prendre en compte le contexte.

VIII. Conclusion

SOA n'est pas la panacée du développement du SI, c'est une approche d'architecture du système d'information avec un ensemble de bonnes pratiques que chaque entreprise doit adapter à ses besoins.

Toutefois, dans un contexte actuel de concurrence toujours plus forte, les besoins stratégiques évoluent rapidement.

Pouvoir répercuter ces changements dans un SI classique nécessite des investissements lourds et coûteux.

Les applications orientés services sont une solution efficace permettant de rendre le système d'information de l'entreprise réactif au changement, indépendant vis-à-vis des éditeurs ou prestataires, tout en maîtrisant les coûts.

La majorité des solutions existantes dans le domaine de la sensibilité au contexte intègrent des règles si-sinon pour décrire comment les systèmes sensibles au contexte doivent réagir aux changements contextuels. Ces règles sont statiques et ne couvrent pas la totalité des informations contextuelles, par exemple si de nouvelles informations contextuelles apparaissent il faut ajouter d'autres règles pour les prendre en charge. Il est à noter aussi que

les informations contextuelles se diffèrent d'une application à une autre. Donc, la représentation, le stockage et le raisonnement sur ces informations est une tache difficile, les informaticiens se trouvent généralement obligés de reprendre tout le cycle de vie de l'application afin de fournir une nouvelle version qui supporte les nouveaux contextes d'utilisation.

Grâce aux bénéfices de l'ingénierie des modèles, des travaux récents proposent d'utiliser l'approche MDA, Cependant, ces travaux se basent sur des significations et représentations différentes des informations contextuelles, (jusqu'à présent il n'existe pas un méta modèle standard pour le contexte) et peu d'entre eux ont bénéficié des apports de cette approche.

Dans la deuxième partie nous allons présenter notre contribution qui consiste à proposer à utiliser l'approche MDA pour la prise en compte du contexte dans des architectures SOA, et nous proposerons un méta modèle permettant de modéliser tout type d'information contextuelle. En suite, Nous utiliserons java et XSLT pour des transformations vers WSDL et BPEL.

Chapitre 2

MDA (Model Driven Architecture)

I. Introduction

Le développement des logiciels est devenu l'une des plus larges industries dans la planète, et beaucoup de grandes compagnies actuelles sont des organisations qui produisent du software et les services en relation [MDSD04].

L'approche MDA est un nouveau paradigme introduit par l'OMG, il a pour objectif d'industrialiser le développement software tout en permettant l'interopérabilité entre les différents systèmes, en réduisant le coût de développement et en augmentent l'évolutivité.

Dans cette approche la notion de modèle devient essentielle et centrale, l'OMG a proposé en 2001 le concept de Platform Independent Model (PIM), et le concept de Platforme Specifc Model (PSM) pour le choix de la plate forme telles que SOAP, XML, JAVA, EJB, CORBA, .NET, Web Services…

Unified Modeling Language (UML) est un standard de l'OMG, il permet de définir plusieurs types de modèles qui peuvent être des PIMs ou des PSM selon notre choix.

Le processus de développement est devenu : PIM->PSM->Code, l'essentiel de la démarche MDA est la capacité de passer d'un modèle à un autre. La transformation de modèles et au cœur de l'MDA, pour cela OMG propose plusierurs standards tels que QVT, MOF, OCL, CWM , XMI.

MDA est une approche qui offre des outils permettant de [OMG03a] :

- ✓ Spécifier un système indépendamment d'une plate forme spécifique.
- ✓ Spécifier plusieurs plateformes.
- ✓ Choisir une plateforme spécifique.
- ✓ Transformer la spécification du système en une plateforme particulière.

II. Le cycle de vie du développement MDA

L'approche MDA a changé le rôle de la modélisation dans le développement software. Les modèles jouent un rôle très important dans le cycle de vie du développement sous MDA. Les modèles suivants sont le cœur de l'approche MDA [Kleppe03].

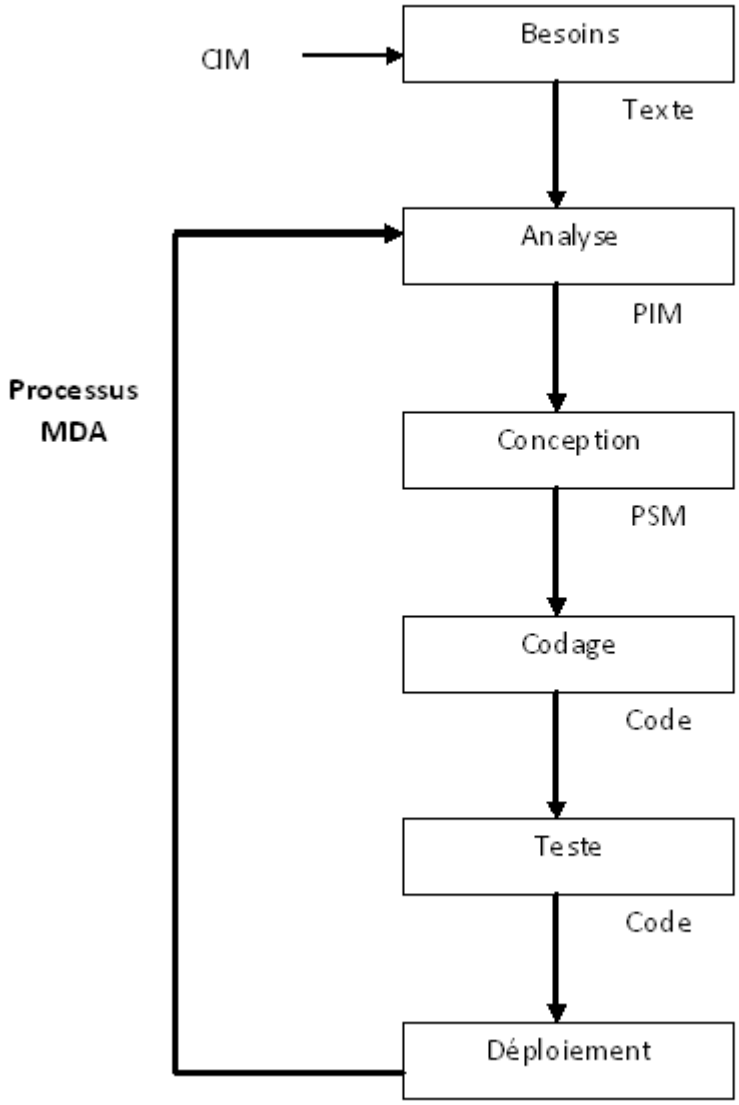

Figure 4 - Cycle de vie du développement MDA [Kleppe03].

1. Computation Independent Model (CIM)

Appelé aussi modèle de domaine, il est indépendant de tout système informatique. Le CIM aide à représenter ce que le système devra exactement faire. Il est utile, non seulement comme aide pour comprendre un problème, mais également comme source de vocabulaire partagé avec d'autres modèles. L'indépendance technique de ce modèle lui permet de garder tout son intérêt au cours du temps et il est modifié uniquement si les connaissances ou les besoins métier changent.

2. Platform Independent Model (PIM)

Il est indépendant de toute plate-forme technique (EJB, CORBA, .NET,…) et ne contient pas d'informations sur les technologies qui seront utilisées pour déployer

l'application. C'est un modèle informatique qui représente une vue partielle d'un CIM. Le PIM représente la logique métier spécifique au système ou le modèle de conception. Il représente le fonctionnement des entités et des services. Il doit être pérenne et durer au cours du temps. Il décrit le système, mais ne montre pas les détails de son utilisation sur la plate-forme. A ce niveau, le formalisme utilisé pour exprimer un PIM est un diagramme de classes en UML qui peut-être couplé avec un langage de contrainte comme OCL (Object Constraint Language).

3. Platform Specifique Model (PSM)

A l'étape suivante, le PIM sera transformé en PSM, ce dernier est un modèle spécifique à une plateforme ou à une technologie particulière. Il sert essentiellement de base à la génération de code exécutable vers la ou les plateformes techniques.

Un PIM peut être transformé en un ou plusieurs PSM. Le PSM utilise une technologie spécifique pour décrire le système, cette technologie peut être EJB, J2EE, Un model de base de données relationnelle…La transformation peut être effectuée par des outils.

4. Code

A la fin, le PSM sera transformé en code, il est possible que la génération de code soit directement à partir du PIM sans passer du PSM.

III. Automatisation des étapes de transformation

L'approche MDA cherche à automatiser le passage d'un modèle à un autres durant le cycle de développement. Les différentes transformations vont être effectuées d'une façon automatique en utilisant des outils de transformations, la figure 5 montre les différentes transformations. Dans le développement traditionnel, la transformation du PSM vers le code est réalisée. Ce qui est nouveau dans l'approche MDA est la transformation du PIM vers PSM, ce passage est la partie la plus difficile de l'approche MDA. Aujourd'hui plusieurs outils de transformation existent tels que UMT, MTL, ATL, GMT, BOTL, OptimalJ.

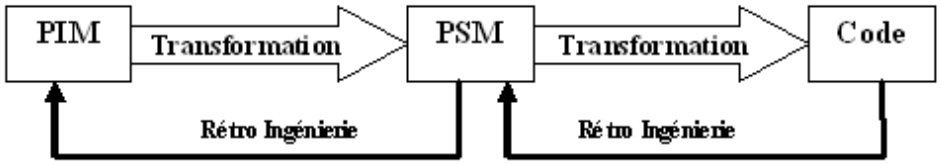

Figure 5 - Transformations de l'approche MDA

Des transformations Code->PSM ou PSM->PIM peuvent être effectuées par la rétro ingénierie.

IV. Application de l'approche MDA

L'application de l'approche MDA dans le développement software dépend de deux conditions principales :

- ✓ Le produit cible est influencé par les changements rapides des nouvelles technologies.
- ✓ Le métier est stable et dure longtemps.

Pour créer une application, un architecte fait un schéma en UML par exemple, mais d'autres langages peuvent aussi être utilisés. En partant du centre de la figure 6 ci-dessus, l'architecte dirigera son application en évoluant de couche en couche pour aller vers le domaine d'application qui l'intéresse (Finance, Télécommunication, Transport, Espace, médecine, commerce électronique, manufacture,…).

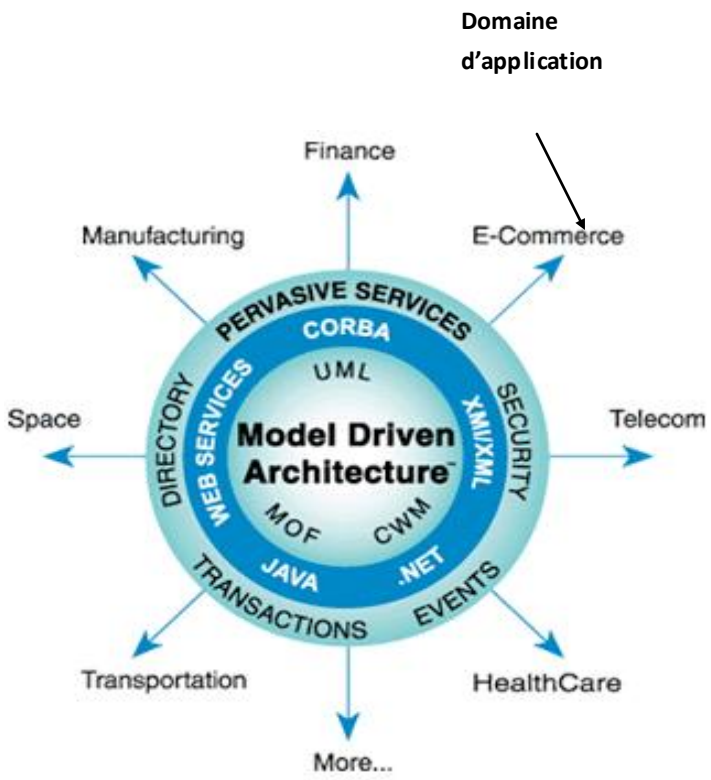

Figure 6 - Application de l' MDA [OMG03a]

Les avantages d'utiliser l'approche MDA sont nombreux :

- ✓ L'indépendance de la logique métier vis à vis de la plate-forme technologique.
- ✓ Grâce à la transformation automatique du modèle vers le code, les programmes sont développés plus rapidement
- ✓ Il y a une réelle adéquation de l'application et des besoins de l'utilisateur.
- ✓ Flexibilité à des nouvelles technologies, le PIM peut être porté vers différentes plateformes et différentes technologies
- ✓ Meilleure maintenabilité, les changements métiers au niveau du PIM vont être transformés en PSM et Code d'une façon très facile.

Bien que la démarche comporte des avantages indéniables, elle peut en rebuter certains :

- ✓ En effet, pour développer avec la démarche MDA, cela nécessite des architectes expérimentés ayant une grande connaissance dans la modélisation et le codage.
- ✓ Un potentiel risque de l'immaturité des standards de l'OMG et des outils supportant ces standards.

V. Méta modèles et architecture à 4 couches

Un modèle est une représentation structurelle ou comportementale d'une application. Une représentation est basée sur un langage ayant une syntaxe et une sémantique, avec probablement des règles permettant d'assurer certaines contraintes. Un méta modèle sert créer un langage pour modéliser. La méta modélisation est le processus de conception des langages à partir des méta et méta notations. A partir des méta modèles on peut s'assurer que les modèles obtenus sont syntaxiquement correctes.

Dans l'approche MDA il y a plusieurs niveaux d'abstraction, pour chaque niveaux il existe un formalisme de représentation, la figure 7 montre ces différents niveaux (architecture à 4 couches).

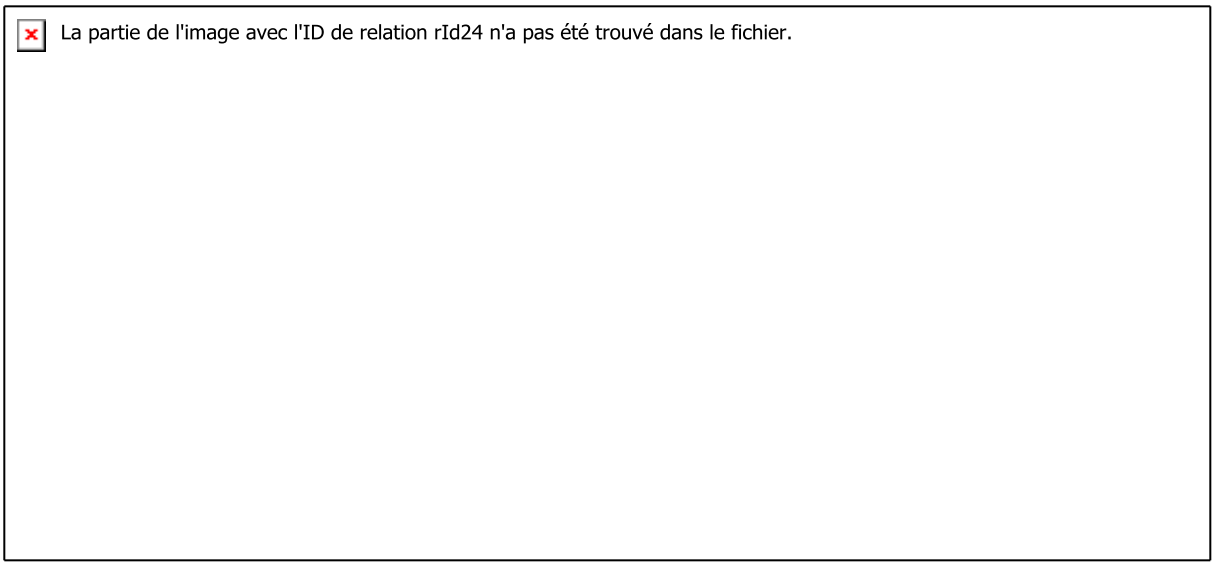

Figure 7 - Architecture à 4 couches [OMG03a]

- Le niveau M0 (ou instance) correspond au monde réel. Ce sont les informations réelles de l'utilisateur, instance du modèle de M1.

- Le niveau M1 (ou modèle) est composé de modèles d'information. Il décrit les informations de M0. Les modèles UML, les PIM et les PSM appartiennent à ce niveau. Les modèles M1 sont des instances de méta modèle de M2.

- Le niveau M2 (ou méta modèle), il définit le langage de modélisation et la grammaire de représentation des modèles M1. Le méta modèle UML qui est décrit dans le standard UML, et qui définit la structure interne des modèles UML, fait partie de ce niveau. Les profils UML,

qui étendent le méta modèle UML, appartiennent aussi à ce niveau. Les méta modèles sont des instances du MOF.

- Le niveau M3 (ou méta méta modèle) est composé d'une unique entité qui s'appelle le MOF. Le MOF permet de décrire la structure des méta modèles, d'étendre ou de modifier les méta modèles existants. Le MOF est réflexif, il se décrit lui-même.

VI. MOF et QVT

MOF 2.0 a été développé en prenant en compte la nouvelle infrastructure d'UML 2.0, permettant ainsi à UML 2.0 d'être utilisé afin de représenter des méta modèles de MOF 2.0. MOF 2.0 introduit une importante nouveauté, c'est la norme de QVT* (Query View Transformation) qui est un langage de requêtes et de transformation de modèles. Le QVT est un effort de normalisation des mécanismes, des techniques et des notations nécessaires à la transformation de modèles source en modèles cible. L'objectif du QVT est de définir et d'établir une manière standard pour interroger les modèles du MOF, pour créer des vues et pour définir les transformations de modèles [Rapela04]. Pour cela, trois éléments fondamentaux du MOF 2.0 sont utilisés.

- Une question (query) est une expression bien formée dans un langage d'interrogation défini comme OCL 2.0 par exemple. Un modèle est évalué par une question qui renvoie les objets, instances de ce modèle satisfaisant à la condition formulée dans la question.

- Une vue (view) est un modèle qui est dérivé d'un modèle différent sur des aspects spécifiques. Elle dépend d'un modèle source. Elle peut être obtenue en appliquant des transformations à ce modèle. Si un changement est effectué sur un modèle source alors la vue change elle aussi.

- Une transformation (transformation) produit un modèle cible en un modèle source. La définition d'une transformation spécifique décrit le rapport entre le modèle source et le modèle cible au niveau du méta modèle. Ainsi, la transformation définie est applicable à tous les modèles, instances du méta modèle source.

Ces deux standards sont importants dans la démarche MDA. Ils ont été spécialement repensés pour améliorer la transformation des modèles et la génération de code.

VII. UML (Unified Modeling Language)

Le langage UML (Unified Modeling Language) a été adopté par l'OMG comme standard de modélisation de système informatique en novembre 1997. Les concepts définis par l'UML sont très proches de ceux définis par le MOF. Ainsi, le MOF utilise les représentations graphiques de l'UML.

L'UML a apporté au domaine de la méta-modélisation sa notation graphique et ses concepts objet. Le langage UML permet la modélisation de systèmes indépendamment de toute démarche ou de plateforme. C'est pourquoi, dans la cadre du MDA, UML peut-être utilisé pour décrire des plates-formes (PDM), des organisations ou des situations (PIM) ou la

plupart des systèmes logiciels (PSM). L'OMG veut intégrer l'UML au sein même des applications de développement afin d'éviter que le passage au code marque la fin de l'utilisation des modèles UML. Ce passage au code exécutable devra se faire de façon automatique, solutionnant ainsi un bon nombre de problèmes de maintenance et d'évolution des logiciels.

1. Vue Structurelle

1.1 Diagramme de classe

Permet de visualiser une vue statique de la conception, qui incluse les packages, les classes, les interfaces et leurs relations.

1.2 Diagramme d'objet

Ce diagramme montre des objets (instances de classes dans un état particulier) et des liens (relations sémantiques) entre ces objets.

1.3 Diagramme de composant

Les diagrammes de composants permettent de décrire l'architecture physique et statique d'une application en terme de modules : fichiers sources, librairies, exécutables,…etc. Ils montrent la mise en oeuvre physique des modèles logiques avec l'environnement de développement.

Les dépendances entre composants permettent notamment d'identifier les contraintes de compilation et de mettre en évidence la réutilisation de composants

1.4 Diagramme de déploiement

Les diagrammes de déploiement montrent la disposition physique des matériels qui composent le système et la répartition des composants sur ces matériels. Les ressources matérielles sont représentées sous forme de noeuds.

Les noeuds sont connectés entre eux, à l'aide d'un support de communication. La nature des lignes de communication et leurs caractéristiques peuvent être précisées.

Les diagrammes de déploiement peuvent montrer des instances de noeuds (un matériel précis), ou des classes de noeuds.

2. Vue comportementale

2.1 Diagramme de cas d'utilisation

Montre le comportement du système, sous système ou classes.

2.2 Diagramme d'activité

UML permet de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation, à l'aide de diagrammes d'activités. Une activité représente une exécution d'un mécanisme, un déroulement d'étapes séquentielles.

Le passage d'une activité vers une autre est matérialisé par une transition. Les transitions sont déclenchées par la fin d'une activité et provoquent le début immédiat d'une autre (elles sont automatiques).

2.3 Diagramme d'état

Ce diagramme sert à représenter des automates d'états finis, sous forme de graphes d'états, reliés par des arcs orientés qui décrivent les transitions.

Les diagrammes d'états-transitions permettent de décrire les changements d'états d'un objet ou d'un composant, en réponse aux interactions avec d'autres objets/composants ou avec des acteurs.

2.4 Diagramme de séquence

Les diagrammes de séquences permettent de représenter des collaborations entre objets selon un point de vue temporel, on y met l'accent sur la chronologie des envois de messages. Contrairement au diagramme de collaboration, on n'y décri pas le contexte ou l'état des objets, la représentation se concentre sur l'expression des interactions.

2.5 Diagramme de collaboration

Les diagrammes de collaboration montrent des interactions entre objets.

3. Points fors et faibles d'UML

1. Points forts : certaines caractéristiques d'UML lui font très utilisé par l'approche MDA [Frankel03] :

1. ***Séparation des syntaxes abstraite et concrète*** : le modèle formel d'UML est basé principalement sur la syntaxe abstraite. La syntaxe concrète est la notation graphique. cette séparation est exigence clef de l'approche MDA.
2. ***Mécanismes d'extension :*** Un profil UML permet d'adapter le langage UML à un domaine qu'il ne pouvait couvrir correctement. Les profils sont utilisés pour la génération de PIM ou de PSM mais aussi pour passer du PIM au PSM. Les spécificités de chaque plate-forme peuvent être modélisées grâce aux mécanismes d'extension d'UML définis par les profils UML. Par exemple, les stéréotypes permettent l'ajout de nouveaux éléments au méta-modèle.
3. ***Modélisation indépendante des plates formes :*** la modélisation se fait indépendamment de la plate forme finale telle que : JEE, CORBA, .Net, JAVA, C, C#....
4. ***Normalisé :*** UML est une norme.

2. Points faibles : malgré ces points forts d'UML, ce dernier présente certaines faiblesses :

1. ***limitation des profiles :*** les mécanismes d'extension sont restrictives et l'ajout de nouveaux éléments du langage est très limité.

2. ***Large :*** Le méta modèle UML est large et les modèles d'activité supporté sont peu séparé. Il est difficile d'utiliser juste une partie d'UML sans l'interdépendance avec les autres parties. Les DTD XML reflètent ce problème.
3. ***Aucun standard pour l'échange des diagrammes :*** UML n'offre aucun moyen pour échanger les diagrammes entre outils, cela à cause de l'absence de méta modèles pour les diagrammes UML. C'est-à-dire, si on essaye d'exporter un diagramme UML d'un outil vers un autre, seules les informations sémantiques peuvent être exporté mais il n'est pas possible d'exporter les propriétés du diagramme.
4. ***Absence de méta modèle pour OCL :*** UML n'a pas de méta modèle pour le langage OCL (Object Constraint Language).

VIII. XML Meta Data Interchage (XMI)

UML (Unified Modeling Language) est un langage graphique utilisé pour la conception orientée objet, et MOF (Meta Object Facility) utilise un sous-ensemble de UML pour décrire les objets manipulés par les outils de conception. XMI (XML Metadata Interchange) indique comment les modèles MOF peuvent être traduits en XML. Le but de ces standards est de permettre à des ateliers logiciels d'explorer et d'échanger les définitions des structures de données, leurs propriétés, les relations les unissant, etc.Enfin, XMI (XML Metadata Interchange) indique comment les modèles MOF peuvent être traduits en XML. Le but de ces standards est de permettre à des ateliers logiciels d'explorer et d'échanger les définitions des structures de données, leurs propriétés, les relations les unissant, etc.

Les méta modèles MOF et UML sont décrits par des DTD et les modèles traduits dans des documents XML conformes à leur DTD correspondante. XMI a l'avantage de regrouper les méta données et les instances dans un même document ce qui permet à une application de comprendre les instances grâce à leurs méta données. Ceci facilite les échanges entre applications et certains outils pourront automatiser ces échanges en utilisant un moteur de transformation du type XSLT [Blanc04, Bezinvin02a, Bezinvin02a, Lemesle00, Oum03, Sriplakich03].

IX. Types de Transformation

Pendant le cycle de développement MD on peut identifier plusieurs transformations. Ces dernières sont comme suit :

1. PIM vers PIM

On utilise ce type de transformation pour enrichir, filtrer ou spécialiser les informations d'un modèle sans rajouter aucune information liée à la plate-forme. Le passage du modèle PIM vers un autre modèle PIM est appelé raffinement. Le raffinement est le processus consistant à introduire des détails supplémentaires dans le premier modèle. Un exemple de transformation PIM vers PIM est de masquer des éléments afin de s'abstraire des détails fonctionnels. Un autre exemple est le passage du modèle d'analyse à celui de conception. Cependant, ces transformations ne sont pas toujours automatisables.

2. PIM vers PSM

Ce type de transformation peut se faire après l'obtention d'un PIM suffisamment raffiné en le spécialisant vers une plate-forme donnée. Il est alors possible de passer d'un modèle indépendant à un modèle dépendant en appliquant des règles de transformation. Ces dernières devront être généralisées et capitalisées pour obtenir dans le futur une automatisation importante [Bezinvin02a]. Cette transformation consiste à ajouter au PIM des informations propres à une plate-forme technique telles que J2EE, .NET ou CORBA, …C'est le PDM, un profil ou le MOF (Meta-Object Facility) qui contient les caractéristiques de transformation.

3. PSM vers PSM

Obtenir un PSM vers PIM n'est pas toujours suffisant pour permettre la génération du code, pour cela on utilise une transformation du PSM vers PSM en utilisant des formalismes intermédiaires. Par exemple, pour générer un code C++, à partir d'un formalisme en UML, un passage d'UML vers SDL* (Specification and Description Language) puis de SDL vers C++ pourrait être utilisé. La transformation PSM à PSM (raffinement) s'effectue lors de phases du déploiement, d'optimisation ou de reconfiguration [Bezinvin02a].

4. PSM vers PIM

Dans ce type de transformation, on parle de la rétroingénierie (reverse engineering) dans laquelle on cherche à revenir à un modèle indépendant de plate-forme (PIM) à partir d'un modèle spécifique de plate-forme (PSM) ou éventuellement du code. Il s'agit d'un processus assez complexe à réaliser et difficilement automatisable. Ces transformations sont néanmoins nécessaires pour permettre l'intégration d'applications existantes dans le processus MDA [Bezinvin02a].

X. La transformation et les standards utilisés

Passer d'un modèle à un autre ou bien faire une transformation apporte plusieurs avantages sur tous le cycle de développement. Les techniques de transformation de modèles sont donc au coeur du MDA. Elles peuvent être manuelles, assistées par des outils ou entièrement automatiques.

1. La transformation par annotation ou marquage

La transformation par marquage consiste à utiliser des annotations ou tags pour annoter (tagger) un PIM puis, à appliquer des règles de transformation pour convertir ce PIM en PSM. Les règles de transformation sont issues des profils UML ou à l'aide des patrons de conception comme montré dans la figure 8. L'utilisateur fait les annotations manuellement. Pour pouvoir revenir du PSM vers le PIM, on peut conserver des traces de cette transformation. Ces traces permettent de maintenir une correspondance entre les éléments source et cible. Depuis la version 2 du MOF, il est aussi possible d'utiliser les méta-modèles pour annoter les PIM mais cela reste encore au stade de la recherche [Belaunde03, Belaunde04, Bezivin03a, Joaquin03a, Joaquin03b, Joaquin03b, Muller04, OMG03b].

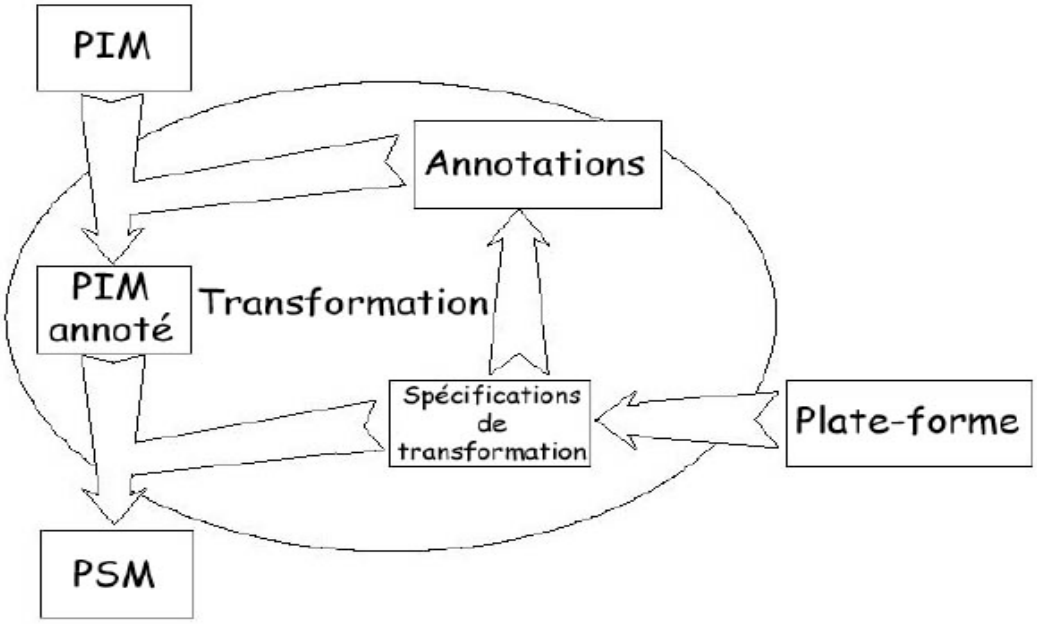

Figure 8 - Transformation par annotation ou marquage [OMG03a]

1. La transformation par méta-modèle

Cette transformation peut être effectuée en utilisant les méta-modèles. Dans un premier temps, il faut spécifier les règles de transformation (décrivant la correspondance entre le langage source et le langage cible), par la suite il faut appliquer ces règles au PIM pour produire le PSM. Comme les méta-modèles peuvent être exprimés dans d'autres langages que l'UML, les profils UML sont exclus de cette transformation.

L'implémentation de cette technique par des outils n'est pas encore maturée. Le PIM utilise le même langage que celui de son métamodèle, donc tout modèle a un méta-modèle. Par exemple, le PIM peut être décrit en UML avec un méta-modèle UML et le PSM peut être spécifique d'une plate-forme particulière comme WSDL. La transformation se base sur un langage source (ici UML) et sur un langage cible (ici WSDL). Dans notre cas d'étude, diagrammes UML seront convertis en WSDL grâce à la transformation [Belaunde03, Bezivin03a, Joaquin03a, Joaquin03b, Joaquin03b, Muller04, OMG03b].

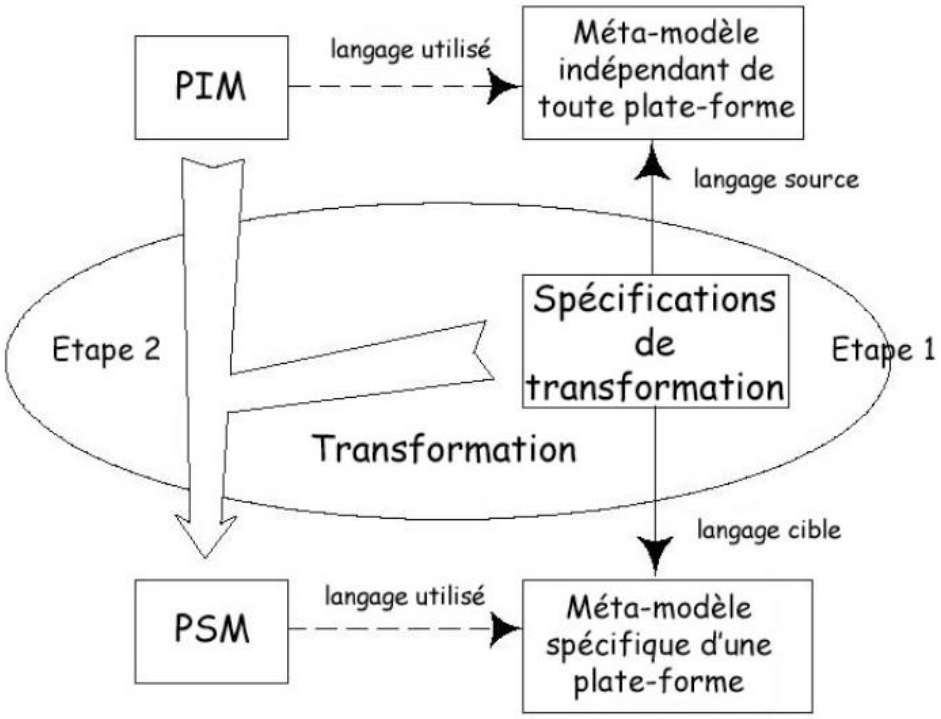

Figure 9 - Transformation par méta modèle [OMG03a]

Pour avoir plus de souplesse et de précision dans la création des modèles PSM, il est possible d'utiliser des approches hybrides. Ci-dessus nous avons présentées les techniques les plus utilisées. Il existe d'autres types de transformations comme la fusion de modèle ou la transformation par addition d'informations mais elles restent au stade théorique.

La transformation par les profils UML est différente que celle par les méta-modèles. En effet, les méta-modèles ont un langage de plus haut niveau par rapport à celui des profils UML. Cela apporte des avantages lors de l'échange des descriptions XMIs. Une description XMI du méta-modèle sera plus claire et concise que celle issu d'un profil UML. Cependant, certains reprochent aux méta-modèles d'être plus complexes à mettre en œuvre et regrettent le manque d'outils facilitant et masquant cette tâche. Ils préfèrent les profils UML voir [Vale08].

2. L'approche en double Y

Cette transformation synthétise les recherches qui se font actuellement sur la transformation des méta-modèles. L'approche simple Y ne permet pas de séparer les exigences non fonctionnelles des exigences fonctionnelles. Elle utilise deux branches, dans la première on trouve les spécifications fonctionnelles, et dans l'autre les informations spécifiques à une plate-forme, c'est une insuffisance que l'approche double Y cherche régler [*Belaunde04*, Bezivin03a, Bezivin02a, Bezivin02b]. Cette approche permet d'avoir une vue globale de toutes les étapes de la démarche MDA. Elle permet de conserver une indépendance vis à vis des plates-formes durant la majorité de la phase de développement. Pratiquement peu de personnes utilisent l'appellation « CIM »pour le tout premier modèle, il peut être aussi appelé PIM fonctionnel. Il peut être raffiné en de nouveaux PIM si nécessaire.

Les exigences non fonctionnelles et la qualité de service (performance, sécurité, persistance…), qui sont indépendantes d'une plate-forme et/ou d'une architecture particulière sont présentes la branche en face de celle du CIM. La transformation peut s'effectuer de différentes façons. Par exemple, elle peut-être soit manuelle, soit automatique, ou intermédiaire ; et utiliser une transformation par annotation ou une transformation par méta-modèle. Les deux branches se rejoignent et elles donnent un nouveau modèle (toujours indépendant d'une plate-forme mais avec en plus des exigences non fonctionnelles et/ou des qualités de service) par une transformation. Si besoin, un raffinement du PIM issu de la transformation peut être effectué. La branche en bas à droite contient des informations spécifiques à une plate-forme. Grâce à ces informations, le dernier PIM non fonctionnel va être transformé en PSM. Ce PSM est donc un modèle spécifique à une plate-forme avec des exigences fonctionnelles et non fonctionnelles. Il va par raffinements successifs donner du code [Bezivin02b, Farcet03a, Farcet03b].

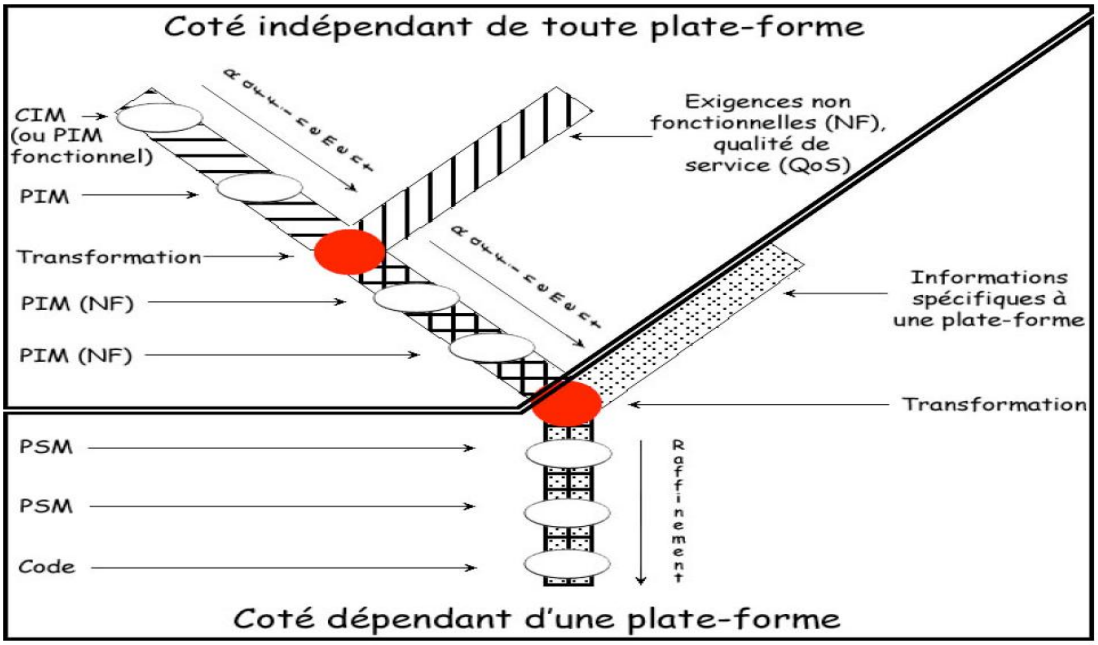

Figure 10 - Cycle de transformation en double Y[OMG03a]

XI. Caractéristiques des langages de transformation

Plusieurs langages de transformation ont été proposés dans la littérature après la révélation du RFP QVT, [Bezivin03b] [Marschall03] [Patrascoiu04] ou soumis directement à l'OMG [DSTC04] [IOSGPT04] [Gardner03].

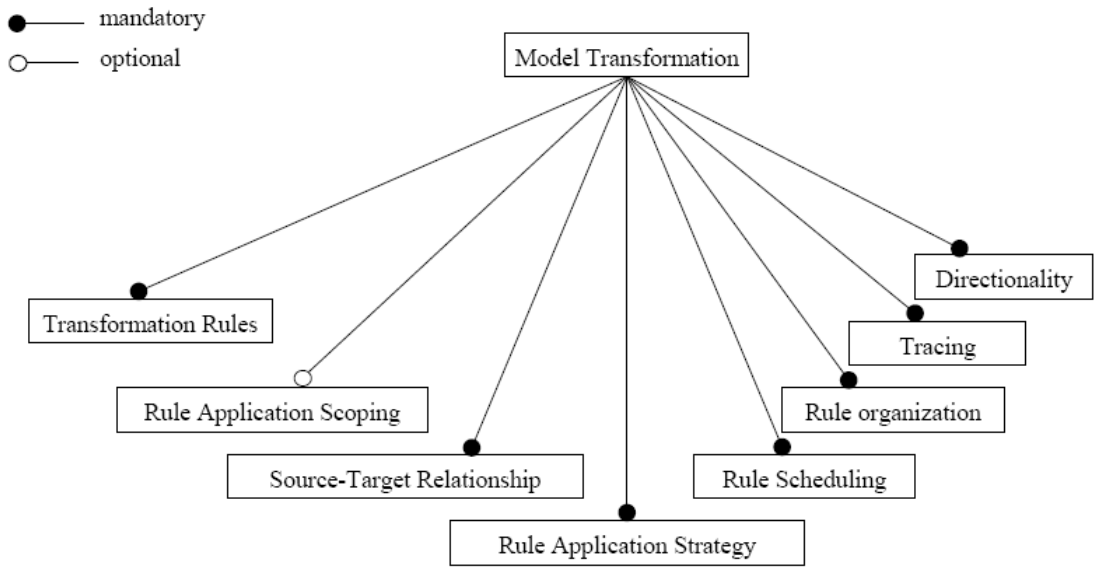

Figure 11 – Le diagramme des caractéristiques de langages de transformation de modèle [Czarnecki03]

Pour résoudre le problème de définition de transformation de modèles, ces langages présentent des approches différentes. On trouve une taxonomie basée sur les caractéristiques présentes dans la majorité des propositions de langages de transformation Dans [Czarnecki03], K. Czarnecki et S. Helsen. Ces caractéristiques sont présentées dans La figure 16. Plusieurs paramètres peuvent distinguer les langages de transformations :

A. Les règles de transformation : une règle de transformation contient deux parties distinctes, un LHS (Left Hand Side) et un RHS (Right-Hand Side). Le LHS permet d'accéder au modèle source, et le RHS permet d'accéder au modèle cible. Les LHS et RHS peuvent être une combinaison de :

1–Variables : les variables permettent la manipulation des éléments du modèle source et/ou cible. Les variables peuvent être non typées, syntaxiquement typées ou sémantiquement typées.

2–Patron : les patrons peuvent être classés suivant la forme, la syntaxe et le typage. La forme d'un patron peut être strings, termes ou graphes. Les patrons string sont utilisés en modèles textuels (textual templates). Les termes et les graphes sont utilisés dans la transformation modèle-vers-modèle. Quant à la syntaxe, elle peut être abstraite ou concrète. Un patron peut être non typé, syntaxiquement typé ou sémantiquement typé.

3–Logique : la logique peut être non exécutable ou exécutable. La logique non exécutable permet de spécifier les relations entre modèles. L'exécutable peut être impérative ou déclarative. La forme déclarative comprend les requêtes OCL pour la récupération des éléments d'un modèle et la création implicite des éléments cibles par le biais des contraintes. La forme impérative contient un type de langage de programmation qui fait appel à une API

pour manipuler les modèles directement, par exemple JMI (Java Metadata Interface) [Java02] fournit une API pour accéder à un dépôt MOF (repository MOF).

D'autres informations peuvent caractériser un règle de transformation :

A– Séparation Syntaxique : la syntaxe permet la distinction explicite entre un RHS et un LHS ou cette distinction peut ne pas avoir lieu.

B– Bidirectionnalité : une règle peut être exécutable dans une seule direction ou dans les deux directions (bidirectionnelle). La règle bidirectionnelle est plus complexe et plus difficile à implémenter [Kent03].

C–Paramétrage : les règles de transformation peuvent avoir des paramètres de contrôle additionnels permettant la configuration ou l'adaptation.

D–Structures intermédiaire : quelques approches demandent la construction de structures de modèles intermédiaires.

B. Portée de l'application de la règle : elle permet à une transformation de restreindre les parties d'un modèle qui participe à la transformation.

1–Source : dans ce cas, nous pouvons établir une portée moindre que celle du modèle source entier.

2– Cible : dans ce cas, nous pouvons augmenter la portée du modèle cible.

C. Relation Source-Cible : plusieurs approches imposent la création d'un nouveau modèle cible qui doit être séparé du modèle source. D'autres approches imposent que le modèle source et le modèle cible soient le même modèle (elles permettent uniquement la mise à jour sur place, ou in-place update).

D. Stratégie d'application de la règle : une règle s'applique depuis une localisation spécifique à l'intérieur de la portée du modèle source. Dans le cas où elle est activée depuis plus d'une localisation, la stratégie de l'application de la règle doit être spécifiée. Elle peut être :

1–Déterministe : la stratégie suit un protocole déterminé et connu d'avance.

2–Non déterministe : une localisation peut être choisie ou plusieurs peuvent être appliquées de façon concurrente.

3–Itérative : l'utilisateur doit dire quelle localisation doit être appliquée à la règle.

E. Ordonnancement d'une Règle : détermine l'ordre d'application des règles individuelles. Le mécanisme de scheduling peut être classé en quatre groupes :

1– Forme : l'aspect de l'ordonnancement peut être implicite et explicite. L'ordonnancement implicite implique que l'utilisateur n'a pas un contrôle explicite dans l'algorithme d'ordonnancement.

L'ordonnancement explicite fournit des constructions dédiées au contrôle de l'ordre d'exécution.

2– Sélection d'une règle: les règles peuvent être sélectionnées par une condition spécifique, ou choisies de façon non déterministe, ou sélectionnées par un mécanisme de priorité, ou encore sélectionnées de manière interactive.

3–Itération : la récursivité, la boucle et le point fixe (fixpoint) se trouvent parmi les mécanismes d'itération de règles.

4–Phases : cet aspect fait référence à l'organisation du processus de transformation en plusieurs étapes.

F. Organisation de règles : détermine la composition et la structuration de plusieurs règles de transformation. Elle peut être classée en :

1– Mécanisme de modularité : permet l'organisation de règles en modules.

La création de modules permet à un module d'importer les éléments d'un autre module.

2– Mécanisme de réutilisation : il fournit une manière de définir une règle basée sur une ou plusieurs autres règles. Parmi les mécanismes de réutilisation, nous citons : composition logique, héritage entre règles, dérivation, extension, spécialisation, héritage entre modules.

3–Structure d'organisation : les règles peuvent être organisées en conformité avec la structure du métamodèle source, ou du métamodèle cible, ou encore avoir une organisation spécifique.

G. Trace : les transformations peuvent enregistrer des liens entre les éléments source et cible. Ces liens peuvent être utiles pour l'analyse de la performance, la synchronisation entre modèles, les débuggeurs orientés modèle, et la détermination de la cible d'une transformation.

H. Directionnalité : indique la direction d'une transformation.

1– Unidirectionnelle : la transformation est exécutée dans une seule direction, dans laquelle un modèle cible est déduit à partir d'un modèle source.

2– Bidirectionnelle : la transformation peut être exécutée dans les deux directions. Elle fournit également le support nécessaire pour la synchronisation entre modèles.

Cette taxonomie de K. Czarnecki et S. Helsen a été crée en prenant en considération les approches modèle-vers-modèle et modèle-vers-code de façon uniforme. Ces deux approches présentent plusieurs similarités. En général, la transformation de modèle vers du code peut être vue comme une transformation modèle vers modèle, puisque le code est aussi un modèle. Cependant, pour des raisons pratiques de réutilisation de la technologie de compilateur existante, le code est souvent généré comme texte simple, qui peut être compilé.

L'approche modèle vers modèle est encore en évolution et en expérimentation alors que l'approche modèle-vers-code est la plus diffusée et utilisée.

L'approche modèle-vers-code peut être :

1. Orientée visiteur : Pour s'affranchir de la représentation interne d'un modèle et écrire le code comme un texte, cette approche fournit un mécanisme de visiteur.

2. Orientée modèle : un template consiste habituellement en un texte cible qui contient des méta codes pour récupérer l'information d'un modèle source et les utiliser pour remplir le texte cible.

L'approche modèle-vers-modèle peut être :

1. Manipulation directe : cette approche fournit une représentation interne de la représentation du modèle ainsi qu'une API pour la manipuler. Elle est habituellement réalisée comme un framework orienté objet.

2. Relationnelle : cette catégorie groupe les approches déclaratives dans lesquelles les relations mathématiques constituent le concept principal. L'idée sous-jacente est d'établir un type de relation entre un élément source et cible et de la spécifier en utilisant des contraintes.

3. Orientée transformation de graphe : cette catégorie utilise la théorie des graphes.

4. Orientée structure : dans cette catégorie, deux étapes peuvent être distinguées : l'une concerne la création de l'architecture hiérarchique du modèle cible, et l'autre l'établissement des valeurs des attributs et des références dans le modèle cible.

5. Hybride : est une composition des approches déclarative et impérative.

6. approches spécifiques : Dans la spécification CWM (Common Warehouse Metamodel) de l'OMG un framework de transformation est proposé. Ce framework fournit un mécanisme pour lier les éléments source et cible, mais la dérivation des éléments cibles doit être réalisée dans un langage concret, lequel n'est pas défini par CWM. Une autre approche significative est XSLT. Depuis que les modèles peuvent être sérialisés à la façon de XML (par exemple, XMI), XSLT peut être utilisé pour réaliser la transformation.

XII. Conclusion

L'une des principaux objectifs de l'approche MDA est de séparer la conception de l'architecture finale. Cette démarche prometteuse suscite un réel intérêt chez bon nombre d'industriels et de développeurs. Elle bouleverse complètement notre façon de concevoir une application et ouvre de nouveaux horizons pour le génie logiciel qui ne sera plus dépendant des évolutions technologiques. En effet, la séparation de la logique métier de l'entreprise de son implémentation physique apporte des avantages sur tous les niveaux : production, qualité, et coût.

Une importance particulière de la démarche MDA, est la notion de transformation et notamment l'utilisation des outils (donc automatisation de la transformation), ce qui permet de basculer entre plateformes facilement et mêmes si de nouvelles plateformes viendront elles seront supportées facilement. Malheureusement, jusqu'à présent cette approche n'est pas

encore maturée car elle exige de bonnes compétences en UML, et la transformation de modèles est généralement difficile à automatiser et à implémenter.

Partie 2

Contribution

Chapitre 3

Services web sensibles au contexte

I. Introduction

Les systèmes sensibles au contexte cherchent à satisfaire les utilisateurs à tout moment, n'importent où ils se trouvent et avec n'importe quel dispositif. Prendre en compte le contexte dans les web services est une autre difficulté qui s'ajoute à la complexité des structures XML manipulées.

Dans ce chapitre, après avoir présenté notre problématique et les objectifs qu'on a fixé dans ce travail, nous essayerons d'expliquer le processus de développement des services web sensible au contexte. En suite nous allons présenter le méta modèle qu'on a adopté pour la modélisation des services sensibles au contexte. Puis nous présenterons l'étape de la transformation, et comment peut on prendre en compte les informations contextuelles pendant cette étape.

II. Problème

La complexité croissante des applications et l'explosion technologique ont suscité une nouvelle vision dans le monde du développement logiciel. Les utilisateurs accèdent aux applications à travers des dispositifs multiples (ordinateurs portables, téléphones mobiles, PDA …), partout (à l'hôpital, à la maison, dans le bus, à l'université, Aéroport …) et à tout moment (le matin, le soir, en hivers, en été…). Ainsi la technologie web services est devenue l'une des architectures software les plus utilisées notamment dans l'informatique pervasif, et ceci grâce à l'utilisation d'internet et aux avantages d'interopérabilité qu'elle offre.

Les utilisateurs pensent que ces web services sont sensibles à l'état actuel de leur environnement par exemple leurs localisation, leurs préférences, leurs activités…etc. ce type de web service est appelé service web sensible au contexte.

A l'heure actuelle le développement des services web sensibles au contexte est une tache très difficile car d'une part les standards actuels de la plateforme web service (exemple : UDDI, WSDL, SOAP) ne sont pas suffisants pour la description et l'exploitation des informations contextuelles. D'autre coté, malgré qu'il existe des outils de développement supportant la plateforme web service (exemple : Java2WSDL qui peut générer des descriptions WSDL à partir des classes java), le développement des services web sensibles au contexte ne peut pas bénéficier directement de tels outils, et les développeurs doivent implémenter tout ce qui est

en relation avec la gestion du contexte (la collection, la représentation, la dissémination et l'exploitation) en intégrant des modules si-sinon.

Ces dernières années, les chercheurs se penchent vers l'utilisation de l'approche MDA pour la prise en compte du contexte dans le développement des applications software. Cependant la plupart des travaux proposés se focalisent sur l'étape de la modélisation et négligent l'étape de la transformation qui représente l'un des fondements de MDA. Par conséquence le développement des services web sensibles au contexte reste une tache qui nécessite beaucoup d'effort et demande plus de temps.

Dans ce travail nous cherchons à développer des services web sensibles au contexte, en adoptant une approche dirigée par les modèles. Pour cela nous nous basons sur le méta modèle proposé par Sheng et Benatallah [Sheng05] pour la modélisation du contexte. Nous proposerons en suite une transformation vers la plateforme web service.

III. Objectifs

Dans ce mémoire nous souhaitons prendre en compte le contexte dans le développement des services web en utilisant une approche dirigée par les modèles. Pour cela, nous avons fixés les objectifs suivants :

1. *Prise en compte du contexte :* pour prendre en compte le contexte nous avons besoins d'un méta modèle permettant aussi bien la représentation des informations contextuelles que la modélisation des services. Pour atteindre cet objectif, nous nous basons sur le ContextUML qui est le méta modèle le plus fameux dans le domaine de la sensibilité au contexte. Ce dernier permet la modélisation du contexte, ainsi que la définition des actions d'adaptation selon le contexte.
2. *Transformation :* dans cette étape nous cherchons à faire la correspondance entre les fichiers obtenus à partir de la première phase (phase de modélisation) et la plate forme web service, par conséquent il est nécessaire de compter sur un transformateur capable de prendre en compte le contexte, pour cela l'un de nos objectifs est de développer ce transformateur.
3. *Développement d'une application sensible au contexte :* Pour mieux illustrer notre travail, nous allons aborder un cas d'étude dans le domaine de la gestion consistant à développer une application de gestion du transport et des livraisons sur le web.

IV. Processus de développement des services sensibles au contexte

Comme montré dans la figure 12, le développement des services sensibles au contexte commence par l'étape de modélisation où le concepteur spécifie chaque service et les informations contextuelles nécessaires à son adaptation. Ceci est garanti en utilisant un méta modèle approprié.

La modélisation se fait dans un niveau d'abstraction élevé permettant ainsi une meilleure compréhension du comportement du service. Pour cela, on utilise les diagrammes UML pour modéliser les interfaces et les orchestrations.

Les diagrammes UML peuvent être représentés sous formes de fichiers XMI à l'aide du standard XMI.

La description des modèles ou bien les fichiers XMI obtenus à partir de l'étape de modélisation sont exportés vers un Transformateur qui assure leur transformation (du format XMI en d'autres formats tels que WSDL pour les interfaces et BPEL pour les orchestrations).

Les interfaces de service sont publiées dans un Registre responsable de la publication des services, UDDI est un exemple de registre. Les orchestrations sont exécutées dans un moteur d'exécution tel que BPWS4J.

Les utilisateurs invoquent les services à partir d'un serveur tel que JBOSS qui contient l'implémentation de chaque service (on parle de déploiement).

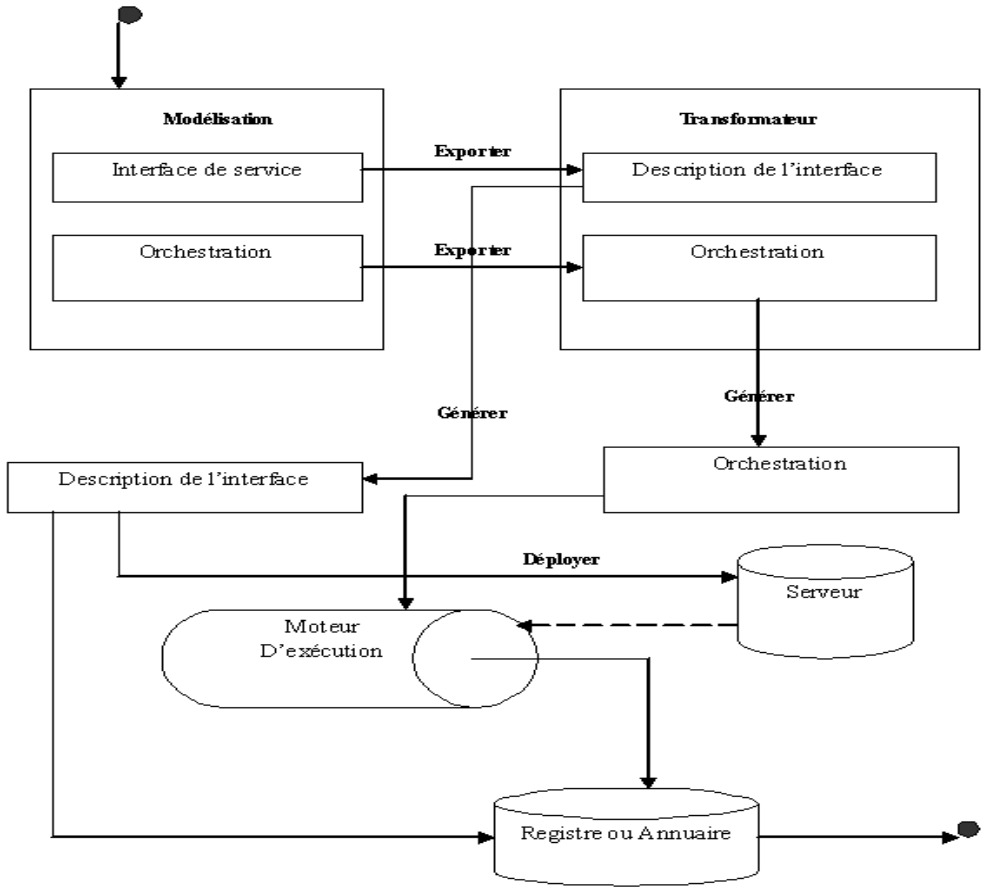

Figure 12 - Processus de développement des services sensibles au contexte

IV. 1. Modélisation des services sensibles au contexte

Il s'agit de la première étape du processus du développement, dans laquelle on modélise les services à développer ainsi que leurs contextes d'utilisation. Pour pouvoir modéliser les informations contextuelles, nous devons nous baser sur un méta modèle permettant de prendre en compte le contexte. Pour cela nous avons choisi de bénéficier du

méta modèle proposé par Sheng et Benatalah [Sheng05], qui le plus fameux dans le domaine de la sensibilité au contexte.

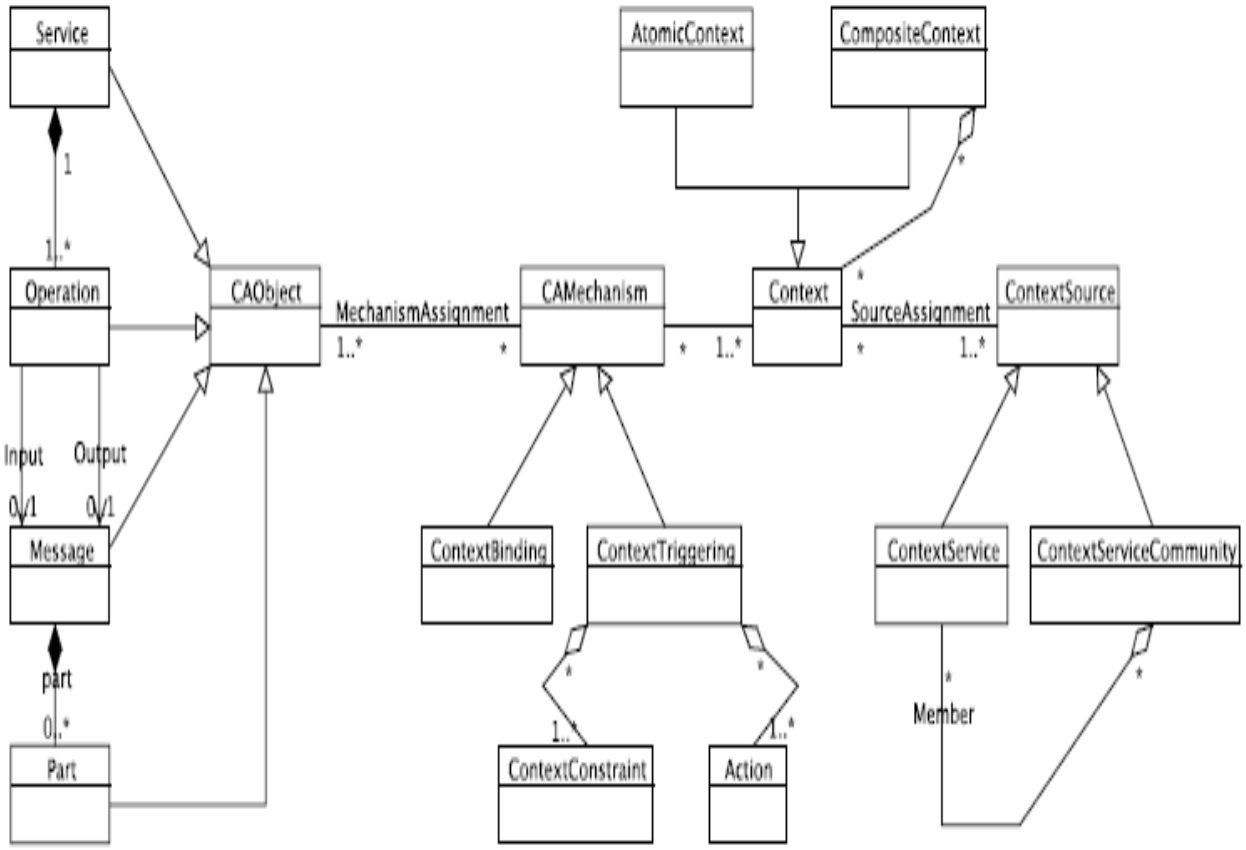

Figure 13 – Le méta modèle ContextUML[Sheng05]

Le méta modèle du ContextUML permet la modélisation de deux aspects : la modélisation du contexte, et la modélisation de la sensibilité au contexte. Il est illustré par la figure 12.

La figure ci-dessous présente un exemple de service sensible au contexte d'utilisation, ce dernier permet de récupérer la liste des engins d'une société de transport. Il contient un ensemble d'opérations permettant de consulter la liste des engins disponibles, en mission, en panne…etc.

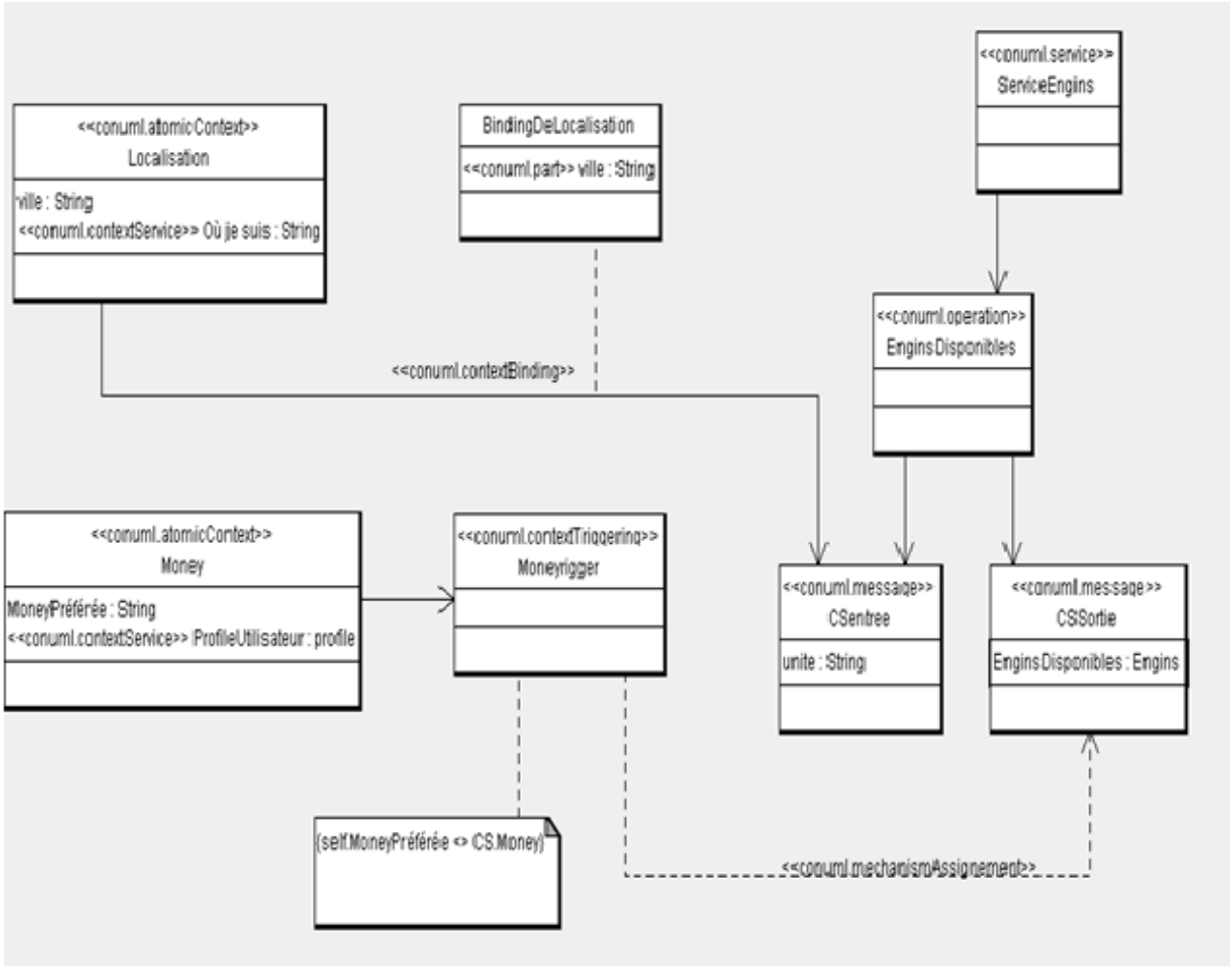

Figure 14 – Exemple de service sensible au contexte

Le résultat fourni par le service Liste des engins, peut être adapté selon le profile utilisateur et selon la localisation de ce dernier, par exemple selon la localisation de l'utilisateur on va renvoyer que les engins les plus proche. Pour le profile : selon la money utilisée (obtenue à partir du profile) le résultat sera avec la money fréquemment utilisée.

IV. 2. Transformation

La transformation de modèles est « le processus de conversion d'un modèle en un autre modèle du même système ».

Durant ces dernières années, la transformation de modèles, en particulier la création d'un langage de transformation normalisé, est devenue le centre des réflexions autour de MDA. L'OMG a lancé un appel à travers son RFP QVT pour que des propositions d'un langage de transformation soient faites. À ce jour, plusieurs d'entre elles ont été reçues, mais elles sont encore en analyse.

D'autres langages de transformation ont aussi démontré la viabilité des concepts autour de MDA, par exemple ATL (Atlas Transformation Language), Mtrans, YATL (Yet Another

Transformation Language), BOTL (Bidirectional Object oriented Transformation Language), TPL (Template Pattern Transformation Language), UMLx et MOLA (MOdel transformation LAnguage).

Ces langages de transformation sont basés sur différents paradigmes : impératives ou déclaratives, non-typés ou typés syntaxiquement ou typés sémantiquement, textuelles ou graphiques. Il est alors possible qu'un langage de transformation soit plus adapté qu'un autre dans un contexte spécifique.

Un outil de transformation de modèles prend comme entrée un modèle et le transforme en un autre modèle. Au niveau actuel de MDA, le modèle d'entrée est souvent le PIM et le modèle cible est souvent le PSM. Cependant, une transformation peut avoir un PSM comme source et un autre PSM comme cible. Une transformation peut aussi avoir un PSM comme le modèle source et un PIM comme le modèle cible. De plus, une transformation de PIM vers PIM est tout à fait possible, même si elle nécessite l'intervention humaine.

1. Processus de transformation

Le processus de transformation consiste à transformer des modèles cibles à partir des modèles sources en utilisant des règles de transformation. Ces dernières décrivent la correspondance entre des éléments du modèle source avec d'autres éléments du modèles cible. Elles peuvent être exprimées de différentes manières, en utilisant le langage OCL, par XMI, par XSLT,… etc.

Selon le besoin, une transformation peut être horizontale ou bien verticale. Par exemple si on a besoin de faire un raffinement on fait une transformation horizontale, c'est le cas de transformation de PIM vers PIM. En revanche dans le cas où on veut faire de la rétro-conception on a recours à une transformation verticale. La figure 14 présente ces deux types de transformation.

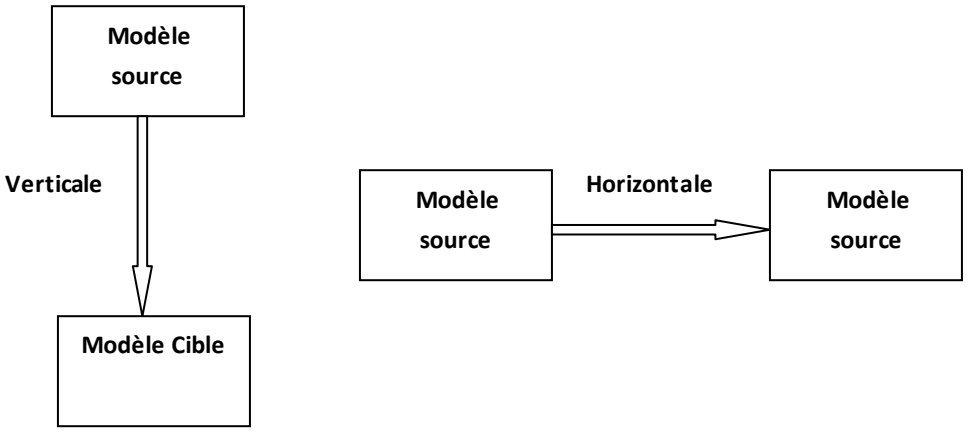

Figure 15 - Transformation Horizontale et Verticale

Les deux modèles source et cibles peuvent avoir le même méta modèle (transformation endogène), comme ils peuvent avoir des méta modèles différents (transformation exogène) tel qu'il est montré dans la figure ci-dessous :

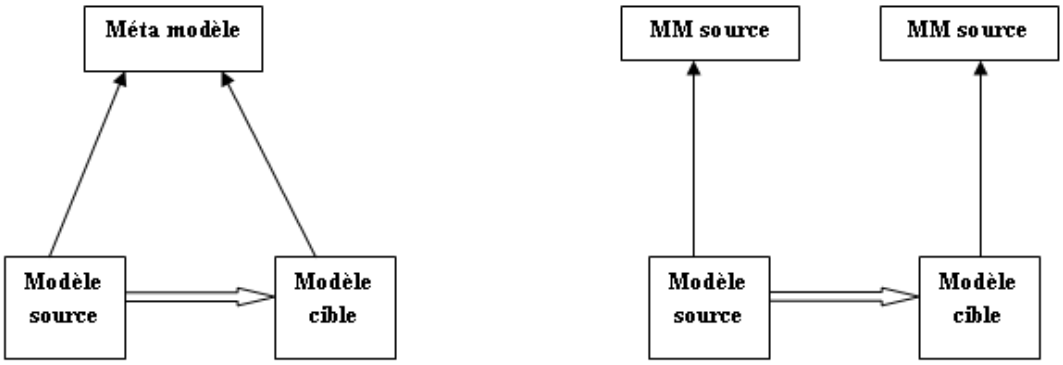

Figure 16 Transformation Endogène et Exogène

Comme on a vu dans la section précédente, il existe plusieurs façons d'implémenter les transformations de modèles, nous proposons de s'appuyer sur XML pour les implémenter.

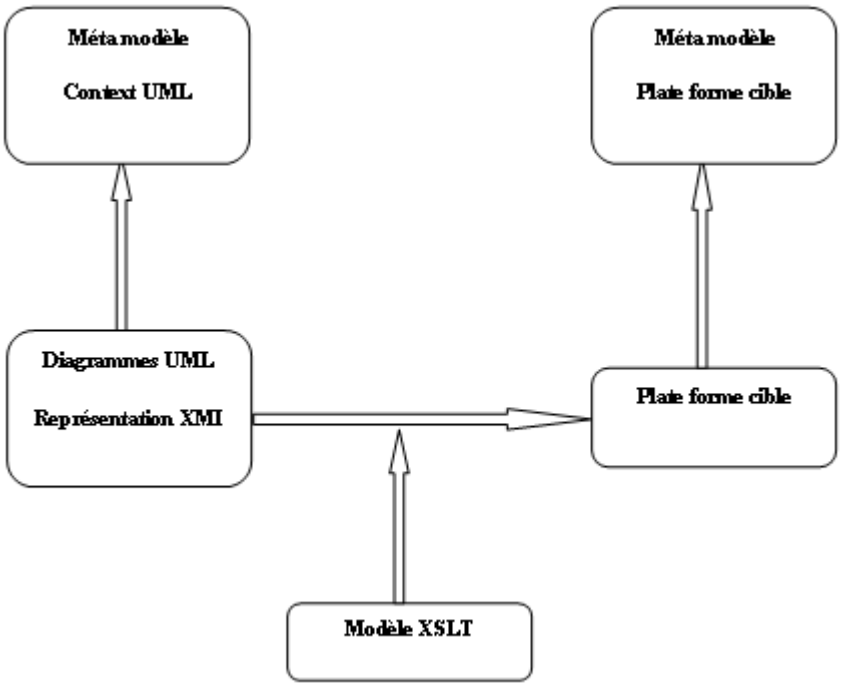

Figure 17 – Transformation en utilisant XSLT

Après l'étape de la modélisation, les diagrammes UML obtenus seront exportés vers le transformateur qui va à son tour utiliser un modèle XLST pour transformer les représentations XMI entrées en plusieurs plates forme cible, comme montré dans la figure ci-dessus.

2. Proposition de transformation du ContextUML vers WSDL

A. Partie Service

1. Service, Opération, Message, et Part

Chaque service a une description qui décrit ses opérations, c'est ce qu'on appelle contrat ou interface du service. Les opérations ont en entrée des messages d'entrées et produisent en sortie des messages de sortie. Chaque message consiste en un ensemble de paramètres ayant chacun un type.

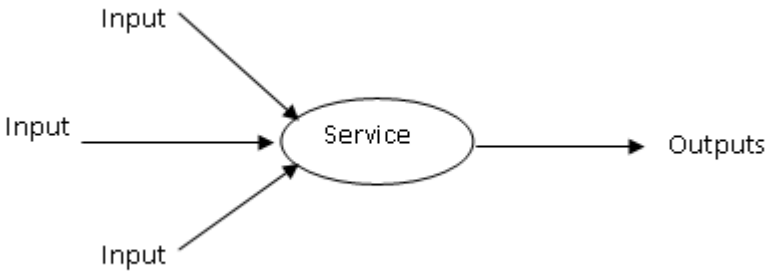

Figure 18- Service

La correspondance entre les éléments de cette partie et WSDL est évidente car, le WDSL contient tous ces éléments, donc la transformation sera comme présenté dans la figure 18.

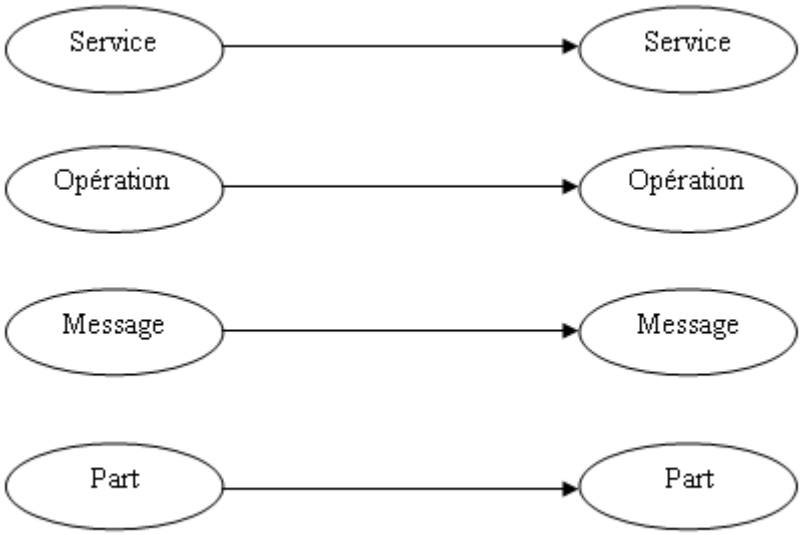

Figure 19- Transformation de la partie service

Pratiquement, cette correspondance peut être réalisée en définissant un modèle XSLT et sur lequel se base le transformateur pour transformer la partie service.

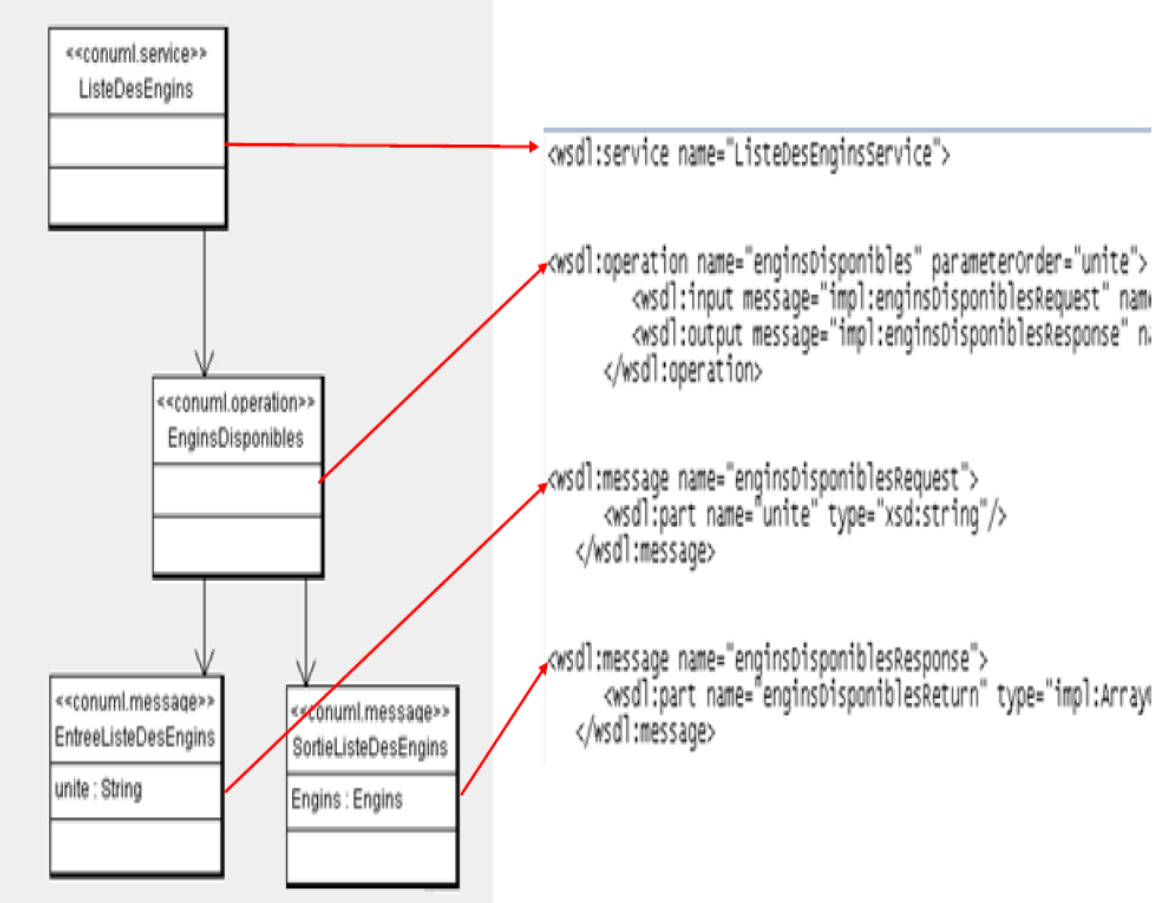

Figure 20- Exemple de la transformation de la partie service

La figure 18 illustre le résultat de la transformation de la partie service, et comme présenté dans la figure 16 les éléments de cette partie ont des correspondants qui sont des balises servant à définir les services, les opérations, les messages, et les paramètres. Donc à partir de la représentation XMI du diagramme modélisant le service on va générer les balises correspondants, et cela en définissant un modèle de transformation (ou bien a Template) en XSLT. Le modèle de transformation de cette partie aura la forme suivante :

- Pour chaque classe service créer une balise <service > avec attribut name le nom du service.
- Pour chaque classe opération créer une balse <wsdl :operation>…
- …

B. Partie Contexte

1. Context

La classe contexte permet de modéliser les informations contextuelles, elle a deux sous classes AtomicContext et CompositeContexte.

2. AtomicContext

Un contexte atomique est une information contextuelle de bas niveau qui n'est pas reliée à un autre contexte et elle peut être obtenue en utilisant une source modélisée par la classe ContexteSource.

3. CompositeContext

Le contexte composé représente des informations contextuelles de haut niveau, il se compose de contextes multiples et il ne peut pas être obtenu directement à partir d'une source. Comme exemple de contexte simple on peut citer : la température, la probabilité de pluie…etc. En revanche, un mauvais temps est un contexte composé qui est la combinaison de plusieurs contextes.

4. ContextSource

Modélise les ressources à partir desquelles on peut retrouver les informations contextuelles, il existe deux catégories de sources qui sont représentées par les deux sous classes ContextService et ContextServiceCommunity.

5. ContextService

Un ContextService est obtenu par une organisation autonome qui assure la collection, le raffinement et la dissémination de l'information contextuelle. Le ContextService Community permet de résoudre le problème d'hétérogénéité du contexte, il permet d'avoir une optimisation du choix des informations contextuelles. Il faut noter que ContextUML ne modélise pas l'acquisition du contexte, le ContextService encapsule la collection, l'interprétation et la transformation des informations contextuelles.

6. ContextServiceCommunity

Se compose de plusieurs ContextService offrant une interface unifiée. Une communauté décrit les possibilités du service (ex chercher la localisation de l'utilisateur) désiré sans référer à aucun service (ex Where I am service). Quand une opération est invoquée, la communauté est responsable de l'invocation du ContextService approprié qui va fournir l'information contextuelle requise. Cela permet de prendre en compte le contexte au moment de la demande de l'information contextuelle plutôt qu'au moment de la conception.

La sélection du ContextService approprié peut se baser sur des critères de qualité tels que : l'exactitude, la tolérance (voir chapitre 3) … etc. La qualité du contexte est un facteur très important pour l'adaptation des services, et l'imperfection des informations contextuelles peut fausser les résultats fournis par nos services.

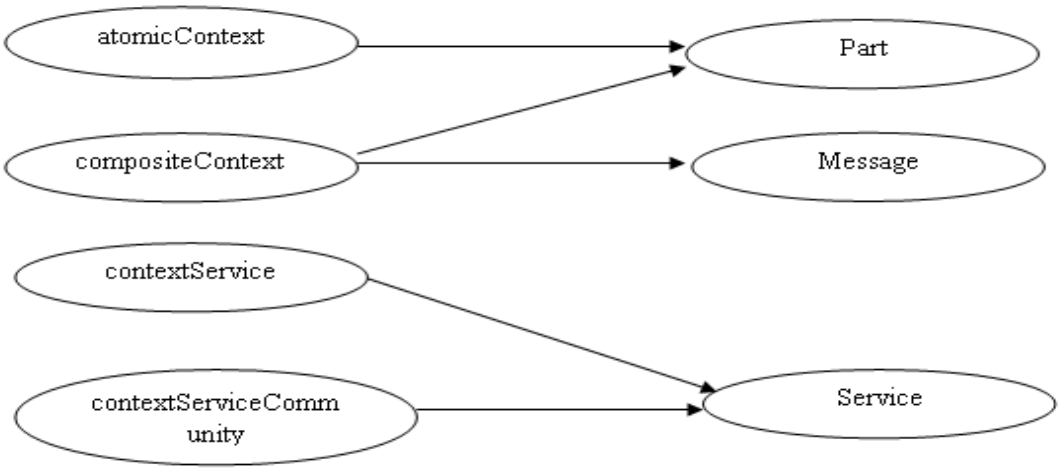

Figure 21- Transformation de la partie contexte

Pour faire la transformation de la partie modélisation du contexte, nous distinguons la classe contexte de la classe contexte source. La dernière comporte deux sous classe qui sont des services servant à retrouver les informations contextuelles donc il est évident qu'elles vont correspondre aux services, et ce choix est justifié par le fait que ces services encapsule les opérations de collection des informations contextuelles. En ce qui concerne la classe contexte, qui à son tour se compose de deux sous classes atomicContext et compositeContexte, nous proposons de transformer la première en paramètre, quand à la deuxième elle va correspondre soit à un paramètre soit à un message qui seront utilisée par les méthodes des contextSources.

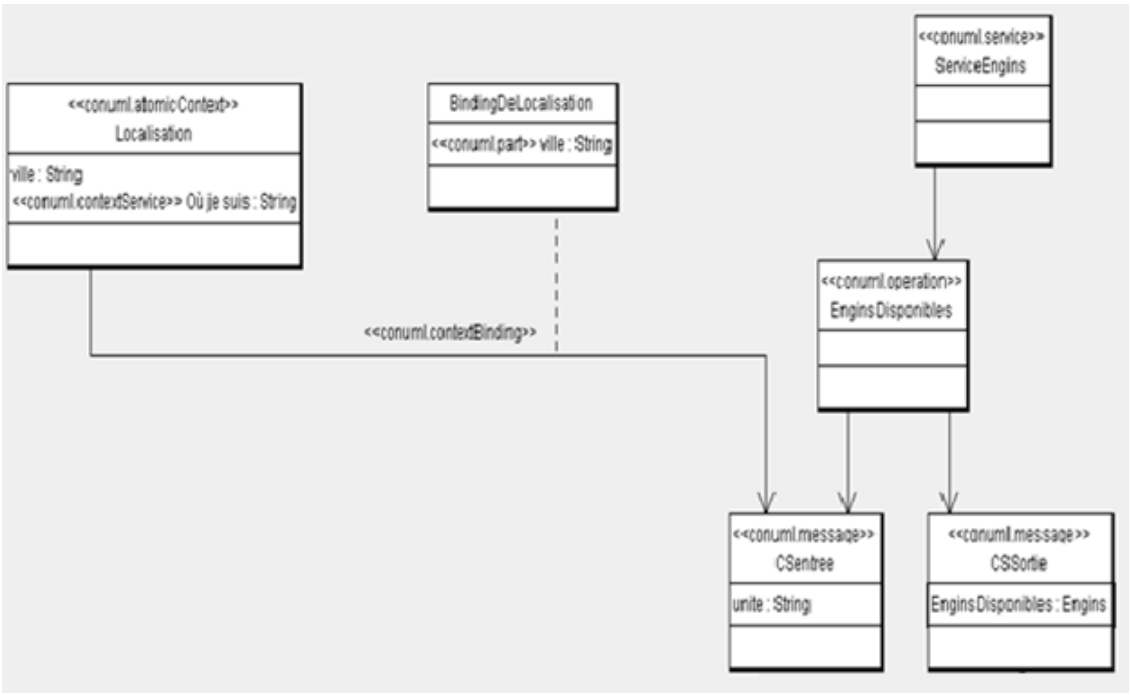

Figure 22- Modélisation du contexte

L'exemple illustré par la figure 21, représente une partie du service Lise des engins présenté dans la section précédente. Dans cet exemple l'opération Engins disponible est sensible à l'information contextuelle localisation, cette dernière peut être obtenue à partir de la source Où je suis qui est un service servant à obtenir l'emplacement (la ville) où se trouve l'utilisateur. Par la suite, cette information va être utilisée comme étant un paramètre d'entrée (au lieu du paramètre unité) de l'opération.

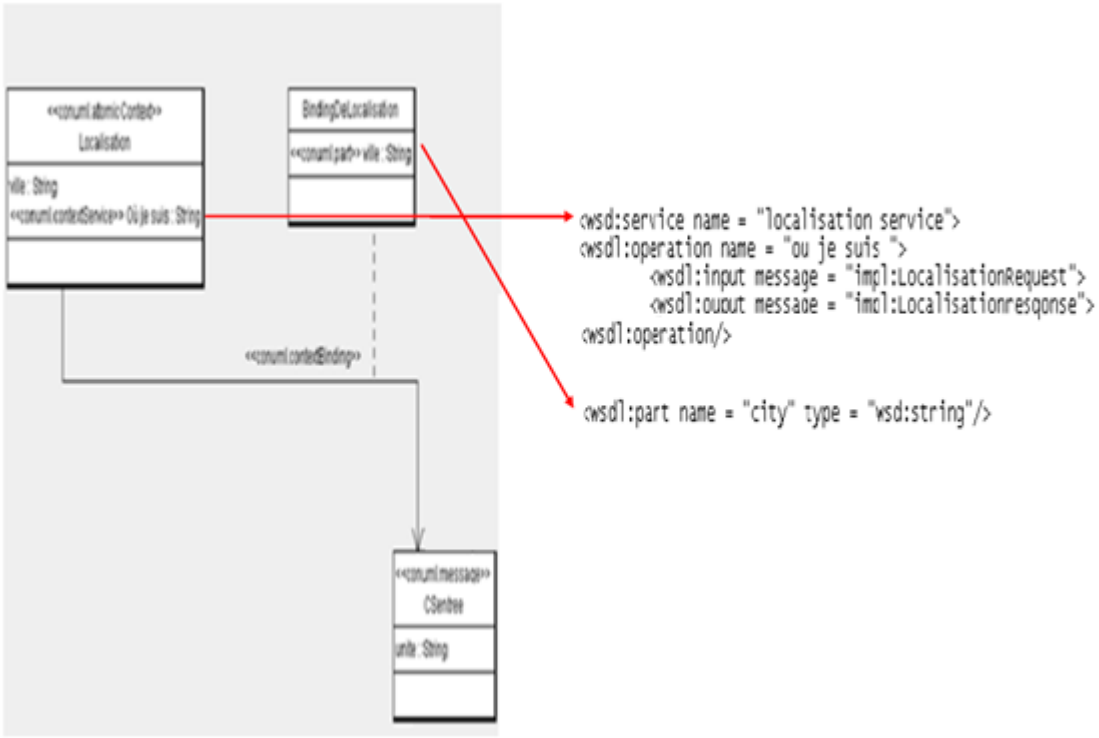

Figure 23 Exemple de la transformation de la partie contexte

La figure 22 illustre un extrait du résultat de la transformation de la partie contexte, selon cet exemple la source servant à retrouver l'information contextuelle localisation qui a comme classe ContexteSource sera transformée en service qui permet de fournir d'opération Où je suis. L'information contextuelle localisation sera à son tour transformée en paramètre d'entrée de l'opération Liste d'engin.

C. Partie sensibilité au contexte

1. ContextAwareMechanisme

Le ContextAwareMechanisme est une classe permettant de formaliser la sensibilité au contexte, on peut distinguer deux mechanismes de sensibilité au contexte dans ContextUML : la sous classe ContextBinding, et la sous classes ContextTriggering. Le ContextAwareMechanisme est attribué à un ContextAwareObject qui peut être un service, une opération, un message ou un paramètre.

2. ContextBinding

Ce mécanisme consiste à faire une liaison entre une information contextuelle est un ContexteAwareObject. Avec ce mécanisme il est possible de fournir des résultats à l'utilisateur en se basant sur une information contextuelle.

Par exemple dans un service qui récupère la liste des engins d'une unité, l'utilisateur doit fournir le nom de l'unité en question. On suppose qu'on a la localisation de l'utilisateur qui représente la ville dans laquelle il se trouve, la liaison (Binding) peut être construite entre le paramètre d'entrée unité et l'information contextuelle localisation de l'utilisateur tel qu'il est présenté dans la figure 23.

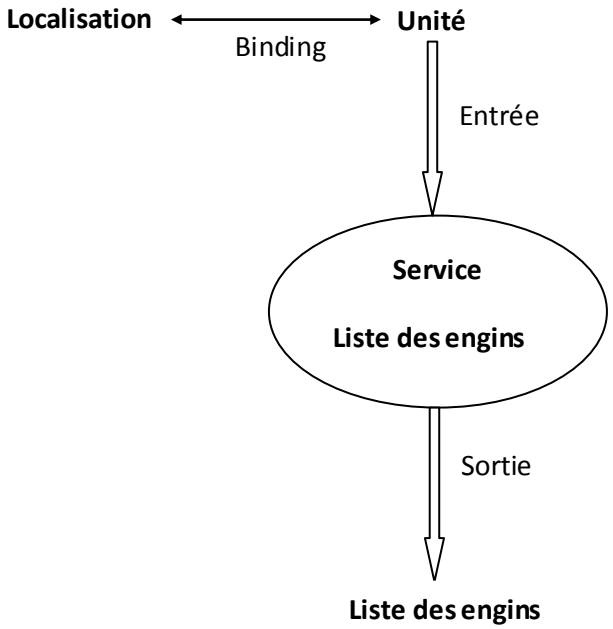

Figure 24 - Binding

Dans la figure 23, le principe de binding permet de substituer l'information unité (qui représente une entrée du service servant à récupérer la liste des engins) par l'information contextuelle localisation de l'utilisateur. Cette façon de faire est appelée Context reconfiguration (voir chapitre 3), où le service fait une reconfiguration en se basant sur une information contextuelle, ainsi le service va renvoyer que les engins de l'unité de l'utilisateur ou bien des unités les plus proches.

3. ContextTriggering

Ce mécanisme modélise la situation où le service peut être exécuté ou modifié en se basant sur une information contextuelle. Il se compose de deux parties : un ensemble de contrainte contextuelle (classe ContextConstraint), et un ensemble d'actions (classe Action).

4. Action

Les actions vont être exécutées si et seulement si toutes les contraintes contextuelles sont validées à vrai.

ContextConstraint

Une contrainte contextuelle spécifie que certain informations contextuelles doivent satisfaire certaines conditions afin d'effectuer des opérations particulières. Par exemple : mauvais temps = vrai, humidité >= 6O%...

Dans un service qui récupère la liste des restaurants d'une ville, on peut avoir un méchanisme de ContextTriggering attribué au message de ce service. La partie Contrainte peut être mauvais temps = faux, et la partie Action peut être une fonction Filtrer(M,R) où M est le message et R et l'action (ex sélectionner seulement les restaurant dans lesquelles on peut manger à l'extérieure). En revanche, si la contrainte est mauvais temps = vrai, le message de sortie peut automatiquement être filtré (ex sélectionner que les restaurants dans lesquelles on ne peut manger qu'à l'intérieur).

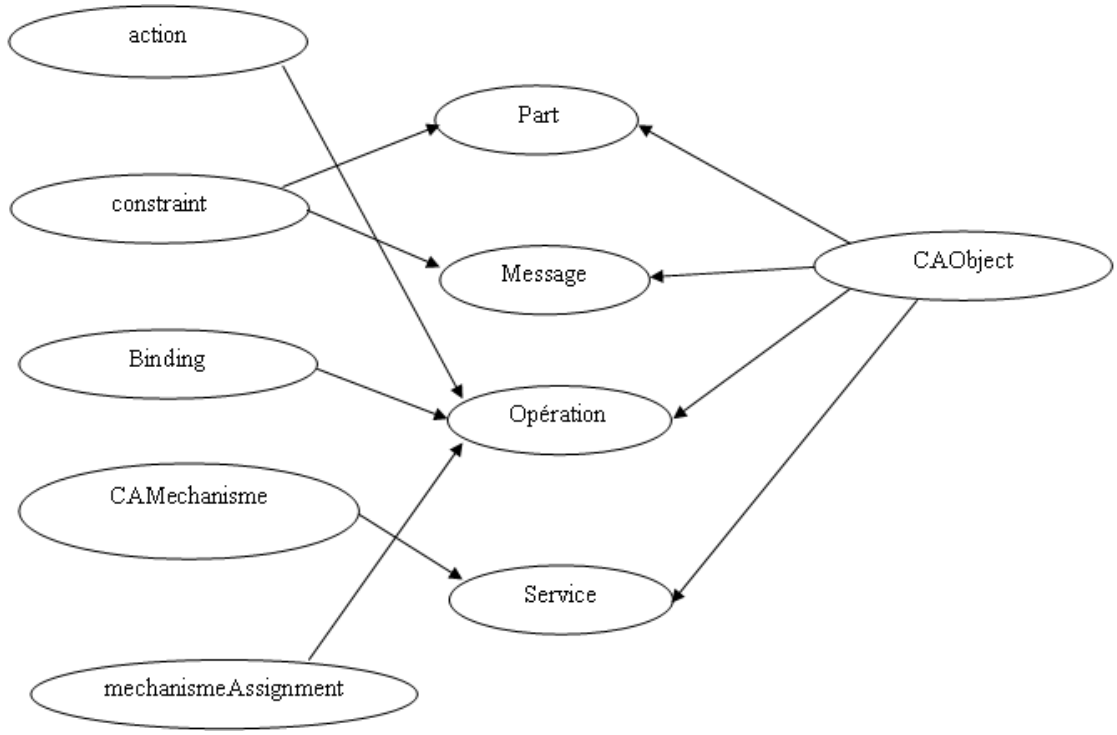

Figure 25 - Transformation de la partie sensibilité au contexte

En résumé, cette partie permet la modélisation des actions d'adaptation selon le contexte, et toutes ces actions sont représentées par la classe CAMechanisme. Cette dernière se compose de deux sous classe : Binging et contextTriggering, nous proposons de transformer ses classes en opérations qui seront fournies par le service CAMechanisme (qui correspond à la classe CAMechanisme).

En ce qui concerne la classe CAObject, elle va être transformée en service, opération, message ou part tout dépend du mechanismeAssignment qui correspond à une opération. Toutes ces transformations sont illustrées par la figure ci-dessus

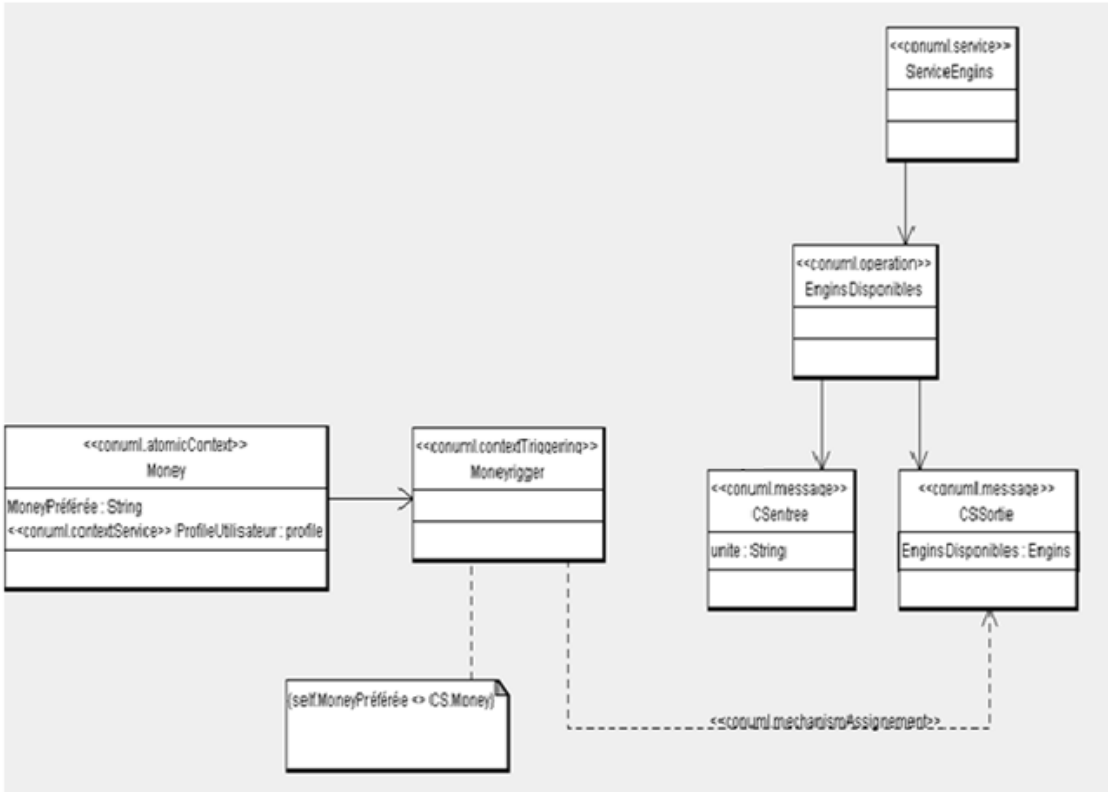

Figure 26- Modélisation de la sensibilité au contexte

La figure 25 illustre la sensibilité du service Liste des engins à l'information contextuelle Money, si cette dernière est différente à la money qui se trouve dans le profile utilisateur le résultat doit être adapté en appliquant une action d'adaptation servant à filtre la liste renvoyé par le service ou bien le message de sortie.

La figure ci-dessous illustre un extrait du résultat de la transformation de la partie sensibilité au contexte, selon la figure l'action d'adaptation au contexte moneyTrigger sera transformée en opération, qui comme message d'entrée l'information contextuelle preferedMoney. Le mechanismeAssignement sera à son tour transformée en opération servant à attribuer l'action d'adaptation à un composant du service, il s'agit dans notre cas au message de sortie de l'opération Liste des engins. Enfin, la contrainte est transformée en paramètre d'entrée de l'opération d'adaptation moneyTrigger.

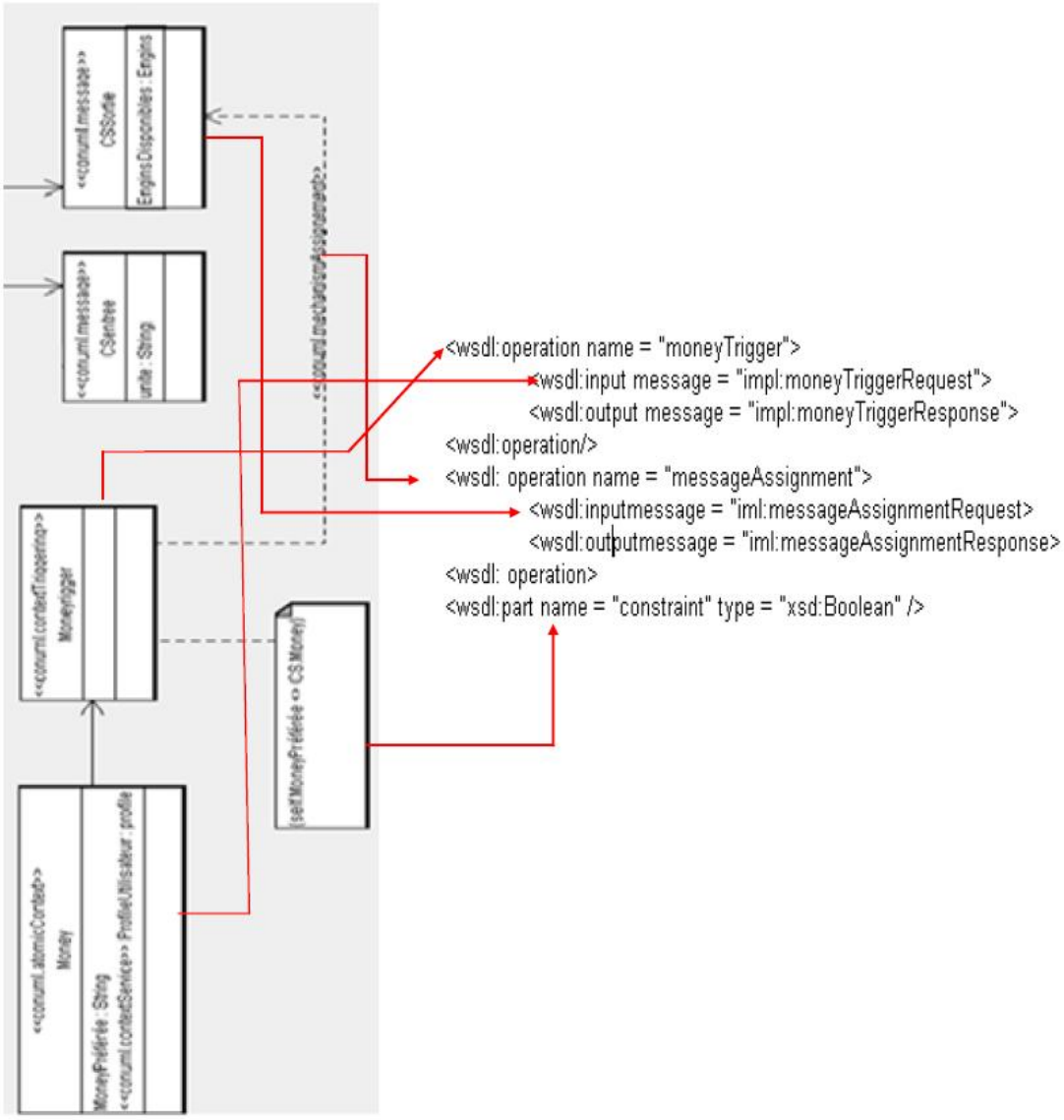

Figure 27 Exemple de la transformation de la partie sensibilité au contexte

En résumé Tous les éléments du méta modèle ContextUML vont correspondre aux éléments du langage WSDL tel qu'il est présenté dans le tableau ci-dessous, et cette correspondance va être réalisée à l'aide du transformateur qu'on va développer.

Eléments du ContextUML	Eléments WSDL
Service	Service
Operation	Opération
Message	Message
Paramètre	Part
Context	Message/Part
AtomicContext	Part
CompositeContext	Message/Part
ContextSource	Service/Opération
ContextService	Service
ContextServiceCommunity	Service
CAMechanisme	Service
Binding	Opération
TrigerringAction	Opération
Constraint	Message/Part
Action	Opération
CAObject	Part/Message/Opération/Service
MechanismAssignement	Opération

Tableau 2 - Correspondance entre les éléments du ContextUML et les éléments du WSDL

Selon le tableau ci-dessus, les éléments en noir sont les composants basiques d'un service web, ils vont correspondre aux éléments service, opération, message et part. En suite, nous montrons les éléments (en rouge) constituant les parties modélisation du contexte et modélisation de la sensibilité en rouge, ils vont correspondre aux éléments WSDL comme suit :

1. ***Context :*** cet élément va correspondre soit un message, soit à un paramètre, ce choix est justifié par le fait que Context représente une information contextuelle, qui va être obtenue en appelant une méthode d'un service de la classe ContextSource, et dans ce cas elle va être soit un part d'entrée pour l'opération de récupération (c'est le cas avec AtomicContext) ou bien un message (c'est le cas du CompositeContext où plusieurs informations contextuelles sont requises).

Context peut aussi être exploité par une opération pour réaliser une adaptation au contexte à travers la classe CAMechanisme et dans ce cas elle peut être un part.

2. ***AtomicContext :*** il va correspondre à un part (c'est un Context).
3. ***CompositeContext :*** il va correspondre à un part ou bien à un message (c'est un Context).
4. ***ContextSource :*** cet élément va correspondre à un service, puisque c'est à lui de récupérer les informations contextuelles en offrant des opérations de collection, d'interprétation et de récupération.
5. ***ContextService :*** il va correspondre à un service (c'est un ContextSource).
6. ***ContextServiceCommunity :*** il va correspondre à un service (c'est un ContextSource).
7. ***ContextAwareMéchanisme :*** cet élément va correspondre à un service, puisque c'est lui qui fait l'adaptation en offrant des d'opérations d'adaptation.
8. ***Binding :*** il va correspondre à une opération, qui sert à effectuer une adaptation selon le contexte, et cela en construisant une liaison entre une information contextuelle et un CAObject.
9. ***TriggerringAction :*** il va correspondre à une opération, qui représente une action d'adaptation selon le contexte.
10. ***Constraint :*** il va correspondre à un message ou bien à un part, puisque les contraintes sont utilisées par des actions (opérations) d'adaptation donc elles vont être soit un message d'entrée d'une action d'adaptation (dans le cas où plusieurs informations contextuelles sont requises), ou bien à un paramètre (cas où une seule information contextuelle est requise).
11. ***Action :*** correspond à une opération servant à effectuer une adaptation selon le contexte.
12. ***CAObject :*** puisque c'est une aggrégation de tous les éléments WSDL, il peut correspondre à un part, un message une opération ou un service.
13. ***MechanismeAssignment :*** il sera transformé en opération servant à attribuer une action d'adaptation ou une information contextuelle à un composant du service qui peut être un paramètre, un message …

IV. Conclusion

Dans ce chapitre, après avoir fait une synthèse des travaux existant sur la sensibilité au contexte, nous avons présenté notre processus basé MDA pour développer des services web sensibles au contexte d'utilisation. En suite, nous avons expliqué le méta modèle ContextUML proposé dans [Sheng05] pour la modélisation des services web sensibles au contexte.

Notre contribution principale est de bénéficier du principal apport de l'approche MDA qui est la transformation de modèle pour le développement des services web sensibles au contexte d'utilisation, Par conséquence, nous avons présenté des travaux qui proposent des transformations vers la plateforme web service et on a essayé de proposer une correspondance

les éléments permettant de modéliser la partie contexte et adaptation au contexte avec les éléments WSDL.

Dans le chapitre suivant nous allons présenter notre cas d'étude, et on va expliquer les étapes d'implémentation des services sensible au contexte ainsi que notre transformateur sensible au contexte.

Chapitre 4

Implémentation et cas d'étude

I. Introduction

Dans l'industrie logicielle, l'adaptation des produits software n'est pas nouvelle, mais plutôt elle représente une exigence de tous les clients qui utilisent ces produits. Généralement cette adaptation est réalisée en offrant des possibilités de paramétrage des applications par les utilisateurs ou bien en définissant des privilèges par les administrateurs**.**Dans tous les cas, l'adaptation est implémentée par des testes si-sinon. Cela s'applique quelque soit le domaine cible.

Le travail présenté dans ce mémoire vise à réaliser l'adaptation des services web aux différentes situations en automatisant le cycle de développement des services et en exploitant toutes les conditions d'exécution et les informations sur l'environnement dans lequel nos services s'exécutent. Cela assure une meilleure pertinence des résultats, améliore l'interaction utilisateur application et facilite considérablement le développement, la maintenance et l'intégration des nouveaux services.

Par automatisation du cycle de développement on parle de l'utilisation des outils tout au long de la production des services, ces outils incluent ceux utilisés lors de la modélisation tels que les outils permettant de définir des diagrammes UML (outils de modélisation), ainsi que ceux utilisés pour la transformation des modèles (outils de transformation).

En ce qui concerne le deuxième point, c'est-à-dire la prise en compte des informations qui entourent nos services lors de leur exécution, on parle du contexte et cela entre dans le cadre de l'informatique ubiquitaire qui cherche à satisfaire les utilisateurs des produits software à tout moment, où ils se trouvent et avec tous les dispositifs qu'ils ont.

Dans ce chapitre nous allons essayer de présenter notre application orientée services sensibles au contexte, et nous allons expliquer la façon d'utiliser l'approche MDA pour le développement de cette application.

II. Cas d'étude

Pour montrer notre approche nous avons choisi de développer une application de gestion de transport et de livraison sur le web. Cette dernière se base sur l'utilisation de services web.

Dans le domaine du transport et de livraison, une entreprise est divisée en unités qui se propagent au territoire national. Chacune de ces unités possède un certain nombre d'engins, de remorques, de conducteurs et d'autres personnels, et offre un certain nombre de services.

Les services offerts par une unité quelconque, dépendent de la zone où elle se trouve, et des engins qu'elle possède. Par exemple une unité qui se trouve dans le sud de notre pays offre des services différents de ceux d'une unité du nord du pays.

Une telle entreprise nécessite une application lui permettant de suivre les différents processus métiers quelque soit l'unité consommatrices. Cette application doit s'adapter aux différentes situations contextuelles telles que : les profiles des utilisateurs, la localisation, leurs préférences, le climat, …etc.

Les utilisateurs de notre application, peuvent être les employés de l'entreprise, des partenaires ou bien des clients. Dans tous les cas, l'application s'exécutera dans des conditions différentes et donc, l'adaptation est nécessaire.

III. Context aware Transpotation

CATransportation est une application orienté service qui permet la gestion du transport et de livraison sur le web. Elle utilise des services web sensibles au contexte pour offrir aux différents utilisateurs (employées, clients, partenaires) des résultats pertinents selon la situation contextuelle. La figure ci-dessous présente d'une façon globale certains services utilisés par CATransportation.

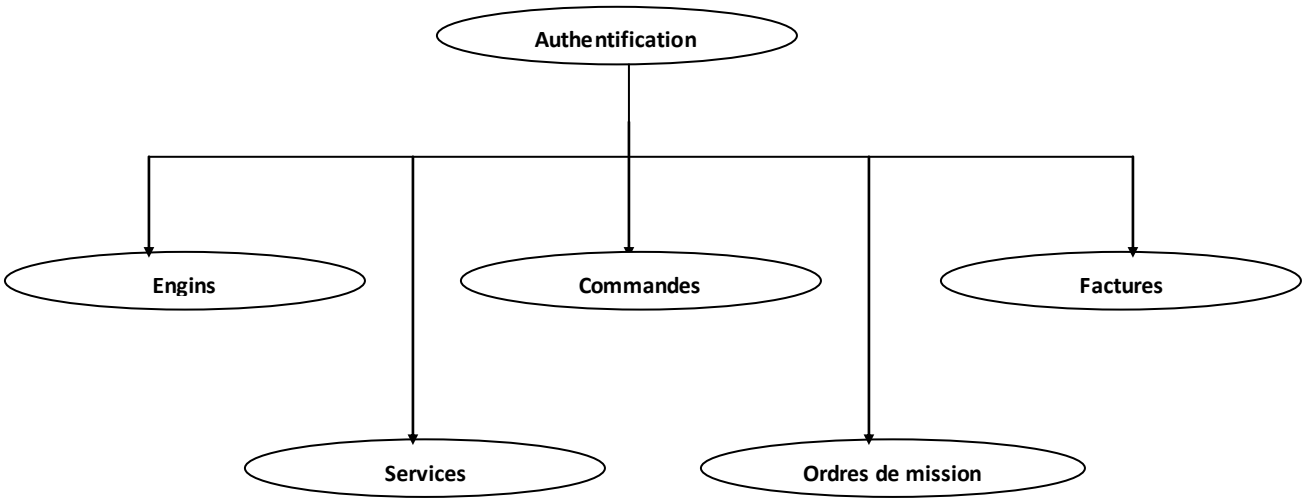

Figure 28 – Services de l'application CATransportation

Les services présentés dans la figure 27 constituent une partie de notre application CATransportation, ils vont être invoqués pour servir plusieurs types d'utilisateurs (employés, clients, et partenaires), et ils seront exécutés dans des conditions différentes. Chacun de ces services offre des opérations de suivi,

Ces services sont comme suit :

- ***Engins :*** permet d'avoir des informations sur les engins de l'entreprise et de connaître à tout moment leurs statuts.
- ***Services :*** permet d'avoir des informations sur les services offerts par l'entreprise.
- ***Commandes :*** permet d'avoir des informations sur les commandes.
- ***Ordres de mission :*** permet d'avoir des informations sur les ordres de mission.
- ***Factures :*** permet d'avoir des informations sur les factures.

D'autres services ont été développés tels que : Conducteurs, Clients, Tarifs, Proformas…

IV. Processus de développement

Dans le chapitre précédent nous avons présenté le processus de développement qu'on a adopté, il se compose de trois étapes : Modélisation du PIM, choix d'une plate forme cible, et Transformation. Dans ce qui suit on va présenter chacune de ces étapes :

1. Modélisation

La modélisation constitue l'étape principale du processus de développement, elle influence directement sur le succès ou l'échec de l'application, c'est le cœur de l'approche MDA. Les services d'une application vont être modélisés en utilisant des diagrammes UML, ces derniers se caractérisent par leur niveau d'abstraction élevé, ce qui sert à la compréhension du fonctionnement des services. Ainsi, ils représentent un moyen de communication efficace.

De nombreux outils de modélisation des diagrammes sont disponibles, dont certains permettent d'exporter les diagrammes sous formes de fichiers XMI. Nous avons utilisé l'outil ArgoUML, pour modéliser les services de notre application.

A. Argo UML

ArgoUML est un logiciel de création de diagrammes UML sous licence libre et programmé en Java (et donc multi-systèmes). Ainsi ArgoUML permet :

- La création des diagrammes UML de manière simple et graphique.
- L'exportation de ces diagrammes dans de nombreux formats (PS, SVG, XMI ...).
- La génération de classes Java (et même C/C++ et Php avec des plugins) des objets décris avec ArgoUML.
- L'analyse de classe Java déjà existante (très bonne pratique).

Nous avons utilisé la version 0.26 téléchargeable à partir du lien suivant : http://argouml.tigris.org. La figure 28 montre la fenêtre principale de l'éditeur ArgoUML

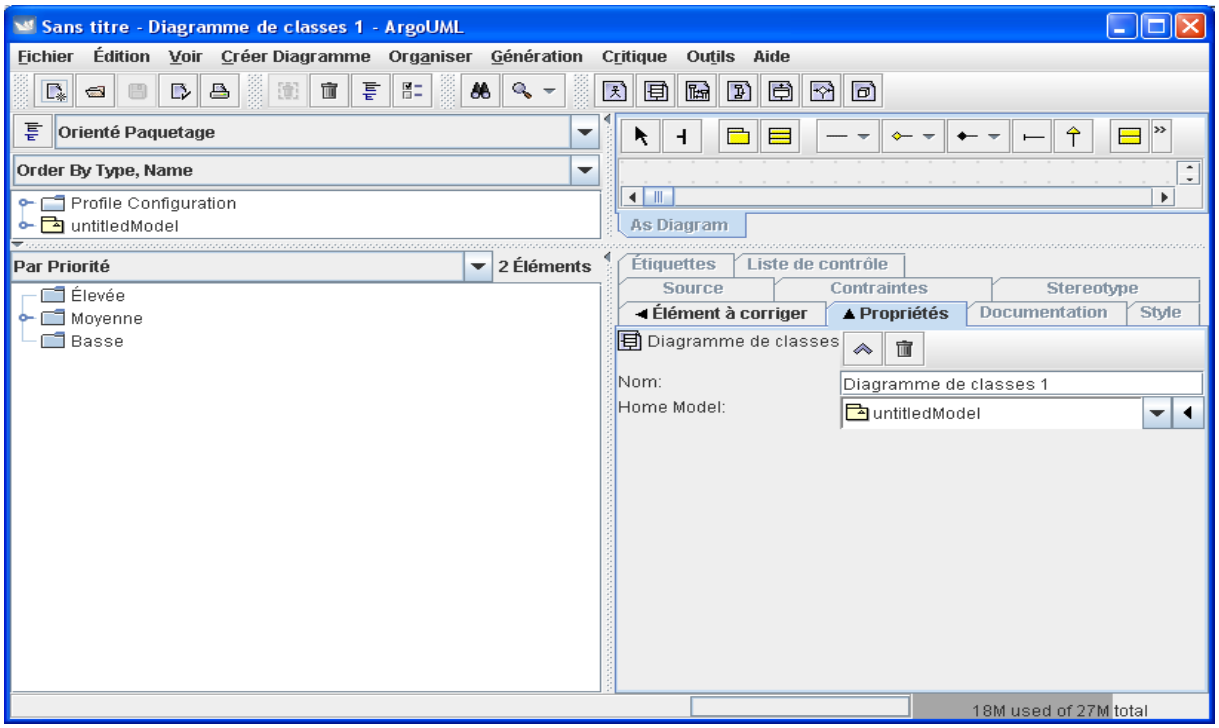

Figure 29 - L'outil ArgoUML

B. ContextUML

Tous les modèles obtenus de la modélisation des services doivent correspondre au méta modèle ContextUML [Sheng05] présenté dans le chapitre 4.

L'outil ArgoUML permet d'importer des représentations XMI. La figure 29 le méta modèle ContextUML dans l'outil ArgoUML.

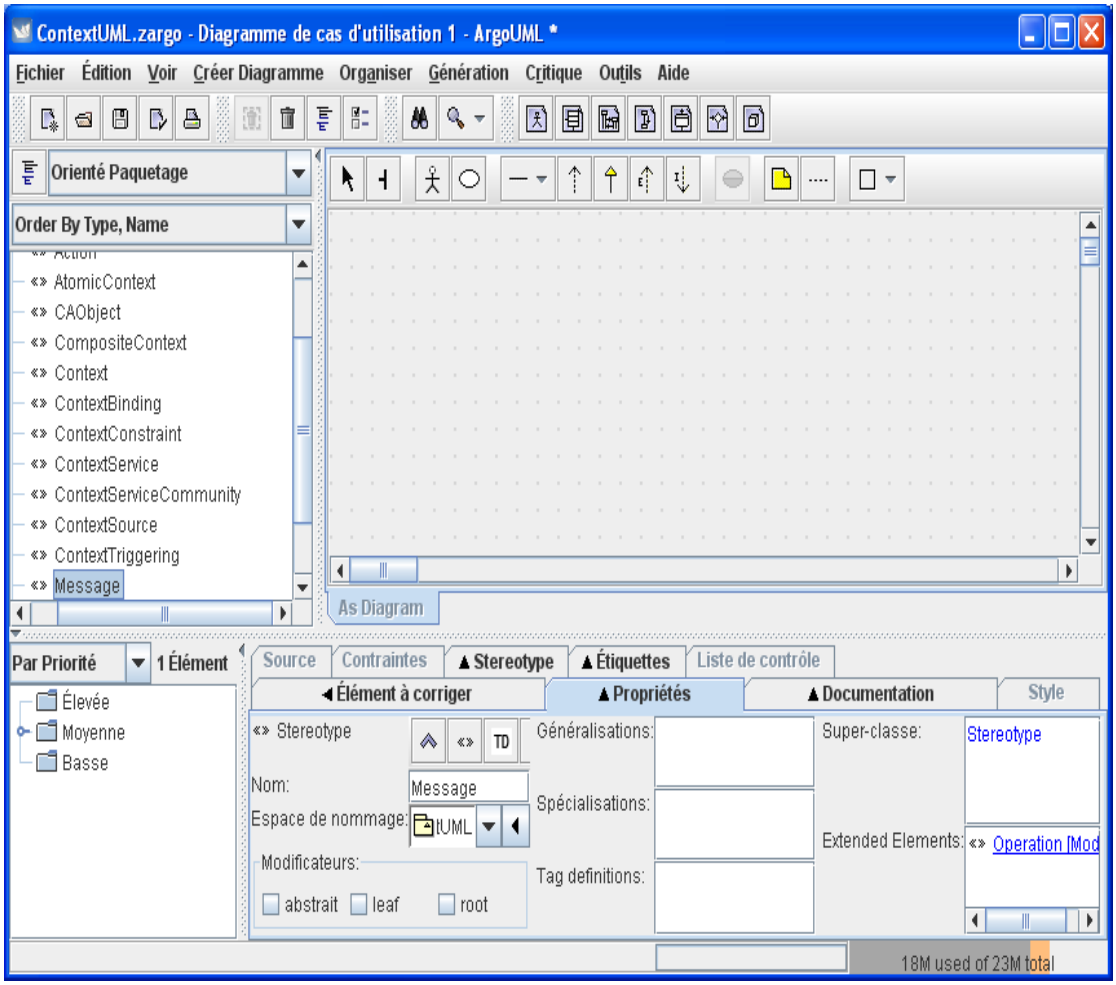

Figure 30 – ContextUML

La modélisation des services sensibles au contexte se fait en utilisant l'outil ArgoUML, ce dernier permet de construire des diagrammes UML qui peuvent être exportés sous formes de fichier XMI. La figure 30 présente la modélisation du service CAEnginsService.

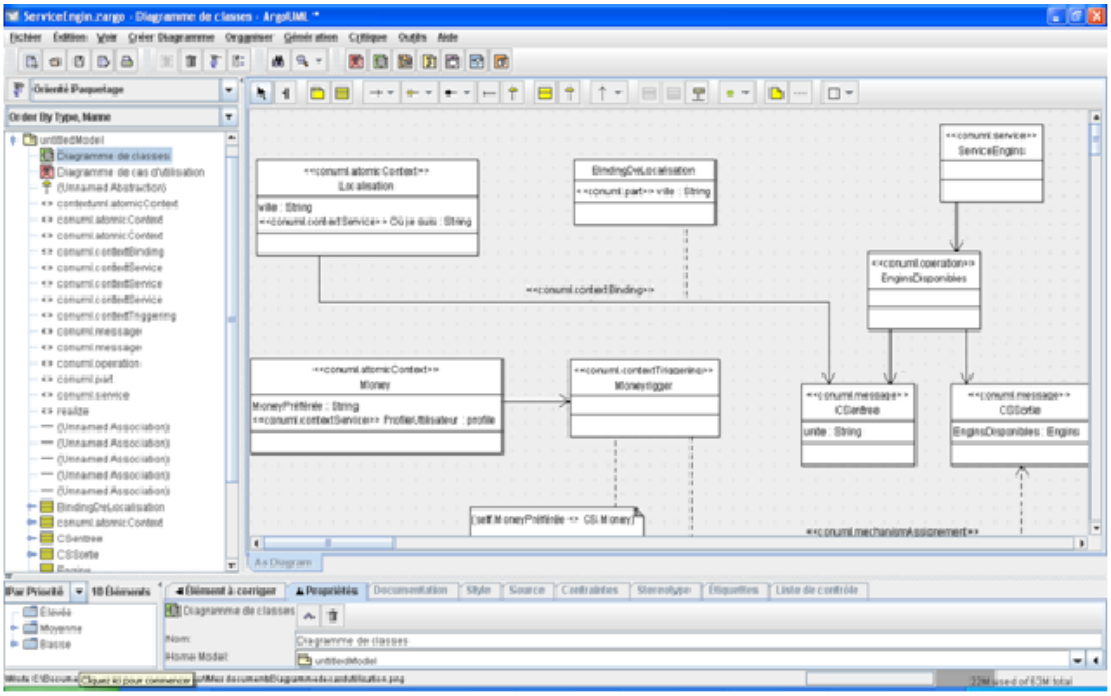

Figure 31 - Modélisation d'un service sensible au contexte

Le service présenté dans la figure ci-dessus se nomme ListeEngins, il contient les opérations suivantes :

- Engins disponibles : cherche les engins disponibles.
- Engins en mission : cherche les engins en mission.
- Engins en panne : cherche les engins en panne.

Chacune de ces opérations accepte comme entrée le nom de l'unité et génère comme sortie une liste d'engins. Les employés de l'entreprise peuvent invoquer ce service pour connaître le statut de chaque engin. Il est possible aussi que le service ListeEngins soit appelé par un client pour effectuer des commandes, et selon quelques informations contextuelles (telles que la localisation, le climat…), le résultat sera adapté et il va être réduit à la liste des engins de l'unité la plus proche au client. Le mécanisme de mapping permet la substitution de l'unité (entrée) par la localisation de l'utilisateur.

Après l'étape de modélisation, le modèle obtenu doit être exporté sous le format XMI vers l'outil de transformation CATransformer, qui va s'occuper de la transformation des modèles obtenu en d'autres formats tels que WSDL, BPEL, …

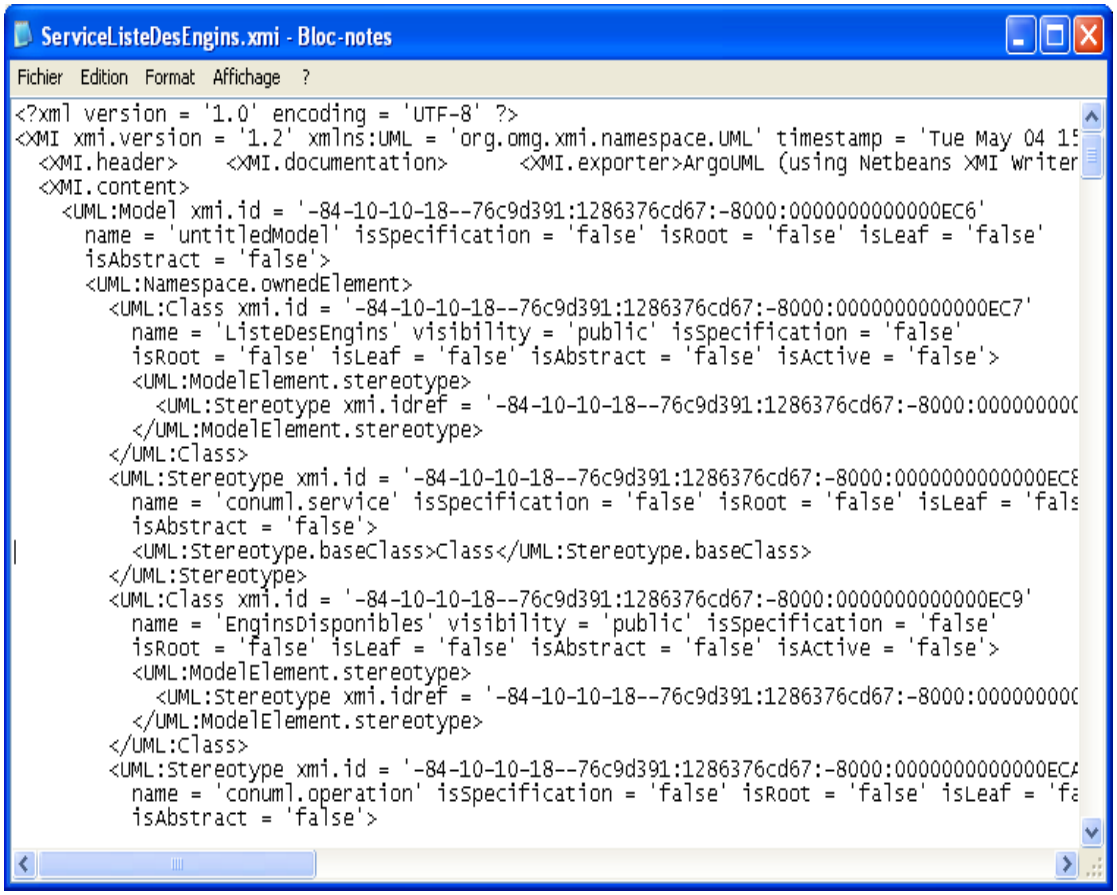

Figure 32 - Format XMI

2. Transformation

La transformation est l'une des atouts importants de l'approche MDA. Cependant, dans la plupart des travaux qui proposent cette approche cette phase est négligée. En revanche, sur le plan pratique, les outils de transformation sont nombreux, et ils permettent de faire divers types de transformation. UMT-QVT (UML Model Transformation Tool) [UMT], MTL(Model Transformation engine), ATL (Atlas Transformation Language), GMT (Generative Model Transformer) [GMT] , sont des exemples d'outils de transformation.

Le problème qui se pose est que ces outils de transformation sont spécifiques à des plateformes particulières et ne prennent pas en considération les informations contextuelles. Pour cela, nous avons développé un transformateur qui fait la correspondance entre les modèles obtenus lors de la première phase et WSDL pour la description des interfaces, et BPEL pour les descriptions des comportements.

Notre transformateur est appelé CATransformer, il est développé en java, et les transformations sont basés sur XSLT, tel qu'il est montré dans le chapitre précédent.

Le processus de transformation qu'on a utilisé est présenté dans la figure ci-dessous :

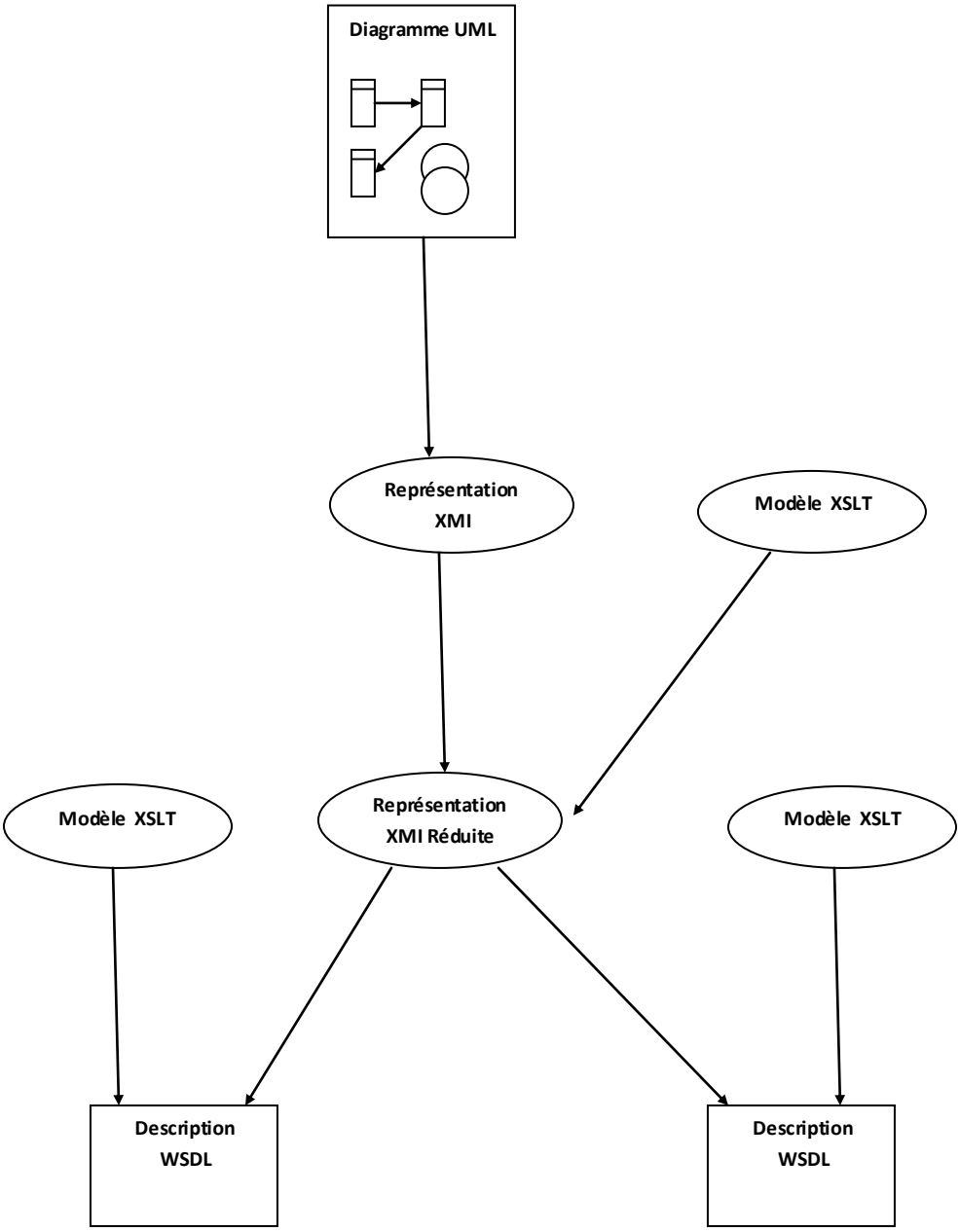

Figure 33 - Processus de transformation

Comme montré dans la figure ci-dessus, la transformation se fait en important des représentations XMI des diagrammes UML, ces dernières vont être transformées dans une première phase, à l'aide d'un modèle XSLT pour obtenir des représentations réduites. Pendant cette première phase, nous cherchons à éliminer les informations spécifiques à l'outil de modélisation. L'importation est faite comme montré dans les figures ci-dessous :

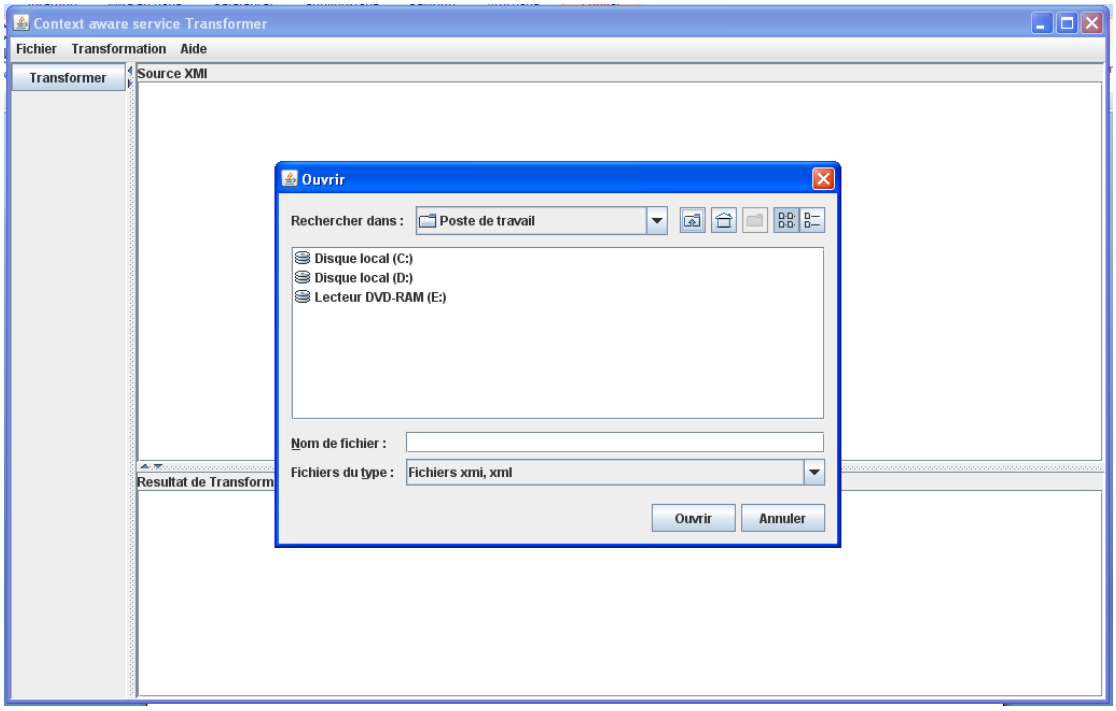

Figure 34 - Importation d'une représentation XMI

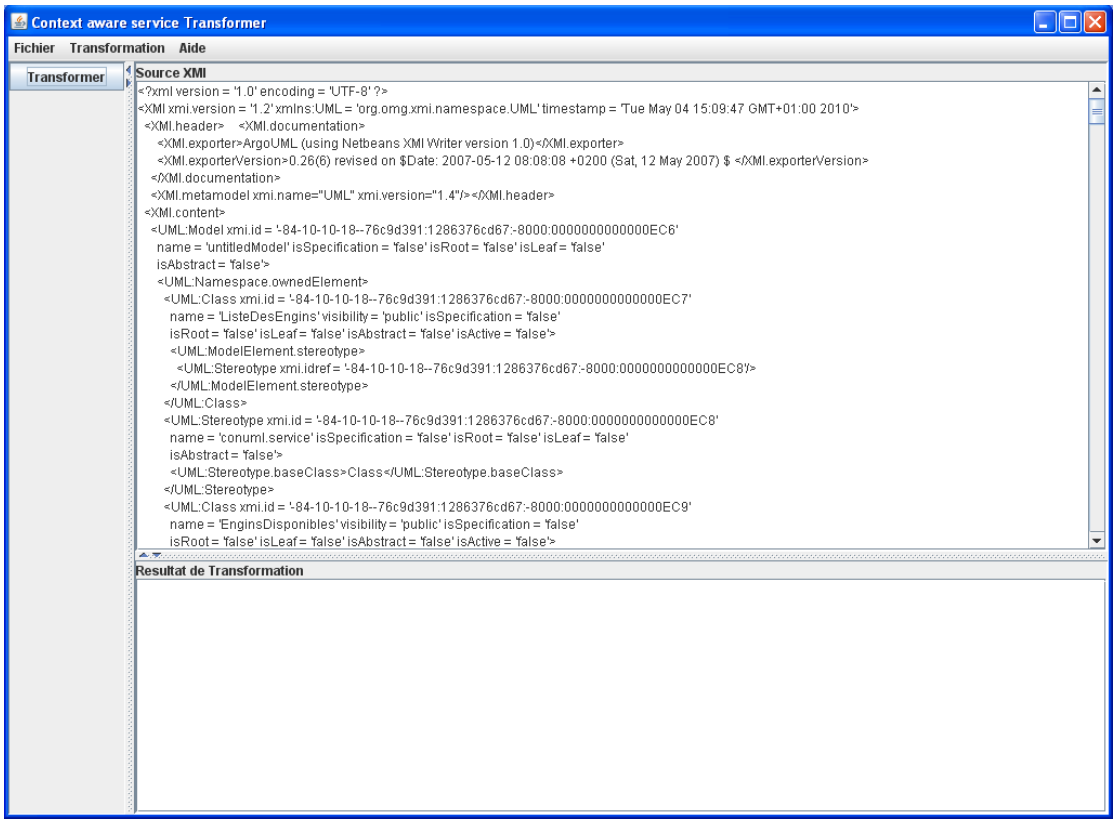

Figure 35 - Représentation XMI d'un diagramme UML

Après avoir importé une représentation XMI, il faut choisir la transformation demandée en choisissant le menu Transformation, tel qu'il est montré dans la figure ci-dessous :

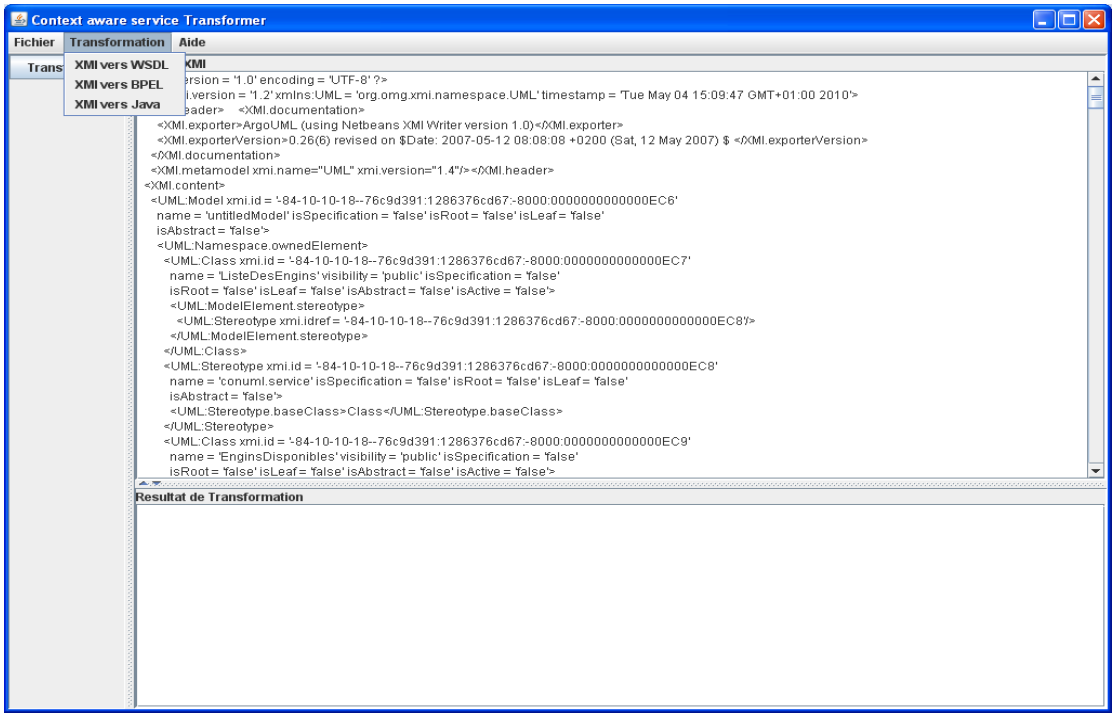

Figure 36 – Transformation avec CATransformer

Pour exécuter la transformation, il suffit de cliquer sur le bouton Transformer, et le résultat va être généré. Le transformateur applique le processus présenté ci-dessus, la figure 36 illustre une fichier résultant d'une transformation :

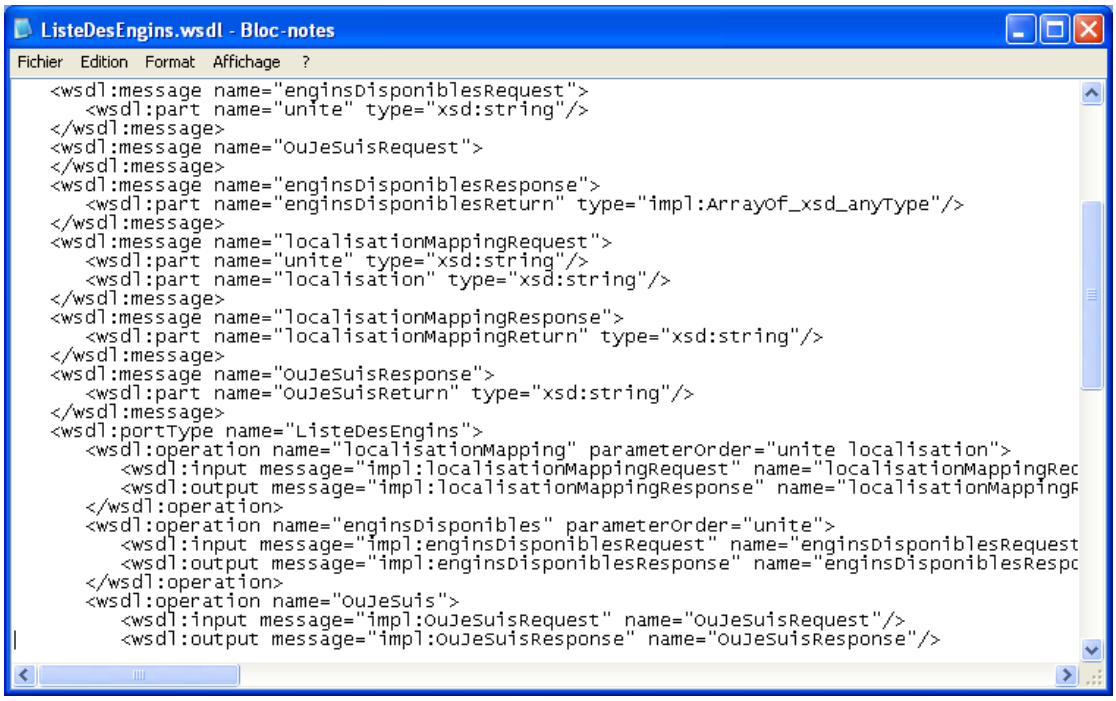

Figure 37 - Résultat de la transformation

V. Présentation de l'application CATransportation

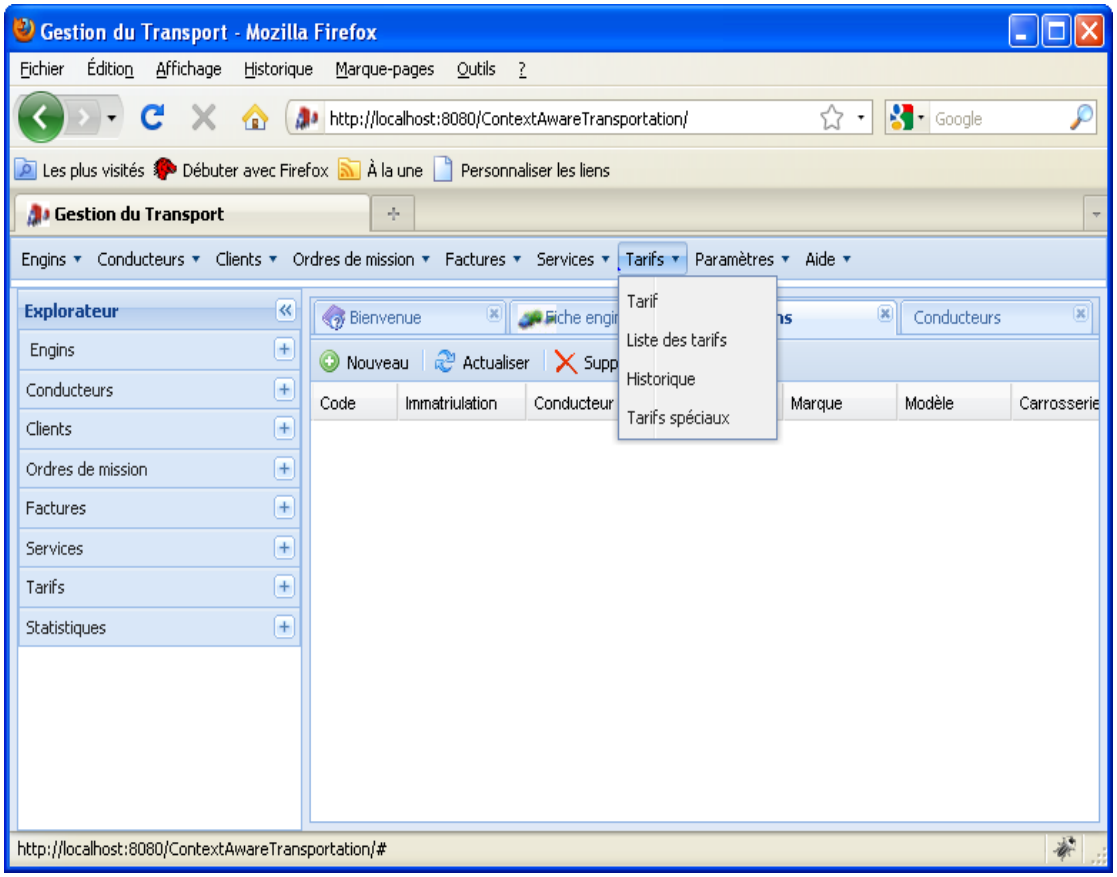

Figure 38 - CATransportation

CATransportation est une application de gestion du transport et de livraisons sur le web, elle est développée en utilisant des outils de développement des applications web tels que : MyEclipse, Ext-js, MySQL.

I. myEclipse

MyEclipse est un outil de développement web développé par Genuitec. Il dispose d'assistants personnalisés pour la création et le développement de ressources Web complètement intégrés à l'éditeur Java d'Eclipse :

- Editeurs HTML et de JSP permettant un affichage en mode preview des pages Web.
- Editeurs de ressources XML/XSL.
- Assistants de création de modules Web (servlets/JSP).
- Assistants de création et de synchronisation des descripteurs de déploiement **web.xml.**
- Outils de déploiement, de test et de débogage de ressources Web, avec :
    1. Déploiement direct de l'environnement MyEclipse vers l'environnement serveur choisi (plus de vingt serveurs supportés, dont JBoss, Apache Tomcat et WebSphere/WebLogic).
    2. Contrôle du serveur Web ou d'applications (arrêt/relance).

3. Débogage à chaud de ressources JSP et Java.
4. Support évolué de certains frameworks standards, incontournables dans un développement Web, comme Struts/JSF ou Hibernate.

MyEclipse fournit en outre un éditeur de formatage de code incluant un correcteur syntaxique fournissant en temps réel le résultat de l'analyse syntaxique de la ressource Web développée. La figure 38 montre la fenêtre principale de MyEclipse.

Nous avons utilisé la version MyEclipse 6.5, téléchargeable à partir du lien suivant : http://www.myeclipseide.com/

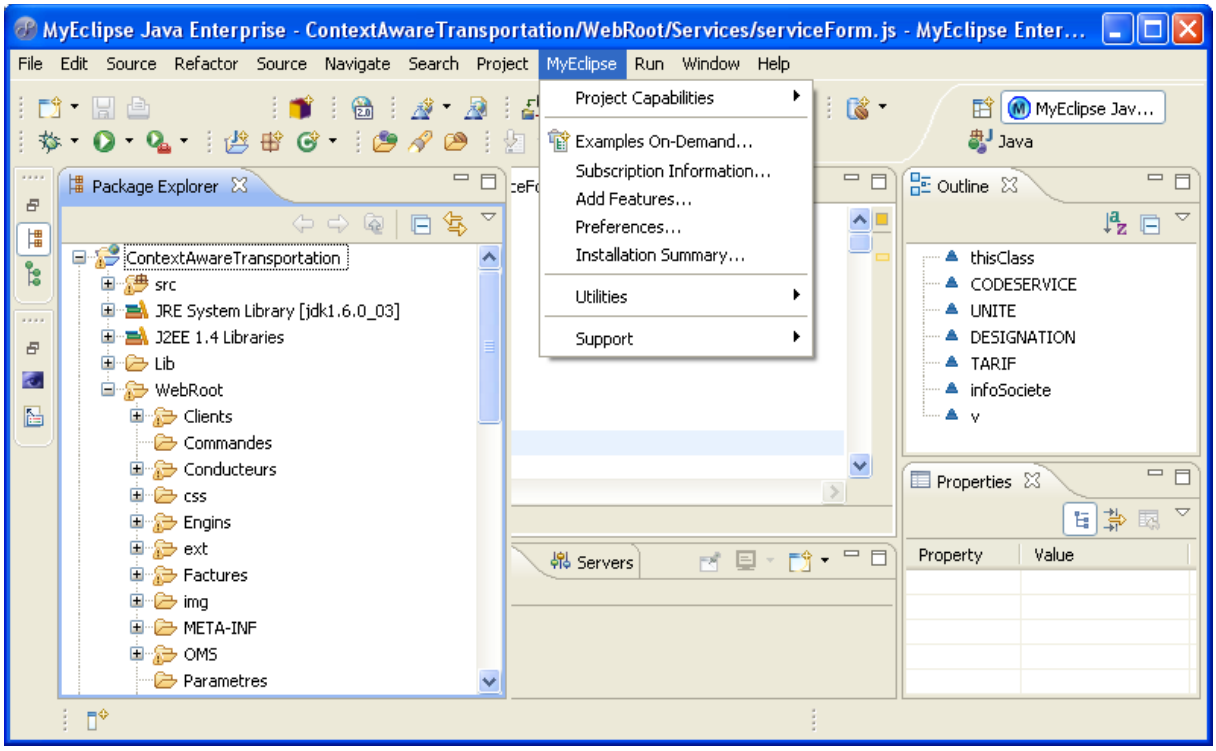

Figure 39 - L'outil MyEclipse

II. Ext js

Ext js est une bibliothèque java script qui permet de développer des applications web, elle contient des librairies java script très utiles dans le développement des interfaces. Dans cette thèse nous avons utilisé la version Ext js 2.0, elle est téléchargeable à partir du lien suivant :

III. Exemple de service : ListeDesEngins

L'exemple du service ListeDesEngins présenté dans le chapitre précédent est montré par la fenêtre ci-dessous. Ce service se déroule dans la situation contextuelle suivante : utilisateur = employé, localisation = Annaba.

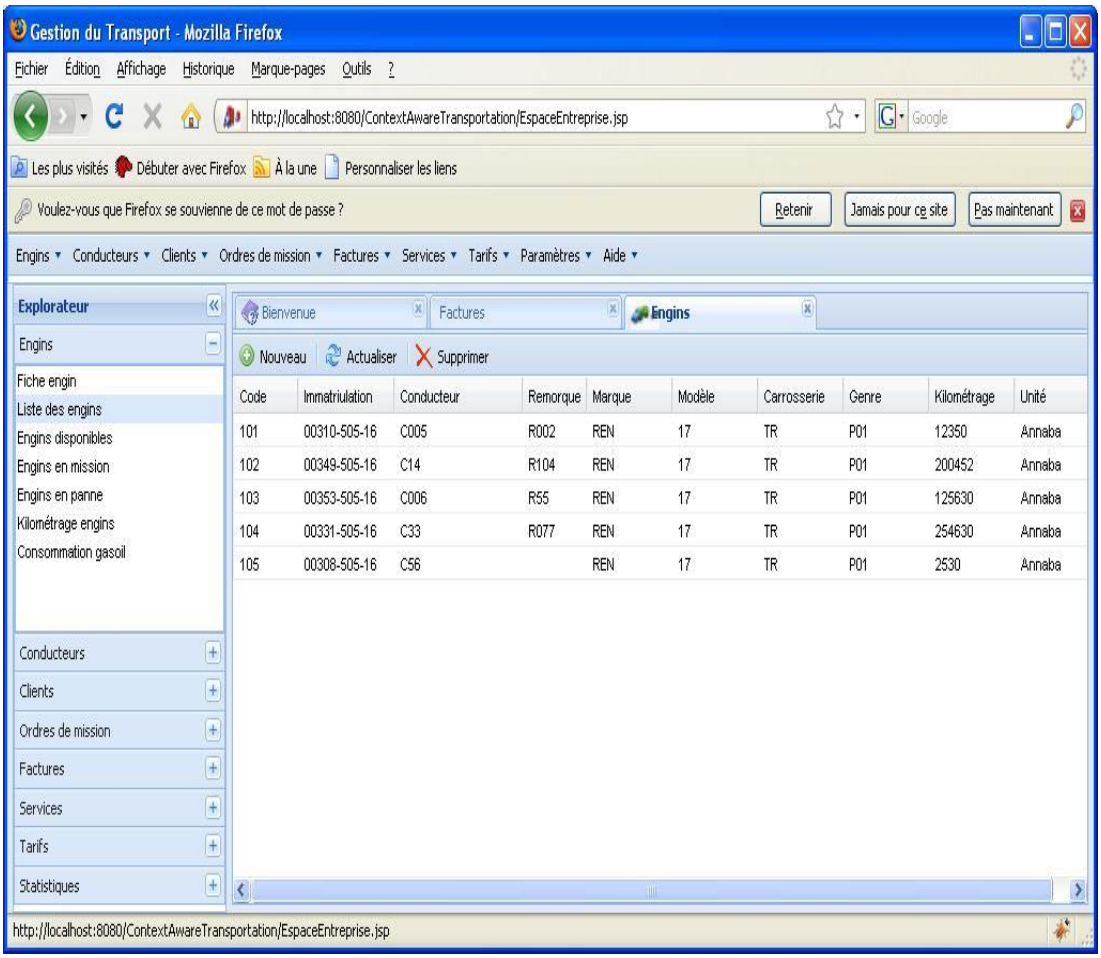

Figure 40 - Service ListeDesEngins (situation contextuelle 1)

Dans une autre situation contextuelle, où : l'utilisateur = client, localisation = Annaba, le résultat de l'appel du service ListeDesEngins se diffèrera.

Selon les deux figures, on constate que le résultat du service se diffère selon le contexte, par exemple dans la situation contextuelle 1 le service récupère les engins de l'unité Annaba, avec toutes les informations nécessaires. En revanche, dans la situation actuelle 2, le service n'affiche que les informations utiles pour le client telles que le type et le prix, et masque d'autres informations jugées inutiles.

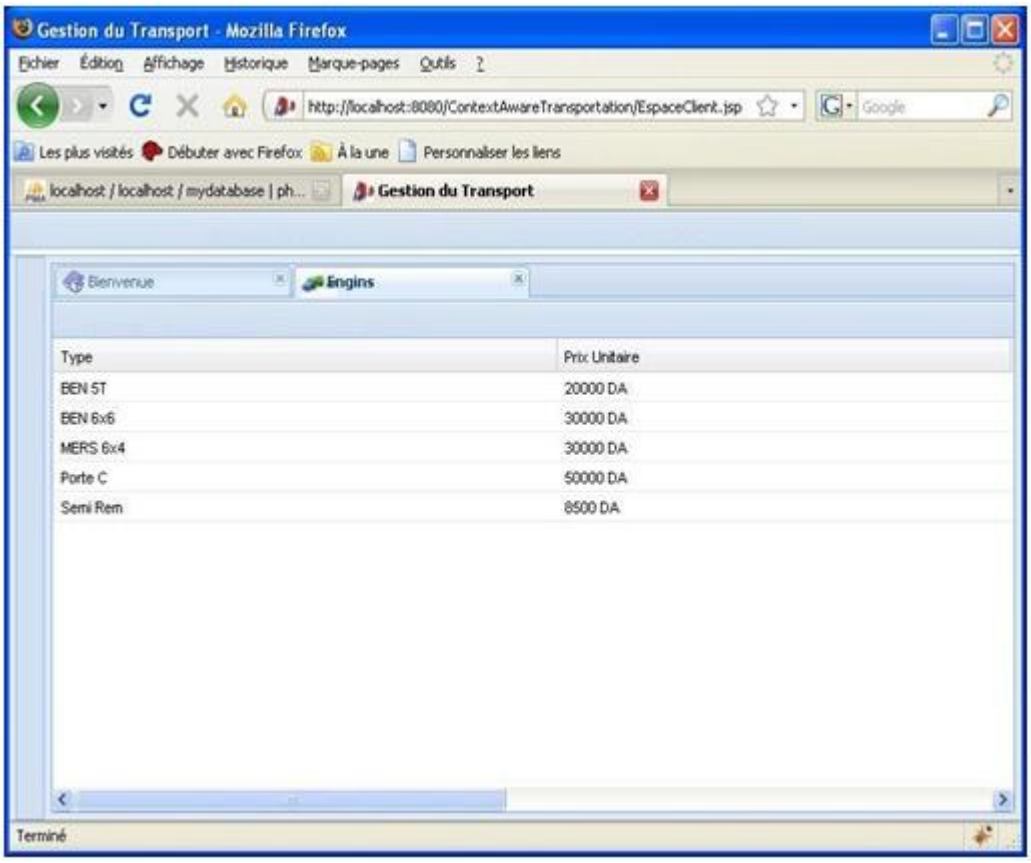

Figure 41 - Service ListeDesEngins (situation contextuelle 2)

VI. Conclusion

Dans ce chapitre, nous avons présenté le processus de développement des services web sensibles au contexte d'utilisation. Notre processus se commence par l'étape de modélisation où on modélise les services ainsi que le contexte d'utilisation. Puis les modèles obtenus vont être transformés en utilisant le transformateur qu'on a développé.

Afin de valider notre proposition, nous avons développé une application orientée service sensible au contexte, servant à gérer le transport et la livraison sur le web. Nous avons présenté un exemple des services de cette application qui le service Liste Des Engins permettant de faire le suivi des engins d'une entreprise, et nous avons expliqué la sensibilité de ce service au contexte d'utilisation.

Conclusion & perspectives

I. Conclusion

Dns ce mémoire, nous avons essayé d'adopter l'approche MDA pour la prise compte du contexte dans des architectures orientées service. Cette approche a plusieurs avantages dans tous le cycle de vie des applications et cela sur tous les plans : qualité, durée et coût. La qualité est assurée grâce à l'utilisation des modèles de haut niveaux d'abstraction, et le gain du temps est du aux techniques de transformation. Ainsi les coûts seront logiquement réduits.

La prise en compte du contexte dans la modélisation améliore d'une façon significative l'interaction entre utilisateur et les applications et augmente la pertinence des résultats fournis.

Dans ce travail, nous avons adopté ContextUML qui un méta modèle permettant de modéliser des services web sensibles au contexte d'utilisation. En plus, nous avons développé un transformateur capable de prendre en compte les informations contextuelles pour faire des transformations diverses.

Pour mieux illustrer notre travail, nous avons présenté un cas d'étude qui est une application de gestion de transport et des livraisons sur le web. Cette dernière est une application orientée service qui s'exécute dans diverses situations contextuelles pour satisfaire ses utilisateurs.

I. Perspectives

Ce travail de thèse nous a permis de tracer plusieurs perspectives de recherche :

- Nous allons essayer dans le future d'adapter le méta modèle qu'on a adopté pour développer des services composés, et par conséquent il faut définir le passage vers BPEL (servant à décrire le comportement des services)

- Il serait intéressant de proposer des transformations (avec des techniques de reverse – engineering) qui permettent de revenir d'une façon automatique, ou semi-automatique, aux diagrammes UML à partir de la description des interfaces des services (et même d'une description du comportement), ceci permet d'adapter des services existants

- Pour pouvoir générer les interfaces utilisateur, nous pensons qu'il est préférable de se baser sur l'approche MDA, c'est-à-dire en modélisant les interfaces des services et en développons des transformations vers les dispositifs cibles.

Bibliographie

[Abowd97]Abowd G. D, Atkeson C. G., Hong et al, *Cyberguide : A mobile contextaware tour guide*. ACM Wireless Networks, 1997, Vol. 5, N°3, pp. 421-433

[Abowd99] Abowd G. D, Dey A. K, Brown P. J, Davies. N, Smith M, and Steggles P, 1999. *Towards a Better Understanding of Context and Context- Awareness*. In Proceedings of the 1st international Symposium on Handheld and Ubiquitous Computing, September 27 - 29, 1999 Karlsruhe, Germany. H. Gellersen, Ed. (Lecture Notes In Computer Science, Vol. 1707, pp. 304- 307).

[Agoston00] Agoston T, Ueda T, and Nisimura Y, Pervasive computing in a networked world, In Proc. of INET 2000, 18-21 july 2000, Japan,

[Anand05] Anand S, Padmanabhuni S, et Ganesh J, (2005), Perspectives on service-oriented architecture. In *ICWS*. IEEE Computer Society.

[Andrews03] Andrews T, Curbera F, Dholakia H, Goland Y, Klein J, Leymann F, Liu K, Roller D, Smith D, Thatte S, Trickovic I, and Weerawarana S, Business Process Execution Language for Web Services (BPEL4WS), Version 1.1, may 2003.

[Assaf02a] Assaf A, Business Process Modeling Language, 13 Novembre 2002. http ://www.bpmi.org/bpml-spec.esp

[Assaf02b] Assaf Arkin, Askary S, Fordin S, Jekeli W, Kawaguchi K, Orchard D, Pogliani S, Riemer K, Struble S, Takacsi-Nagy P, Trickovic I, and Zimek S, Web Service Choreography Interface 1.0, 2002.

[ATL]:http://eclipse.org/downloads/index.php, http://download.eclipse.org/tools/emf/scripts/downloads.php

[Bederson95] Bederson B. B, *Audio Augmented Reality: A Prototype Automated Tour Guide*. In Proceedings of Human Factors in Computing Systems (CHI 95) New York : ACM Press, 1995, pp. 210-211

[Beigl00] Beigl M, MemoClip: A Location-Based Remembrance Appliance. Personal and Ubiquitous Computing, 4(4):230–233, 2000.

[Belaunde03] Belaunde M, *L'approche MDA et les expérimentations à France Télécom*, [en ligne], 26 juin 2003, Disponible sur :

http://aristote1.aristote.asso.fr/Presentations/CEA-EDF-

2003/Conferences/MarianoBelaunde/ExperimentationsMDA-EcoleDEte-26Juin2003.ppt

[Belaunde04] Belaunde M, *Le MDA* [en ligne], 9 janvier 2004, Disponible sur :

http://www.bretagne.enscachan.fr/DIT/People/Claude.Jard/sem_20_04_2004_FTRD_trans9.pdf

[Bennett94] Bennett F, Richardson T, and Harter A, *Teleporting - Making Applications obile*. Proc. In IEEE Workshop on Mobile Computing Systems and Applications, December 1994, Santa Cruz, California, pp. 82-84

[Bezinvin02a] Bezinvin J. et Blanc X, *MDA vers un nouveau paradigme(1)*, Développeur Référence, [en ligne], 15 juillet 2002, v2.16, pp. 7-11, Disponible sur : http://www.weblmi.com/devreference

[Bezivin02b] Bezivin J, *Les nouveaux défis des systèmes complexes et la réponse MDA de l'OMG*, [en ligne], 2002, Disponible sur :

http://www.lifl.fr/jfiadsma2002/talks/jfiadsma2002-Bezivin.pdf

[Bezivin03a] Bizivin J, *Ecole d'Eté d'Informatique CEA EDF INRIA 2003*, Ingénierie des modèles logiciels [en ligne], 2003, Disponible sur :

http://aristote1.aristote.asso.fr/Presentations/CEA-EDF-2003/Cours/JeanBezivin/IndexJeanBezivin.html

[Bezivin 03b] Bezivin. J, Dupe. G, Jouault. F, Pitette. G, et Rougui. J. E, First Experiments with the ATL Model Transformation Language: Transforming XSLT into XQuery. 2nd OOPSLA Workhop on Generative Techniques in the context of Model Driven Architecture, October 2003.

[Birnbaum97] Birnbaum J, 1997. Pervasive information systems. *Commun. ACM* 40, 2 (Feb. 1997), 40-41. DOI= http://doi.acm.org/10.1145/253671.253695 (consulté le 02.07.2007)

[Blanc04] Blanc X, *Ingénierie des modèles*, [en ligne], juillet 2004, Disponible sur :

http://www-src.lip6.fr/homepages/Xavier.Blanc/courses/XB-Model-Engineering.ppt

[Bordbar04a] Bordbar B, et Staikopoulos A, Modeling and Transforming the Behavioural Aspects of Web Services. WiSME@UML 2004, October 2004.

[Bordbar04b] Bordbar B, et Staikopoulos A, On Behavioural Model Transformation inWeb Services. Proceeding of the 23rd International Conference on Conceptual Modeling (ER2004) and eCOMO 2004, 3289:667–678, November 2004.

[Brotherton99] Brotherton J, Abowd G. D, & Truong, K. *Supporting capture and access interfaces for informal and opportunistic meetings*. In Atlanta, GA : Georgia Institute of Technology, GVU Center, (GIT-GVU-99-06), January 1999.

[Brown96] Brown P. J, The Stick-e Document: A Framework for Creating Context-Aware Applications. In Electronic Publishing, pages 259– 272, Laxenburg, Austria, September 1996.

[Burrell02] Burrell J, and Gay G, (2002), 'E-graffiti: evaluating real-world use of a context-aware system', *Interacting with Computers – Special Issue on Universal Usability*, Vol. 14, No. 4, pp.301–312.

[Carl04] CARL-OLAV. MDA as Framework for Supporting Enterprise Standards. Practitioner's Reports - ECOOP2004, June 2004.

[Ceri07] Ceri S, Florian D, and Matera M, Facca F. M, 2007. Model Driven Development of Contextaware Web Applications, ACM Transactions in Internet Technology(TOIT), Volume 7, Issue 1.

[Chen00] Chen G, and David Kotz D, A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Department of Computer Science, Darmouth College, Hanover, New Hampshire, USA, November 2000.

[Chen03] Chen H, Finin T, and Joshi A, An Ontology for Context-Aware Pervasive Computing Environments. Knowledge Engineering Review, 18(3):197–207, 2003.

[Cheverst 99] Cheverst K, Mitchell K, and Davies N, Design of an Object Model for a Context Sensitive Tourist GUIDE. Computers and Graphics, 23(6):883–891, 1999.

[Czarnecki03] Czarnecki K, et Helsen S, Classification of Model Transformation Approaches. proceedings of the 2nd OOPSLA'03 Workshop on Generative Techniques in the Context of MDA, October 2003

[Davies98] Davies N, Mitchell K, Cheverst et al. . *Developing a context-sensitive tour guide*. In Proceedings of the 1st Workshop on Human Computer Interaction for Mobile Devices, 1998, Glasgow, Scotland.

[Dey98] Dey A. K, Abowd G. D, and Andrew Wood. CyberDesk: a Framework for Providing Self-Integrating Context-Aware Services. In Proceedings of the Third International Conference on Intelligent User Interfaces, pages 47–54, San Francisco, California, USA, 1998. ACM Press.

[Dey99] Dey A. K, and Abowd G. K, Towards a Better Understanding of Context and Context-Awareness. Technical Report GIT-GVU-99-22, Georgia Institute of Technology, College of Computing, Atlanta, Georgia, USA, June 1999.

[Dey00a] Dey A. K, and Abowd G. D, CybreMinder: A Context- Aware System for Supporting Reminders. In Proceedings of the Sec- ond International Symposium on Handheld and Ubiquitous Com- puting (HUC2K), pages 172–186, Bristol, UK, 2000.

[DeyOOb] Dey A. K. Providing Architectural Support for Building Context-Aware Applications. PhD thesis, Georgia Institute of Technology, Atlanta, Georgia, USA, 2000.

[Dey01] Dey A. K, Salber D, and Abowd G. D, A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human- Computer Interaction Journal* 16(2-4), pp. 97-166, 2001.

[Dourish00] Dourish P, Edwards W. K, LaMarca et al, *Extending document management systems with active properties*. ACM Transactions on Information Systems, 2000, Vol.18, N°2, p. 140-170.

[Dertouzos] Dertouzos M. L, The Future of Computing. Scientific Amer- ican, 281(2):52–63, August 1999.

[DSTC04] DSTC, IBM et CBOP. MOF Query / Views / Transformations - Second Revised Submission, ad/2004-01-06, January 2004.

[Elser99] Esler M, Hightower J, Anderson G, and Borriello G, Next Century Challenges: Data-centric Networking for Invisible Computing: the Portolano Project at the University of Washington. In Proceedings of the Fifth Annual ACM/IEEE Interna-tional Conference on Mobile Computing and Networking (MOBI- COM), pages 256–262, Seattle, Washington, United States, August 1999. ACM Press.

[Erik01] Erik C, Francisco C, Greg M, and Sanjiva W, Web Services Description Language (WSDL) 1.1. Specification available at http://www.w3.org/TR/wsdl, 2001. (Last accessed 1st March 2006).

[Espinoza01] Espinoza F, Persson P, Sandin A, Nystrom H, Cac-ciatore E, and Bylund, M. (2001) 'GeoNotes: social and navigational aspects of location-based information systems', *Proceedings of the 3rd International Conference on Ubiquitous Computing*, Atlanta, Georgia, USA, pp.2–17.

[Farcet03a] Farcet N, *MDA assessment and deployment within Thales*, [en ligne], 2003, Disponible sur :

http://aristote1.aristote.asso.fr/Presentations/CEA-EDF-2003/Conferences/NicolasFarcet/MIRROR-EDF-CEA-summer-school-2003-publication.ppt

[Farcet03b] Farcet N, *MDA assessment and deployment within Thales*, 16 - 27 Juin 2003, Ecole d'Eté d'Informatique CEA - EDF - INRIA 2003 [en ligne], Disponible sur :

http://www.aristote.asso.fr/Presentations/CEA-EDF-

2003/Conferences/NicolasFarcet/Video/Conference_5_Thales.rm

[Feiner97] Feiner S, MacIntyre B, Hollerer et al.. *A Touring Machine : Prototyping 3D mobile augmented reality systems for exploring the urban environment*. Personal Technologies, 1997, Vol. 1, N°4, pp. 208-217

[Fels98] Fels S, Sumi Y, Etani et al. *Progress of C-MAP: A context-aware mobile assistant*. (1998). In Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments. Menlo Park, CA : AAAI Press, pp. 60-67

[Frankel03] Frankel D. S, Model Driven Architecture Applying MDA to Enterprise Computing. 2003, ISBN 0-471-31920-1

[Gardner03] Gardner. T, Griffin. C, Koehler. J, et Hauser. R, A Review of OMG MOF 2.0 Query / Views / Transformations Submissions and Recommendations towards the final Standard. July 2003.

[Garlan02] Garlan D, Siewiorek D. P, Smailagic A, and Steenkiste P, Project Aura: Toward Distraction-Free Pervasive Computing. IEEE Pervasive Computing, 1(2):22–31, 2002.

[Gervais02] Gervais M. P, Towards an MDA-Oriented methodology. In COMPSAC '02 : Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life : Development and Redevelopment, pages 265– 270, Washington, DC, USA, August 2002. IEEE Computer Society.

[GMT] http://www.eclipse.org/gmt

[Gronmo04] Gronmo. R, Skogan. D, Solheim. I, et Oldevik. J, Model-Driven Web Services Development. 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04), pages 42–45, March 2004.

[Gudgin02a] Gudgin M, Hadley M, Mendelsohn N, Moreau. J. J, and Frystyk. H, Nielsen. SOAP Version 1.2 Part 1 – Messaging Framework, 19 Décembre 2002. http ://www.w3.org/TR/2002/CR-soap12-part1-20021219.

[Gudgin02b] Gudgin M, Hadley M, Mendelsohn N, Moreau J. J, and Frystyk. H, Nielsen. SOAP Version 1.2 Part 2 – Adjuncts, 19 Décembre 2002. http ://www.w3.org/TR/2002/CR-soap12-part2-20021219.

[Harrison98] Harrison B. L, Fishkin K. P, Gujar et al, *Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces*. In Proc. CHI 98, April 18-23, Los Angeles. New York, NY, USA : ACM, 1998, pp.17-24

[Healey98] Healey J, and Picard R, W. *StartleCam: A cybernetic wearable camera*. In Proceedings of the 2nd International Symposium on Wearable Computers (ISWC98), Pittsburgh, Pennsylvania, 19-20 October 1998. Los Alamitos CA : IEEE ,1998, p. 42-49

[Heiner99] Heiner J. M, Hudson S. E, and Tanaka K, *The Information Percolator: Ambient information display in a decorative object*. **In** Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology (UIST96). New York, NY: ACM Press, 1999.

[HTTP97] Network Working Group. *Hypertext Transfer Protocol { HTTP/1.1, ftp://ftp.rfc-editor.org/in-notes/rfc2068.txt*, 1997.

[IOSGPT04] INTERACTIVE OBJECTS SOFTWARE GMBH AND PROJECT TECHNOLOGY, INC.. 2nd Revised Submission to MOF Query / View / Transformation RFP, January 2004. Disponible sur http://www.omg.org/docs/ad/04-01-14.pdf,

[Java02] JAVA COMMUNITY PROCESS. JavaTMMetadata Interface(JMI) Specification, Version 1.0, June 2002.

[Joaquin03a] Joaquin M, MDA *Model Driven Architecture*, [en ligne], 2003;Disponible sur :http://www.joaquin.net/MDA/Model_Driven_Architecture_Tutorial.pdf

[Joaquin03b] Joaquin M, The several styles of *Model Driven Architecture*, [en ligne], 2003, Disponible sur : http://www.joaquin.net/MDA/The_Several_Styles_of_MDA.pdf

[Joaquin03c] JOAQUIN M, *What's a Platform -- An Overview of Model Driven Architecture*, [en ligne], 2003, Disponible sur :

http://www.joaquin.net/MDA/What%27s_a_Platform_An_Overview_of_Model_Driven_Architecture.pdf

[Kadima03] Kadima H, and Monfort V, Les Web Services – Techniques, démarches et outils – XML, WSDL, SOAP, UDDI, Rosetta, UML. Dunod, mars 2003.

[Kalnins04] Kalnins A, Barzdins J, et Celms E, Basics of Model Transformation Language MOLA. Workshop on Model Driven Development (WMDD 2004) at ECOOP 2004, June 2004.

[Kavantzas05] Kavantzas N, Burdett D, Ritzinger G, Fletcher T, Lafon Y, and Barreto C, Web Services Choreography Description Language Vesion 1.0, Novembre 2005. W3C Candidate Recommendation.

[Kent03] Kent. S, et Smith. R, The Bidirectional Mapping Problem. Electronic Notes in Theoretical Computer Sciences, 82(7), 2003.

[Kerer04] Kerer C, Dustdar S, Jazayeri M, Gomes D, Szego A, and Caja J. A. B, (2004) 'Presence-aware infrastructure using web services and RFID technologies', *Proceedings of the 2nd European Workshop on Object Orientation and Web Services*, Oslo, Norway.

[Kleppe03] Kleppe A, Warmer J, and Bas. W, MDA Explained- The Model-Driven Architecture: Practice and Promise. 2003, ISBN 0-321-19442-X

[Lemesle00] Lemesle R, *Techniques de modélisation et de Méta-Modélisation*, Thèse, Université de Nantes, [en ligne], 26 octobre 2000, 276 p., Disponible sur :

http://www.sciences.univ-nantes.fr/info/lrsg/Recherche/mda/richard.pdf

[Lamming94] Lamming M. and Flynn M, Forget-me-not: Intimate computing in support of human memory. Proceedings of the FRIEND 21: International Symposium on Next Generation Human Interfaces, 1994, Tokyo, pp.125-128

[Leymann01] Leymann F, Web Services Flow Language (WSFL 1.0), Mai 2001. http ://www- 3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.

[Marschall03] Marschall. F, et Braun. P, Model Transformations for the MDA with BOTL. Proceedings of the Workshop on Model Driven Architecture: Foundations and Applications, June 2003.

[Marmasse00] Marmasse N, and Schmandt C, *Location aware information delivery with comMotion*. In Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC2K). Heidelberg, Germany : Springer Verlag, 2000.

[McCarthy99] McCarthy J. F, and Meidel E. S, *ActiveMap: A visualization tool for location awareness to support informal interactions*. **In** Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC 99). Heidelberg, Germany: Springer Verlag, 1999.

[McCarthy00] McCarthy J. F, and Anagost T. D, *EventManager: Support for the peripheral*

awareness of events. In Proceedings of the 2nd International Symposium on Handheld and Ubiquitious Computing (HUC2K). Heidelberg, Germany: Springer Verlag, 2000.

[MDSD04] Model-Driven Software Development,

http://www.mdsd.info/mdsd_cm/page.php?page=mdsd, 2004

[Mitra02] Mitra N, SOAP Version 1.2 Part 0 – Primer, 19 Décembre 2002. http ://www.w3.org/TR/2002/CR-soap12-part0-20021219.

[MTL] http://modelware.inria.fr/rubrique5.html.

[Muller04] Muller A, *Transformation de modèles : d'un modèle abstrait aux modèles EJB et CCM*, [en ligne], 2004, Disponible sur :

http://www.lifl.fr/lmo2004/slides_lmo2004/muller.pdf

[Munõz03] Munõz M. A, Gonzalez V. M, Rodriguez M, and Fa vela. J, (2003) 'Supporting context-aware collaboration in a hospital: an ethnographic informed design', *Proceedings of Workshop on Artificial Intelligence, Information Access, and Mobile Computing 9th International Workshop on Groupware, CRIWG 2003*, Grenoble, France, pp.330–334

[OMG02] OMG. Request for Proposal: MOF 2.0 Query/Views/Transformations RFP, October 2002. Disponible sur http://www.omg.org/docs/ad/02-04-10.pdf,

[OMG03a] OMG, *MDA Guide Versions 1.0.1*, [en ligne], juin 2003, Disponible sur :

http://www.omg.org/docs/omg/03-06-01.pdf

[OMG03b] Object Management Group, XML Metadata Interchange Specification, Version 2.0, OMG Document: formal/03-05-02, May 2003.

[OMG03c] OMG. Unified Modeling Langage: Superstructure version 2.0 Final Adopted Specification, OMG ptc/03-08-02. OMG, August 2003. ,

[Oum03] OUM O. S, *Interopérabilité des Outils d'assistance à l'évolution des systèmes logiciels*, Mémoire de DEA, Laboratoire d'Informatique du Littoral, [en ligne], 26 juin 2003, 77 p., Disponible sur : http://lil.univ-littoral.fr/~oumoumsack/publi/MemDocFinal.pdf

[Pascoe97] Pascoe J, The Stick-e Note Architecture: Extending the Interface Beyond the User. In Johanna Moore, Ernest Edmonds, and Angel Puerta, editors, Proceedings of the International Conference on Intelligent User Interfaces, pages 261–264, Orlando, Florida, USA, January 1997. ACM.

[Pascoe98] Pascoe J, *Adding generic contextual capabilities to wearable computers.* In Proceedings of 2 nd International Symposium on Wearable Computers, October 1998, pp. 92-99

[Patrascoiu04] Patrascoiu. O, YATL: Yet Another Transformation Language. In Proceedings of the 1st European MDA Workshop MDA-IA, pages 83–90, January 2004.

[Peltier03] Peltier M, Techniques de Transformation de Modèles Basées sur la méta-modélisation. Thèse de Doctorat, UFR Sciences et Techniques, Université de Nantes, 2003.

[QVT04] QVT-MERGE GROUP. Revised submission for MOF 2.0 Query/Views/Transformations RFP (ad/2002-04-10), April 2004. Disponible sur http://www.omg.org/docs/ad/04-04-01.pdf,

[Rapela04] Rapela D, *MODA-TEL (Deliverable 3.3), MDA modelling and application principles* [en ligne], mai 2004, 75 p., Disponible sur : http://www.modatel.org/~Modatel/pub/deliverables/D3.3-final.pdf

[Rekimoto99] Rekimoto J, *Time-Machine Computing : A time-centric approach for the information environment*. In Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology (UIST99), Asheville, USA. New York : ACM Press, 1999, p .45 – 54

[Rhodes97] Rhodes B. J, The wearable remembrance agent. Los Alamitos. In Proceedings of the 1st International Symposium on Wearable Computers (ISWC97), October 13-14, 1997, Cambridge, Massachusetts. Los Alamitos, Calif. : IEEE Press, 1997, 123-128

[Ryan99] Ryan N, ConteXtML: Exchanging Contextual Information between a Mobile lient and the FieldNote Server. Language Specifications available at http://www.cs.kent.ac.uk/ projects/mobicomp/fnc/ConteXtML.html, 1999. (Last accessed 1st March 2006).

[Schilit94] Schilit B, Adams N, and Want R, Context-Aware Computing Applications. In Proceedings of the IEEE Workshop on Mo- bile Computing Systems and Applications, Santa Cruz, CA, USA, 1994.

[Schmidt99] Schmidt A, Asante K. A, Takaluoma A,Tuomela U, Laerhoven.K.V, and Van de Velde . W, Advanced Interaction in Context. In Proceedings of the First Interna- tional Symposium on Handheld and Ubiquitous Computing (HUC), pages 89–101, Karlsruhe, Germany, 1999. Springer-Verlag.

[Sheng05] Sheng Q. Z, and Benatallah B, 2005. ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services, The 4th International Conference on Mobile Business (ICMB 2005), IEEE Computer Society. Sydney, Australia.

[SINTEF04] SINTEF. UML Model Transformation Tool (UMT), december 2004. Disponible sur http://umtqvt. sourceforge.net,

[Skogan04] Skogan. D, Gronmo. R, et Solheim. I, Web Service Composition in UML. Eight IEEE International Enterprise Distributed Object Computing Conference (EDOC 2004), pages 47–57, September 2004.

[Shumao06] Shumao O, Nektarios G, Manooch A. K. Y, and Xiantang S, 2006. A Model Driven Integration Architecture for Ontology-Based Context Modelling and Context-Aware Application Development, Springer-Verlag Berlin Heidelberg. A. Rensink and J. Warmer (Eds.): ECMDA-FA 2006, LNCS 4066, pp. 188 – 197.

[Soap00] SOAP Specifications. Website available at http://www.w3.org/ TR/soap/, 2000. (Last accessed 1st March 2006).

[Sriplakich03] Sriplakich P, *Techniques des transformations de modèles basées sur la métamodélisation*, Mémoire de DEA Systèmes Informatiques Répartis, Université Pierre et Marie Curie, [en ligne], septembre 2003, 64 p., Disponible sur :

http://modfact.lip6.fr/ModFactWeb/qvt/SimpleTRL.pdf

[Strang03] Strang T, and Linnhoff-Popien C, Service Interoperabil-ity on Context Level in Ubiquitous Computing Environ-ments. Intl. Conf. on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet (SSGRR2003w), L'Aquila, Italy, January 2003.

[Weiser95] Weiser M, 1995. The computer for the 21st century. In *Human-Computer interaction: Toward the Year 2000*, R. M. Baecker, J. Grudin, W. A. Buxton, and S. Greenberg, Eds. San Francisco, CA : Morgan Kaufmann Publishers, pp. 933-940

[Weiser97] Weiser M. and Brown, J. S, *The coming age of calm technology*. In Denning, P. J. & Metcalfe, R. M. (Eds.) Beyond Calculation: The Next Fifty Years of Computing. New York : Springer Verlag, 1997.

[Weiser99] Weiser M, 1999. Some computer science issues in ubiquitous computing. In

Mobility: Processes, Computers, and Agents. New York, NY : ACM Press/Addison-Wesley Publishing Co., pp. 420-430

[Willink03] Willink E.D, UMLX - A Graphical Transformation Language for MDA. Workshop on Model Driven Architecture Foundations and Applications, pages 13–24, June 2003.

[Thatte01] Thatte S, XLANG - Web Services For Business Process Design, 2001.

http ://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm.

[Tom02] IBM Tom Bellwood, Microsoft Luc Clément, IBM David Ehnebuske, IBM Andrew Hately, IBM Maryann Hondo, HP Yin Leng Husband, Microsoft Karsten Januszewski, Oracle Sam Lee, IBM Barbara McKee, Intel Joel Munter, and SAP Claus von Riegen. UDDI Version 3, 19 Juillet 2002. http ://uddi.org/pubs/uddiv3.00- published-20020719.htm.

[UMT] http://umt-qvt.sourceforge.net/.

[Vale08] Vale S, Hammoudi S, "Towards Context Independence in Distributed CAA by the MDA". ACM SIPE'08, Sorrento, Italy, July 7, 2008.

[Ward97] Ward A, Jones A, and Hopper A, A New Location Technique for the Active Office. IEEE Personal Communications, 4(5):42–47, October 1997.

[XML96] World Wide Web Consortium (W3C). *Extensible Markup Language, http://www.w3.org/XML/*, 1996.

[BPEL] Andrews T et al, Business Process Execution Language for web services 1.1. http://www106.ibm.com/developerworks/library/ws-bpel.