HAL
open science

Mobilité et sensibilité au contexte pour la gestion de documments multimédias personnels : CoMMediA

Windson Viana de Carvalho

To cite this version:

Windson Viana de Carvalho. Mobilité et sensibilité au contexte pour la gestion de documments multimédias personnels : CoMMediA. Informatique [cs]. Université Joseph-Fourier - Grenoble I, 2010. Français. NNT : . tel-00499550

HAL Id: tel-00499550

https://theses.hal.science/tel-00499550v1

Submitted on 9 Jul 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° attribué par la bibliothèque
|__|__|__|__|__|__|__|__|__|__|

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE DE GRENOBLE

Spécialité : Informatique

préparée au Laboratoire d'Informatique de Grenoble

dans le cadre de l'École Doctorale
Mathématiques, Sciences et Technologie de l'Information, Informatique

présentée et soutenue publiquement

par

Windson VIANA DE CARVALHO

Février 2010

Mobilité et sensibilité au contexte pour la gestion de documents multimédias personnels : CoMMediA

Directeur de Thèse : Hervé MARTIN

JURY

M. Lionel BRUNIE	Rapporteur
M. Philippe ROOSE	Rapporteur
M. Didier DONSEZ	Examinateur
M. José Celso FREIRE	Examinateur
M. Hervé MARTIN	Directeur de Thèse
M. Jérôme GENSEL	Co-encadrant

Remerciements

J'aimerai tout d'abord remercier Monsieur Hervé MARTIN et Monsieur Jérôme GENSEL pour m'avoir accueilli au sein de leur équipe de recherche et donné la chance de pouvoir m'exprimer à travers ce que j'aime faire. Je voudrais leur remercier pour m'avoir fait confiance en se lançant dans les idées de ma thèse. Je remercie également pour leur bienveillante attention qui m'a constamment accompagnée tout au long de cette thèse. Leur encadrement et leurs conseils m'ont été très précieux. Je leur prie d'être assurés de ma sincère reconnaissance.

J'adresse également mes sincères remerciements à Madame Marlene VILLANOVA-OLIVER pour m'avoir guidé et encouragé durant mon séjour à Grenoble, pour ses conseils et surtout pour son optimisme qui ont été beaucoup apprécié et qui m'ont permis de développer et de rédiger ce mémoire.

Je remercie les rapporteurs de ma thèse Monsieur Lionel BRUNIE, Monsieur Philippe ROOSE et d'avoir accepté de juger ce travail. Je remercie également les examinateurs Monsieur Didier DONSEZ et Monsieur José Celso FREIRE pour avoir accepté de participé de mon jury de thèse. Qu'ils trouvent ici le témoignage de ma reconnaissance.

Tous mes remerciements et profondes amitiés à mes anciens et présents collègues du laboratoire, plus particulièrement Manu, Céline, Bogdan, Marius, Samira, Dia, Angela, Aurélie, Gaël, Raphael, Laurent, Christine, Sandro, Benoit, Isaac, Carlos, José, Reinaldo, Carina et Yan pour leur gentillesse, leur bonne humeur et la chaleureuse ambiance qui vont rester gravées dans ma mémoire.

Et puis, il y a toutes les personnes qui m'ont entouré durant ces dernières années et qui ont fait que les journées semblaient toujours trop courtes. Mes amis qui m'ont soutenus et encouragés tout au long de cette thèse, plus particulièrement Elton, Raquel, Ledi, Renan, Cris Dora, Isaac, Shirley, Fabi, Lucas, Claudia, Rodrigo, Poly, Manu, Angelo, Céline, Marius, Bogdan, Rafa, Claudio, Bernard, Aida, Emma, Marina, Leo, Carlos, Andréia, Luciano, Edson, Mateus, Jana, Lina, Davida, Ale et Nina.

Mes remerciements s'adressent également à toutes les personnes avec qui j'ai l'opportunité de co-encadrer durant ma thèse, en particulier, Tianyi, Lilia, Céline Défaut, Perrine, Gaël, Fong, qui ont contribué à l'aboutissement de ce travail.

Je tiens tout particulièrement à remercier Madame Rossana ANDRADE de l'Université Fédéral do Ceará pour m'avoir guidé pendant ma carrière scientifique et pour mon ouvrir mon appétit pour la recherche et a rendu possible mon arrivée à Grenoble et mon retour au Brésil.

Je veux également remercier mes parents, mes frères, ma belle famille et tous mes neveux et nièces. Merci beaucoup pour le soutien que vous m'avez apporté tout au long de mes études. Je vous aime.

Je remercie bien évidement et avant tout mon épouse Karoline SABOIA. Ce travail est surtout pour et grâce à toi (Te amo princesa).

OBRIGADO A TODOS !

Windson

Table des matières

1 INTRODUCTION 1

1.1 PROBLEMATIQUE 1
1.1.1 La gestion sensible au contexte de documents multimédias personnels 2
1.1.2 Applications mobiles, multimédias et sensibles au contexte 3
1.2 APERÇU DE LA PROPOSITION 4
1.3 ORGANISATION DU DOCUMENT 6

2 L'EVOLUTION ET LES NOUVEAUX USAGES DES DISPOSITIFS MOBILES 11

2.1 L'EVOLUTION DES ASSISTANTS PERSONNELS 12
2.2 L'HISTOIRE DU TELEPHONE MOBILE 14
2.3 LA CONVERGENCE 15
2.4 LES USAGES D'AUJOURD'HUI ET LES PERSPECTIVES POUR DEMAIN 16

3 GESTION DE DOCUMENTS MULTIMEDIAS PERSONNELS 19

3.1 LE DOCUMENT MULTIMEDIA PERSONNEL ET SES USAGES 19
3.1.1 Les nouveaux comportements dans la création de documents multimédias personnels 20
3.1.2 Les outils desktop d'aide à la gestion de multimédias personnels 22
3.2 L'ANNOTATION SEMANTIQUE DE DOCUMENTS MULTIMEDIAS 23
3.2.1 Les mots-clés 24
3.2.2 Les inconvénients de l'annotation dans les outils de gestion de multimédias 24
3.2.3 La représentation de l'annotation 25
3.2.4 Génération automatique de métadonnées 32
3.3 LA SENSIBILITE AU CONTEXTE ET LA GESTION DE DOCUMENTS MULTIMEDIAS 33
3.3.1 Le temps 35
3.3.2 L'espace 35
3.3.3 Le contexte de création 36
3.3.4 La mobilité et la sensibilité au contexte pour l'annotation et l'organisation de multimédias 38
3.4 RECHERCHE DE DOCUMENTS MULTIMEDIAS 40
3.4.1 Recherche par le contenu 40
3.4.2 Recherche à base de propriétés 41
3.4.3 Recherche par texte 42
3.4.4 Recherche à l'aide d'informations contextuelles 43
3.5 PARTAGE DE DOCUMENTS MULTIMEDIAS ET D'INFORMATIONS CONTEXTUELLES 44
3.5.1 Mobilité et sensibilité au contexte pour le partage de documents multimédias 44
3.5.2 Partage d'informations contextuelles 47
3.6 SYNTHESE 48

4 L'INFORMATIQUE MOBILE 51

4.1 DEVELOPPEMENT SUR LES PLATES-FORMES MOBILES 51
4.1.1 Les applications Web pour les dispositifs mobiles 52
4.1.2 Les plates-formes logicielles mobiles et leurs kits de développement 55
4.2 SYNTHESE DU DEVELOPPEMENT D'APPLICATIONS MOBILES 56
4.3 DEPLOIEMENT D'APPLICATIONS SUR LES PLATES-FORMES MOBILES 57
4.3.1 Les étapes du processus de déploiement d'applications 58
4.4 SYNTHESE DU DEPLOIEMENT D'APPLICATIONS MOBILES 62

5 SENSIBILITE AU CONTEXTE ET ADAPTATION D'APPLICATIONS MOBILES 65

5.1 ADAPTATION D'APPLICATIONS MOBILES 66
5.2 LA SENSIBILITE AU CONTEXTE 67
5.2.1 La notion de contexte 68
5.2.2 Modèles de contexte 69

5.2.3 Les architectures de gestion de contexte ... *71*
5.3 PROFILS DE DESCRIPTION DE DISPOSITIFS MOBILES ... 74
5.3.1 Composite Capabilities/Preference Profiles (CC/PP) ... *74*
5.3.2 User Agent Profile (UAProf) ... *76*
5.3.3 Wireless Universal Resource File - WURFL ... *77*
5.3.4 Profils de description des plates-formes mobiles ... *77*
5.4 ADAPTATION D'APPLICATIONS MOBILES LORS DU DEVELOPPEMENT ... 79
5.4.1 Les approches basées sur des directives ... *80*
5.4.2 Les approches dirigées par les modèles ... *81*
5.5 ADAPTATION DYNAMIQUE ET DEPLOIEMENT ADAPTATIF D'APPLICATIONS MOBILES ... 82
5.5.1 Programmation par composants et OSGi ... *83*
5.5.2 Déploiement adaptatif d'applications mobiles ... *87*
5.5.3 Adaptation dynamique d'une application mobile ... *92*
5.6 SYNTHESE ... 97

PROBLEMATIQUE ET APPROCHE ... 101

6 MODELES POUR LA REPRESENTATION DES METADONNEES DE DOCUMENTS MULTIMEDIAS PERSONNELS ... 105

6.1 L'ONTOLOGIE CONTEXT TOP ... 106
6.1.1 Notion élargie de contexte ... *106*
6.1.2 ContextTop ... *109*
6.2 L'ONTOLOGIE CONTEXTMULTIMEDIA ... 111
6.2.1 Quels concepts sont les plus pertinents pour l'annotation ? ... *112*
6.2.2 Ontologie ContextMultimedia ... *115*
6.2.3 Exemple d'annotation ... *123*
6.3 CONTEXTANNOTATOR : UNE PLATE-FORME D'ACQUISITION ET D'ENRICHISSEMENT DES METADONNEES ASSOCIEES AUX DOCUMENTS MULTIMEDIAS PERSONNELS ... 127
6.3.1 Architecture de ContextAnnotator ... *128*
6.3.2 Processus d'interprétation du contexte de création ... *131*
6.3.3 Processus d'inférence de métadonnées ... *137*
6.4 CONCLUSION ... 144

7 DEPLOIEMENT ADAPTATIF ET ACQUISITION DE CONTEXTE ... 146

7.1 DEPLOIEMENT ADAPTATIF D'APPLICATIONS MOBILES ... 147
7.1.1 Description des profils de dispositifs mobiles ... *150*
7.1.2 Description d'une méta-application ... *153*
7.1.3 L'infrastructure à composants de CoMMeDiA ... *158*
7.1.4 Génération d'une version d'application ... *162*
7.2 DEVAC - DEVICE AWARE CONTEXT TOOLKIT ... 165
7.2.1 Les services d'acquisition de contexte ... *165*
7.2.2 Déploiement adaptatif d'une application sensible au contexte ... *172*
7.3 CONCLUSION ... 176

8 COMMEDIA ET EXPLOITATION DES METADONNEES CONTEXTUELLES ... 179

8.1 ARCHITECTURE COMMEDIA ... 179
8.2 INDEXATION SEMANTIQUE DE MULTIMEDIAS PERSONNELS ANNOTES PAR CONTEXTMULTIMEDIA ... 180
8.2.1 Vers des systèmes de recherche contextuelle de documents multimédias ... *180*
8.2.2 Notre proposition d'indexation sémantique ... *182*
8.2.3 Génération des termes d'indexation ... *184*
8.2.4 Poids des termes d'indexation ... *186*
8.2.5 Requête et mise en correspondance ... *189*
8.2.6 Processus d'expansion spatiale ... *190*
8.2.7 Exemple d'expansion spatiale ... *193*
8.3 PARTAGE SENSIBLE AU CONTEXTE DE DOCUMENTS MULTIMEDIAS PERSONNELS ... 194
8.3.1 Notification sensible au contexte de publication de documents multimédias ... *195*
8.3.2 Génération automatique de messages contextuels ... *198*

8.4 CONCLUSION 200

9 IMPLEMENTATION ET EXPERIMENTATIONS 201

9.1 PHOTOMAP 201

9.1.1 Application mobile 202

9.1.2 Application de bureau d'aide à l'annotation 205

9.1.3 Le système Web d'organisation et de publication de photos 208

9.2 EXPERIMENTATIONS 211

9.2.1 Indexation Sémantique 211

9.2.2 Évaluation du déploiement adaptatif 216

10 CONCLUSION 219

ANNEXE 1 : LES PLATES-FORMES LOGICIELLES MOBILES ET LEURS KITS DE DEVELOPPEMENT 227

11 BIBLIOGRAPHIE 243

Table des illustrations

Figures

Figure 1 - Vue globale de la proposition. 5
Figure 2 - Évolution du nombre de clients des opérateurs mobiles. Source : *ITU*. 11
Figure 3 - Illustration de la classification de dispositifs mobiles selon leur portabilité [Ganneau, 2009].... 12
Figure 4 - Deux dispositifs mobiles conçus par Apple : l'Apple Newton (1993) et l'iPhone (version 2007). Source de la figure : www.engadget.com. 13
Figure 5 - Évolution de dispositifs mobiles ver la convergence aux smartphones. 15
Figure 6 - Widget mobile pour accompagner en temps réel les positions des voitures pendant une course de Formule 1. 18
Figure 7 - Les téléphones mobiles suréquipés de capteurs. Source de la figure : Citron [Yamaba *et al.*, 2005]. 18
Figure 8 - Comparaison, fournie par InfoTrend, du nombre de téléphones mobiles dotés d'appareils photos numériques, et le nombre d'appareils conventionnels. 21
Figure 9 - Une vue satellitaire des photos prises à Paris disponibles sur le site Web 2.0 de partage de photos et vidéos Flickr. 22
Figure 10 - Prise de vue de l'outil PICASA. 23
Figure 11 - Nuage des mots-clés les plus populaires du site Flickr. 24
Figure 12 - Exemple d'utilisation de RDF pour la représentation de métadonnées d'une photo (adapté du site du MAWG). 29
Figure 13 - Diagramme UML de l'ontologie d'annotation de photo proposée par Schreiber [Schreiber *et al.*, 2001]. 30
Figure 14 - Ontologie OWL d'annotation de photos personnelles [Chai *et al.*, 2008]. 30
Figure 15 - Taxonomie des personnes. 31
Figure 16 - Découpage d'une image en zones pour aider l'annotation sémantique sur le contenu 32
Figure 17 - Processus multicritère de suggestion d'annotation proposé en [Elliott *et al.*, 2008]. 33
Figure 18 - Comparaison entre les aspects perçus comme étant importants pour une recherche et ceux effectivement décrits par les utilisateurs [Naaman *et al.*, 2004]. 34
Figure 19 - Interface d'organisation automatique de photos de l'outil PhotoCompas [Naaman *et al.*, 2004]. 36
Figure 20- Les interfaces de validation d'annotation de MediAssist [O'Hare *et al.*, 2006][O'Hare *et al.*, 2007]. 37
Figure 21 - La suggestion d'annotation sur le service ZoneTag. 39
Figure 22 - Proposition d'identification des personnes sur une photo à l'aide de Bluetooth [Monaghan *et al.*, 2006]. 39
Figure 23 - Schéma général du processus de mise en correspondance en utilisant le modèle vectoriel, adapté de [Wang *et al.*, 2006]. 42
Figure 24 - Interface de partage de multimédias du widget mobile ShoZu. 45
Figure 25 - Exemple de MMS enrichi par des informations automatiques sur le contexte de l'utilisateur capturées à l'aide de la plate-forme ContextPhone [Raento *et al.*, 2005]. 46
Figure 26 - Le système CONNECTO de partage de location et d'activités [Barkhuus *et al.*, 2008]. 47
Figure 27 - Partage d'informations contextuelles. 48
Figure 28 - Nuage des termes de ce chapitre généré à l'aide du logiciel Wordle. 49
Figure 29 - Schémas de flux de données lors d'accès à une page Web : I) au sein des approches de générations de versions; II) au sein des approches de proxy d'adaptation. 54
Figure 30 - Un processus générique de déploiement [Lestideau *et al.*, 2001]. 58
Figure 31 - Exemples de portails d'applications mobiles : l'Apple App Store et le Nokia Ovi Store. 60
Figure 32 - Les différentes phases de déploiement d'une application mobile sur la plate-forme Windows Mobile. Source portail MSDN. 63
Figure 33 - Site de téléchargement de l'application Opera Mini. L'utilisateur doit indiquer le modèle de son dispositif afin de télécharger la bonne version du navigateur mobile. 64
Figure 34 - Modèle objet proposé par Kirsch [Kirsch, 2006] pour la représentation du contexte de systèmes coopératifs. 70
Figure 35 - Modèle de contexte de l'architecture SOCAM. 71

Figure 36 - Ontologie pour la représentation de contexte de l'architecture CoCA [Ejigu *et al.*, 2008]. 71
Figure 37 - Architecture générale d'un système sensible au contexte proposée par [Henricksen *et al.*, 2004]. 73
Figure 38 - Exemple de Profil CC/PP (Source: W3C). 75
Figure 39 - Ontologie CoDaMoS pour la description de dispositifs mobiles et des profils/configurations de la plate-forme Java [Wagelaar, 2005]. 79
Figure 40 - Mécanisme de génération de versions de l'approche J2MEPolish. 81
Figure 41- Cycle de vie des *bundles* sur OSGi. 86
Figure 42 - Processus de déploiement adaptatif OTA-PSD (Source :[Cong *et al.*, 2008]). 88
Figure 43 - Schéma généralisé des approches de déploiement adaptatif orientées composants [Ayed *et al.*, 2008] [Donsez, 2006] [Fjellheim, 2006]. 90
Figure 44 - Deux applications différentes qui contrôlent le même dispositif UPnP (Source : [Donsez, 2006]). 91
Figure 45- Un fragment du plan de déploiement proposé dans l'architecture CAdeComp. 91
Figure 46 - Proposition ACEEL: I) script des politiques d'adaptation, II) Diagramme de classes avec la description des alternatives de comportement d'un composant. 95
Figure 47- Schéma de reconfiguration dynamique proposée par Louberry *et al.* [Louberry *et al.*, 2008]. 96
Figure 48 - Contexte défini comme étant l'intersection de la zone d'intéret du système et de sa zone d'observation. 107
Figure 49 - Processus de caractérisation du contexte. 108
Figure 50 - Illustration graphique de notre ontologie Context Top. 110
Figure 51 - Extension et instanciation de l'ontologie Context Top. 111
Figure 52 - Aspects contextuels couvert par l'annotation de multimédias personnels. 113
Figure 53 - Exemple d'annotation d'une photo. 115
Figure 54 - Squelette des concepts de l'ontologie ContextMultimedia. 115
Figure 55 - Éléments de la dimension spatiale de ContextMultimedia. 117
Figure 56 - Exemples d'utilisation des relations spatiales qualitatives. 118
Figure 57 - Concepts et relations de la dimension sociale du contexte de création. 119
Figure 58 - Concepts des dimensions spatio-temporelle et informationnelle de ContextMultimedia. 120
Figure 59 - Exemple d'annotation relative au contexte informationnel. 121
Figure 60 - Représentation d'une collection et de la notion d'événement dans ContextMultimedia. 122
Figure 61 - Exemple de collection associée à un événement. 122
Figure 62 – Représentation visuelle de l'annotation d'une photo à l'aide de *ContextMultimedia*. 124
Figure 63 – L'annotation sur le contenu et sur les contextes spatial et social à l'aide de *ContextMultimedia*. 125
Figure 64 - Cas d'utilisation de ContextAnnotator. 128
Figure 65 - La modélisation de services d'enrichissement de métadonnées 129
Figure 66 - Architecture de la plate-forme ContextAnnotator. 130
Figure 67 - Services de la dimension spatiale. 132
Figure 68 - Un exemple d'interface d'insertion de lieux personnels. Les polygones dessinés sur la carte indiquent le lieu de travail (« mon travail ») et objet géoréférencé (« BU Sciences ») à proximité. . 133
Figure 69 - Un exemple d'utilisation du service AllSpatialElements. 134
Figure 70 - Exemples de résultats du service de détection de visages. 137
Figure 71 - Exemple de description d'un réseau social. 138
Figure 72 – Processus d'inférence du contexte social. 141
Figure 73 - Schéma du déploiement adaptatif. 149
Figure 74 - Ontologie pour la description d'un dispositif mobile. 151
Figure 75 – D'autres concepts de l'ontologie pour la description d'un dispositif mobile. 151
Figure 76 - Ontologie de la plate-forme mobile J2ME. 152
Figure 77 - Diagramme de séquences décrivant l'installation d'une application lorsque l'extension de description du dispositif n'existe pas. 153
Figure 78 - Description d'une Méta-application. 154
Figure 79 - Exemple visuel d'une méta-application. 156
Figure 80 – Représentation en XML de méta-application. 157
Figure 81 - Cycle de vie des composants *Ilets*. 159
Figure 82 - Les classes permettant la communication entre le conteneur et le code d'un composant. 160
Figure 83 – Manifeste du composant SVGMap. 160
Figure 84 - Exemple de connexion entre les services. Le service ShowMap fournit par Ilet JPEGMap est utilisé par l'Ilet PhotoMap. 161
Figure 85 - CDG généré pour l'application de l'exemple de la Figure 79. 162

Figure 86 - Pseudo-code du processus de génération d'une application. 163
Figure 87 - La plate-forme d'acquisition de contexte DevAC. 166
Figure 88 - Organisation interne d'un composant de capture de contexte (composant du type *Sensor*). . 167
Figure 89- Framework de composants de capture. 168
Figure 90 – Définition d'un composant client du DevAC et l'exemple du Manifeste du composant TakePhoto. 170
Figure 91 - Reconfiguration Dynamique à l'aide du patron de conception *Businness Delegate*. 171
Figure 92 - Exemple d'application mobile et sensible au contexte. 172
Figure 93 - Processus de déploiement adaptatif d'une application sensible au contexte. 173
Figure 94- CGD générés pour le processus de déploiement d'une application sensible au contexte. 174
Figure 95 - Pseudo-code du processus de génération d'une application. 176
Figure 96 - Architecture CoMMediA après l'installation de l'application mobile. 180
Figure 97- Flux de données lors de l'indexation sémantique proposée. 182
Figure 98 - Nuage de tags généré automatiquement pour une collection créée à Vizille (une ville à proximité de Grenoble). 183
Figure 99 - Un exemple d'interface segmentée pour la recherche contextuelle de documents multimédias. 184
Figure 100 - Une partie de l'ontologie spatiale utilisée pour l'expansion spatiale. 192
Figure 101 - Exemple d'expansion des termes 194
Figure 102- Application du modèle ECA pour le partage sensible au contexte. 195
Figure 103 – Une partie de la grammaire EBNF d'une règle contextuelle de partage (*context rule*). 196
Figure 104 – Exemple de règle de partage (*context rule*). 196
Figure 105 – Exemple de règle de partage (*context rule*). 197
Figure 106 – Le haut de la figure illustre le patron de conception ECAA. Le bas de la figure présente notre architecture de notification. 197
Figure 107 - Application de génération d'un MMS. 198
Figure 108 - Architecture client-serveur du service de partage. 200
Figure 109 - Architecture du système PhotoMap. 202
Figure 110 - Association de coordonnés géographiques à une collection en utilisant une adresse postale. 203
Figure 111 - Capture d'écran du service d'annotation de l'application mobile de PhotoMap. 205
Figure 112 - Le flux d'information dans les sous-systèmes de PhotoMap. 206
Figure 113 - Le fichier OWL décrivant une collection de documents multimédias. 207
Figure 114 - Interface de validation d'annotations. 208
Figure 115 - Page d'entrée de l'application Web. 209
Figure 116 - Réseau social augmenté de l'utilisateur Bogdan. 210
Figure 117 – Les onglets d'annotation d'une photo. 211
Figure 118 - Interface de requête à base de propriété intégrée à PhotoMap. 213
Figure 119- Comparaison entre des requêtes avec des mots-clés manuels avec des requêtes formulées avec le nom d'un objet géoréférencé. 215
Figure 120 - Les trois images mieux classées de la requête « Tour Eiffel .where ». 215
Figure 121 - Méta-application du CA Messenger, en évidence les composants de localisation du DevAC. 217
Figure 122 - Les deux versions d'interfaces qui sont sélectionnés en fonction des caractéristique du dispositif. 217
Figure 123 - Graphe d'évolution du temps d'exécution pour générer une application mobile par le gestionnaire de déploiement. 218
Figure 124 - Le système d'exploitation Symbian OS s'exécutant sur le dispositif N95 et sur l'émulateur du SDK Series 60. 228
Figure 125 – *À gauche*) capture d'écran de l'iPhone et des émulateurs exécutant l'application de visionnement de photos en ligne : Flickr. *À droite*) Un exemple de zone de couverture du service de localisation par Wi-Fi fourni par Skyhook. 230
Figure 126 - Windows Mobile s'exécutant sur le HTC Touch Diamond et sur son émulateur pour Windows XP. 231
Figure 127 - Dispositif et émulateurs exécutant le système d'exploitation Android. 232
Figure 128 - *À gauche)* L'ensemble des technologies Java. *À droite*) la plate-forme J2ME MIDP/CLDC en détail. 233
Figure 129 - Les bibliothèques optionnelles de la plate-forme J2ME MIDP. 234
Figure 130 - Captures d'écrans de trois applications développées à l'aide la plate-forme J2ME MIDP. . 235

Tableaux

Tableau 1 - Comparaison des dispositifs Newton Message Pad 100 et l'iPhone 3G. 16
Tableau 2 - Synthèse des approches de gestion sensible au contexte de documents multimédias. 50
Tableau 3 - Un comparatif entre les terminaux mobiles disposant de l'iPhone OS 57
Tableau 4 - Une partie du vocabulaire WURFL pour la description des propriétés de la machine virtuelle J2ME MIDP embarquée sur le dispositif mobile. 77
Tableau 5 - Vue synthétique des propositions de déploiement adaptatif. 98
Tableau 6 – Propriétés de Dublin Core associées au concept Personal_Multimedia. 116
Tableau 7 – Description en OWL de certains concepts de l'ontologie ContextMultimedia. 123
Tableau 8 - Sérialisation en RDF/XML de l'annotation de la photo exemple. 125
Tableau 9- Règle SWRL pour inférer la présence des amis du créateur du document. 140
Tableau 10- Règle SWRL pour attribuer un mot-clé au document si sa date de création est le jour de l'anniversaire du créateur. 141
Tableau 11- Règle SWRL pour attribuer un mot-clé au document si sa date de création est le jour de l'anniversaire d'une personne présente lors de la création du document. 142
Tableau 12 - Règle SWRL pour inférer la présence des amis des amis du créateur du document. 143
Tableau 13 - Code de découverte des propriétés de la plate-forme mobile J2ME. 153
Tableau 14 - Suffixes utilisés sur notre approche d'indexation. 185
Tableau 15 - Exemple de calcule des poids de termes qui indexent la photo de Bob. 189
Tableau 16 - Mots raccourcis du service de MMS sensible au contexte. 199
Tableau 17 - Résultats de tests de recherche à l'aide de timbres sémantiques. 214
Tableau 18 - Temps d'exécution pour générer l'application CA Messenger par le serveur web d'installation. 218
Tableau 19 - Vue synthétique des propositions de CoMMeDiA pour la gestion sensible au contexte. 222
Tableau 20 - Vue synthétique des propositions de CoMMeDiA pour le déploiement adaptatif. 223
Tableau 21 - Comparaison entre les plates-formes mobiles. 236
Tableau 22 - Comparaison entre les plates-formes mobiles. 237
Tableau 23 - Comparaison entre les plates-formes mobiles. 237
Tableau 24 - Comparaison des activités du déploiement sur les plates-formes mobiles. 239
Tableau 25 - Comparaison des activités du déploiement sur les plates-formes mobiles. 240
Tableau 26 - Comparaison des activités du déploiement sur les plates-formes mobiles. 241

1 INTRODUCTION

1.1 Problématique

De nos jours, l'essor des technologies mobiles (communication omniprésente, dispositifs mobiles puissants, internet mobile) permet à un utilisateur nomade d'accéder à des informations depuis – théoriquement – n'importe où. L'ajout progressif de capteurs (GPS, boussole, lecteur RFID, etc.) aux dispositifs mobiles a également entraîné l'émergence de systèmes mobiles capables d'adapter les informations délivrées à un utilisateur en fonction du *contexte d'utilisation* (par exemple, les caractéristiques matérielles du dispositif, la localisation de l'utilisateur, les personnes à proximité, etc.) [Kirsch, 2006]. Ces systèmes, qualifiés de *sensibles au contexte* [Dey *et al.*, 2000], ont été l'objet d'intenses études lors de ces dernières années dont le principal objectif était la conception de mécanismes d'adaptation contextuelle des interfaces graphiques, du comportement fonctionnel, et des contenu fournis par ces systèmes [Baldauf *et al.*, 2007].

Les dispositifs mobiles, en plus d'être des dispositifs d'accès à l'information, se sont transformés en véritables studios mobiles de création de documents multimédias. Une partie des utilisateurs nomades produit des photos, des vidéos, des textes et des messages vocaux à l'aide de leurs dispositifs personnels. Ils se servent de ces dispositifs pour publier directement ces documents multimédias sur les sites Web 2.0 de partage de contenu (Flickr, Youtube, etc.). Dans ce nouveau cadre, l'information sur le *contexte d'utilisation* peut aussi jouer un rôle déterminant. En effet, le contexte d'utilisation peut offrir des informations suffisamment riches pour décrire un document multimédia (par exemple, photo prise sur le Champs de Mars, Paris, le 20 mai 2005, la nuit).

Le présent travail de thèse se situe à l'intersection de ces deux thématiques de recherche : l'Informatique Sensible au Contexte et la Gestion de Documents Multimédias. Nous souhaitons intégrer la sensibilité au contexte dans la gestion de documents multimédias personnels produits à l'aide des dispositifs mobiles, car les informations contextuelles peuvent améliorer toutes les étapes de la gestion de documents multimédias : l'annotation, l'organisation, le partage et la recherche d'un document.

Nous adressons donc ici deux problématiques principales :

I. La modélisation, l'acquisition, la représentation et l'exploitation de l'information contextuelle dans la gestion de documents multimédias personnels[1] ;

II. Le développement et le déploiement d'applications mobiles, multimédias et sensibles au contexte.

1.1.1 La gestion sensible au contexte de documents multimédias personnels

L'utilisation en masse d'appareils photos numériques, en incluant ceux intégrés aux dispositifs mobiles, et la baisse du coût du stockage de données a entraîné une augmentation quasi exponentielle de la quantité de documents multimédias sur nos ordinateurs personnels [Monaghan *et al.*, 2006]. Si le coût de la création de multimédias et de leur stockage ne pose plus de problèmes majeurs, la grande quantité de ces documents multimédias rend les processus d'organisation en collections de documents, et la recherche d'un document en particulier, plus ardus.

Les principes de la gestion de documents multimédias sont, en conséquence, d'aider les utilisateurs à retrouver plus facilement leurs documents, et de proposer des mécanismes d'automatisation du processus d'organisation. Les solutions de gestion de documents multimédias reposent essentiellement sur l'utilisation de métadonnées associées à ces documents (par exemple, la date de création, le nom du fichier, les annotations manuelles).

La représentation, l'acquisition et l'exploitation de ces métadonnées sont les thématiques de recherche centrales de ces approches. Dans la plupart de ces propositions, l'acquisition des métadonnées dépend d'un processus d'annotation extensif et manuel qui est réalisé par les utilisateurs eux-mêmes. Cependant, hormis certains utilisateurs experts, la majorité des personnes ne sont pas prêtes à consacrer du temps à indexer et annoter correctement chacun de leurs documents multimédias [Ames *et al.*, 2007]. De plus, il peut exister un écart considérable entre l'instant de la création du document et le moment de l'annotation. Cet intervalle de temps peut compliquer la tâche de l'utilisateur qui doit alors se souvenir des informations associées aux documents (Où le document multimédia a été créé ? Qui était avec moi ce jour là ?, Que contient-il ?).

Un des défis de la gestion de documents multimédias est ainsi la proposition de mécanismes d'acquisition et d'extraction automatique des métadonnées.

Des études d'utilisabilité fournis par les auteurs [Matellanes *et al.*, 2005][Naaman *et al.*, 2004] soutiennent qu'une partie importante des informations qui aident les personnes à se souvenir d'un document multimédia fait référence au *contexte de création* de ce document («quand », « où », « avec qui »). Fréquemment, les utilisateurs annotent leurs documents multimédias à l'aide de mots-clés qui décrivent exactement ce *contexte de création* (c'est-à-dire, des informations sur l'utilisateur, sur la localisation, sur les conditions physiques lors de la création du document). Des approches d'annotation contextuelle automatique ont, ainsi, essayer d'inférer automatiquement ces métadonnées à partir des informations capturées lors de la création du document, telles que la date et l'heure de création et les coordonnées géographiques acquises par le biais de capteurs [Ames *et al.*, 2007] [O'Hare *et al.*, 2006].

Notre travail de thèse s'insère dans cette même perspective en proposant une évolution de ces propositions. En effet, bien qu'il existe aujourd'hui un nombre important d'approches de suggestion d'annotations et d'organisation reposant sur ces informations contextuelles, un vocabulaire formel standard de représentation de ces informations n'est pas encore disponible. De nombreux vocabulaires destinés au domaine de multimédia existent. Cependant, ils n'offrent quasiment pas de description du contexte de création de ces documents. Ces

[1] Un document multimédia personnel est un document multimédia (photo, vidéo, etc.) produit par les utilisateurs eux-mêmes, sans un objectif, *a priori*, professionnel [Lehikoinen *et al.*, 2007].

vocabulaires sont principalement destinés à la caractérisation du contenu et sont exploitées pour une annotation extensive et manuelle des documents.

L'exploitation de ces métadonnées est fréquemment segmentée dans ces approches. Soit elles sont utilisées pour l'annotation, soit pour l'organisation automatique des documents multimédias, soit pour leur partage. De plus, peu de ces métadonnées sont effectivement considérées dans la recherche par texte de ces documents.

1.1.2 Applications mobiles, multimédias et sensibles au contexte

Le succès des solutions de gestion sensible au contexte de documents multimédias est étroitement lié à l'acquisition de contenu et de métadonnées créés sur les dispositifs mobiles des utilisateurs. Cependant, ces systèmes incorporent les mécanismes de capture et de gestion de contexte directement dans leur code. Ceci rend difficile la flexibilité et l'extensibilité de ces approches, et rend également difficile la réutilisation de ces mécanismes [Baldauf *et al.*, 2007].

D'autres architectures (canevas, intergiciels) d'applications sensibles au contexte offrent des modules, plus génériques, d'acquisition de contexte qui assurent généralement les processus de capture, d'agrégation, d'interprétation et d'inférence de l'information contextuelle. Ces architectures proposent aussi certains mécanismes d'adaptation. Toutefois, ces canevas et intergiciels sont peu adaptés au développement d'applications sensibles au contexte pour les dispositifs mobiles. Elles s'avèrent très consommatrices de ressources (mémoire et batterie) plutôt limités sur les dispositifs mobiles. De plus, **ces architectures s'appuient sur la méthodologie de conception du tout-ou-rien, c'est-à-dire, soit le concepteur intègre toutes les classes de canevas/intergiciel dans le code de l'application, soit il n'utilise pas l'architecture** [Preuveneers *et al.*, 2007]. Or, les plates-formes de programmation d'applications pour les dispositifs mobiles sont des environnements de programmation très hétérogènes. Cette hétérogénéité des dispositifs mobiles peut être caractérisée par la différence entre les propriétés matérielles (la mémoire, la vitesse du processeur, les capteurs intégrées, etc.) et les propriétés logicielles (le système d'exploitation, les bibliothèques disponibles, la version des machines virtuelles) [Chu *et al.*, 2004]. Les applications multimédias et multi-capteurs sont les plus touchées par l'hétérogénéité, car elles accèdent à des fonctions et à des ressources qui existent sous diverses formes sur les dispositifs (par exemple, certains appareils sont équipés de GPS, d'autres non).

Ce support hétérogène rend plus difficile la conception d'applications susceptibles d'être activées par une grande diversité de dispositifs et exige des développeurs la construction de différentes versions de l'intergiciel et de l'application mobile pour chaque couple potentiel (modèle de dispositif, plate-forme). Ceci n'est clairement pas envisageable car le nombre de modèles de dispositifs croît constamment. En conséquence, **il y a un réel besoin d'outils pour automatiser le processus de conception d'applications mobiles et sensibles au contexte.**

De plus, les processus de modélisation, d'acquisition et d'enrichissement du contexte d'utilisation fournis par ces architectures sont conçus pour adapter des applications ou des services. La gestion de multimédia à l'aide du contexte n'est pas envisagée par ces approches.

Plusieurs défis scientifiques et technologiques doivent ainsi être relevés afin de permettre le développement de ces nouvelles applications mobiles, multimédias et sensibles au contexte. Parmi ces défis, nous pouvons énumérer :

(i) le développement d'une infrastructure de capture d'information contextuelle le plus indépendante possible d'un modèle spécifique de dispositif mobile;

(ii) la modélisation et la représentation du contexte pour une exploitation à la fois pour l'adaptation et pour la gestion de documents multimédias ;

(iii) la proposition de mécanismes d'inférence et d'interprétation afin d'enrichir et d'augmenter le niveau sémantique des informations contextuelles;

(iv) la proposition de méthodes d'exploitation du contexte qui profitent de la sémantique offerte par les métadonnées pour améliorer la gestion de documents multimédias.

1.2 Aperçu de la proposition

Nous cherchons ainsi dans cette thèse à proposer des solutions de représentation, capture et d'enrichissement sémantique des métadonnées, spécialement, celles décrivant le contexte de création d'un document multimédia. Notre objectif final est d'exploiter ces métadonnées produites dans **l'organisation, l'annotation, la recherche et le partage** de documents multimédias personnels**.**

Nous souhaitons également concevoir des solutions plus flexibles, notamment pour l'acquisition de contexte et la production de multimédias, activités qui sont réalisées sur les dispositifs mobiles. Nous proposons des mécanismes pour faciliter le développement et le déploiement des applications mobiles qui exploitent les informations contextuelles. Ces solutions doivent être conçues de manière à faciliter leur portabilité sur un grand nombre de modèles de dispositifs mobiles.

L'ensemble de nos solutions est intégré au sein d'une plate-forme nommé CoMMediA (de l'anglais *COntext-aware Mobile multiMEDIa Architecture*). La Figure 1 illustre les flux d'informations au sein de notre proposition.

Étape I - Initialement, un processus de déploiement adaptatif est mis en place afin de générer, puis installer une application mobile de création et/ou partage de multimédias sur le dispositif de l'utilisateur. Les services fournis par cette application sont automatiquement adaptés lors de l'installation conformément aux caractéristiques matérielles et logicielles du dispositif mobile de l'utilisateur (par exemple, un service d'enregistrement de vidéos est installé seulement si le dispositif fournit les bibliothèques nécessaires pour son exécution). Les caractéristiques du dispositif mobile sont découvertes automatiquement par une approche qui combine le modèle d'installation OTA (Over The Air) et une extension du registre de dispositifs mobiles WURFL. Cette application mobile accède aux capteurs et aux sources d'informations disponibles sur le dispositif mobile de l'utilisateur, chaque fois qu'un contenu multimédia est produit.

Étape II – L'acquisition initiale de contexte est elle aussi adaptée en fonction des propriétés du dispositif mobile et des services de l'application qui sont en cours d'exécution à cet instant précis. Après la production d'un document multimédia par l'utilisateur et l'acquisition du contexte de création par l'application, les métadonnées initialement obtenues sont représentées à l'aide d'un modèle d'annotations de multimédias : l'ontologie *ContextMultimedia*.

Étape III – Les métadonnées initiales sont soumises à un processus d'enrichissement dont l'objectif est d'augmenter le nombre et le niveau sémantique des informations concernant le document multimédia. L'idée principale est d'acquérir la quantité la plus large possible d'informations relatives au contexte de l'utilisateur lorsqu'il a créé un document multimédia (une photo, une vidéo, un texte, etc.). Le but majeur de cette étape est de réduire le temps consacré à l'annotation manuelle des multimédias, en dérivant automatiquement un ensemble d'annotations utiles. Ce processus est divisé en deux étapes : i) l'interprétation du contexte, et ii) l'inférence spatiale et sociale.

Lors de l'interprétation du contexte, la plate-forme CoMMediA accède à un ensemble de sources de données (gazetteer, services de météorologie, base de données de points d'intérêt, etc.), afin de transformer les données « symboliques » décrivant le contexte en des

informations de niveau sémantique plus élevé (telles que la transformation des coordonnées géographiques en le nom de l'endroit où se trouve l'utilisateur : sa maison, son travail, etc.). Ce processus de complétion d'information est conçu de façon à rendre plus facile l'ajout de nouvelles sources d'information. À chaque cycle d'interprétation, les nouvelles informations sont à nouveau représentées sous forme de concepts et de propriétés de l'ontologie *ContextMultimedia*.

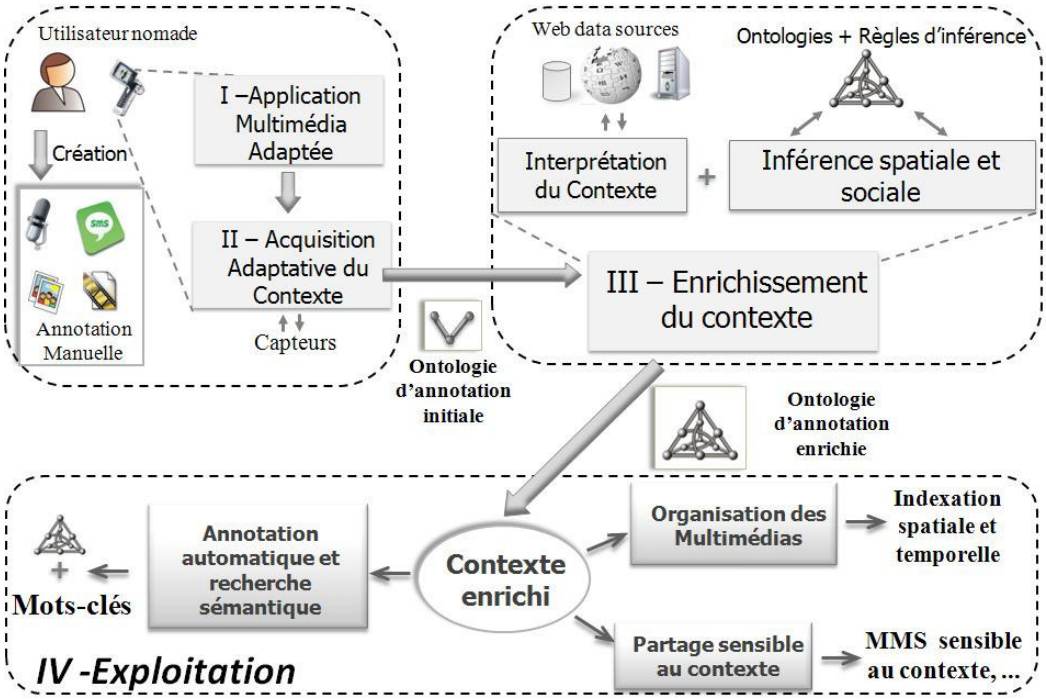


Figure 1 - Vue globale de la proposition.

La deuxième partie du processus d'enrichissement consiste en la mise en œuvre d'un ensemble de règles d'inférence dont l'objectif est de dériver de nouvelles informations sur les contextes sociaux et spatiaux de l'utilisateur lors de la création du document multimédia. À l'instar des approches d'adaptation, ce processus de raisonnement bénéficie de l'extensibilité offerte par la représentation déclarative de règles d'inférence. Le raisonnement est réalisé par un moteur d'inférence qui s'appuie sur les règles fournies par la plate-forme, et sur la formalisation des métadonnées décrites dans l'ontologie d'annotation pour déduire de nouveaux concepts et relations. Les règles d'inférence concernent principalement la découverte de relations sémantiques entre l'utilisateur et les personnes détectées à proximité (Y avait-il des personnes proches de l'utilisateur ? Ces personnes possèdent-elles des relations d'amitié, familiales ou professionnelles avec l'utilisateur ?). Ce système de règles est également employé pour le raisonnement spatial (relations de distance et topologiques).

Toutes les informations dérivées par le processus d'enrichissement sont ajoutées à l'ontologie d'annotation de départ. Ce fichier est le paramètre d'entrée des services d'exploitation de contexte de la plate-forme (étape IV de la Figure 1).

Les services **d'organisation de documents multimédias** bénéficient des informations spatiales (coordonnées géographiques, etc.) et temporelles (date et heure de création, etc.) pour la génération d'index spatiaux et temporels. Ces index sont exploités par d'autres services de la plate-forme lors de la navigation et de la visualisation de la collection des multimédias. Les services **d'annotation** exploitent l'ontologie pour la proposition de mots-clés qui peuvent être validés par l'utilisateur sur leur dispositif mobile ou sur une application de bureau de gestion de documents multimédias. La proposition de mots-clés est suivie d'un

processus de génération d'index sémantiques qui seront utilisés lors de la phase **d'interrogation** des collections de documents. De plus, les informations contextuelles sont la base de plusieurs services de **partage** proposés par la plate-forme.

1.3 Organisation du document

Ce document se divise en deux parties : *état de l'art* et *proposition*. Premièrement, nous présentons un état de l'art concernant la gestion de documents multimédias, le développement d'applications mobiles, la sensibilité au contexte, et l'adaptation d'applications mobiles. Une seconde partie du document est consacrée à la description de nos propositions de gestion sensible au contexte de documents multimédias personnels, et de déploiement adaptatif d'applications mobiles, multimédias et sensibles au contexte.

L'état de l'art est organisé en quatre chapitres :

- **Le chapitre 2 décrit l'évolution technologique des dispositifs mobiles**. Nous présentons un bref historique des dispositifs mobiles et un survol de ces nouveaux usages qui incluent l'accès au Web et la création de documents multimédias.
- **Le domaine de la gestion de documents multimédias est présenté dans le chapitre 3**. Nous décrivons la création et l'exploitation de multimédias dans la nouvelle génération des systèmes Web. Nous présentons également les approches existantes d'organisation automatique de documents multimédias et de représentation de métadonnées. Nous consacrons une partie du chapitre à la présentation des solutions de gestion sensible au contexte de documents multimédias.
- **Le chapitre 4 est dédié à l'informatique mobile**. Nous présentons dans ce chapitre une étude sur le développement et le déploiement d'applications mobiles. Nous souhaitons identifier les principaux défis pour la création d'applications mobiles qui accèdent à la fois aux capteurs, et aux fonctions multimédias de ces dispositifs.
- **Le chapitre 5 présente les domaines de la sensibilité au contexte et les approches d'adaptation d'applications mobiles**. Nous présentons dans ce chapitre des systèmes et des architectures sensibles au contexte. Notre objectif est de mieux comprendre comment un système sensible au contexte peut être conçu afin d'appliquer ces mêmes mécanismes pour le développement d'une architecture de gestion de documents multimédias personnels. Nous étudions également les approches existantes pour le déploiement adaptatif d'applications mobiles.

Nous organisons également notre contribution en quatre chapitres :

- **Le chapitre 6 présente nos modèles de représentation de l'information contextuelle, et un processus d'enrichissement de ces métadonnées**. Un modèle générique nommée *Context Top* est proposé pour la représentation du contexte visant une utilisation pour l'adaptation de services mobiles et pour l'annotation de multimédias. Ce modèle est ensuite étendu pour concevoir l'ontologie *ContextMultimedia* qui fournit un vocabulaire pour l'annotation d'un document multimédia, notamment les concepts caractérisant son contexte de création. Nous présentons également la plate-forme *ContextAnnotator* qui combine une composition de services Web et un mécanisme de raisonnement à base de règles SWRL pour élargir automatiquement les métadonnées relatives au contexte de création de documents. La modélisation de ce processus d'enrichissement est inspirée des architectures de gestion de contexte pour l'adaptation de Systèmes d'Information.

- **Le chapitre 7 présente nos propositions de déploiement adaptatif**. Nous présentons un ensemble de solutions pour l'adaptation d'applications mobiles et un intergiciel d'acquisition de contexte. À travers ces propositions, nous souhaitons sensiblement diminuer le besoin de développer différentes versions d'une application mobile sensible au contexte pour chaque couple potentiel (modèle de dispositif, plate-forme). Le chapitre décrit une approche de description d'applications mobiles, un gestionnaire de déploiement adaptatif, et une infrastructure à composants qui exécute les applications adaptées. Nous présentons également un intergiciel de capture de contexte, le DevAC, qui adapte dynamiquement la capture de contexte en fonction des services s'exécutant au dessus de lui.
- **L'architecture CoMMediA et des mécanismes d'exploitation de métadonnées contextuelles sont présentés dans le chapitre 8.** Nous présentons un mécanisme d'indexation sémantique qui est une extension du modèle vectoriel (*Vector Space Model*). L'objectif de ce mécanisme est de observer, lors de la phase d'indexation, les relations sémantiques entres les termes d'indexation et les documents multimédias. Cette extension assure la prise en compte de ces relations sémantiques lors de l'interrogation du corpus de documents. Nous proposons un mécanisme d'étiquetage des termes d'indexation et un modèle de requête qui utilise la segmentation de l'annotation des multimédias représentée à l'aide de *ContextMultimedia*. Un processus d'expansion sémantique est également ajouté à l'indexation afin d'éviter les problèmes de requêtes imprécises. Les connaissances exprimées à l'aide de l'ontologie *ContextMultimedia* sont exploitées sur la plate-forme CoMMediA, à la fois pour enrichir les messages partagés, et pour définir un mécanisme de notification de multimédias.
- **L'ensemble de nos solutions est validé dans le chapitre 9**. Nous présentons le système PhotoMap construit à l'aide de la plate-forme CoMMediA. Deux expérimentations réalisées sont présentées dans ce chapitre. La première décrit des tests de mesure de l'efficacité de notre mécanisme de recherche sémantique dans un corpus d'images géoréférencées. La deuxième expérimentation illustre l'impact de notre processus d'adaptation sur le temps d'installation d'une application mobile particulière.

Enfin, **le chapitre 10** conclut cette thèse, en présentant notre bilan et les perspectives pour ce travail.

ÉTAT DE L’ART

2 L'EVOLUTION ET LES NOUVEAUX USAGES DES DISPOSITIFS MOBILES

Au lancement de l'assistant personnel Newton, en 1993, par l'entreprise Apple[2], rares sont ceux qui imaginent l'impact économique et social qu'aura plus tard l'introduction de dispositifs mobiles sur la vie quotidienne. Le nombre de téléphones mobiles utilisés dans le monde a déjà dépassé les 4 milliards en 2008 (selon l'ITU - *International Telecommunication Union*[3]), ce qui représente environ 60% de la population mondiale. La Figure 2 expose la progression, quasiment exponentielle, du nombre de clients d'opérateurs mobiles entre 2000 et 2008. Le pourcentage d'abonnements est encore plus significatif en France.

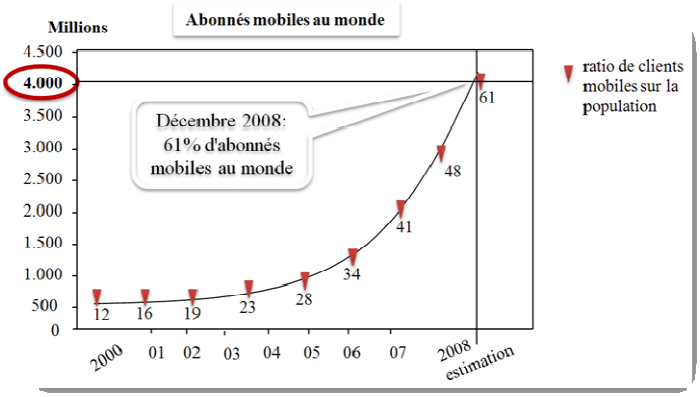


Figure 2 - Évolution du nombre de clients des opérateurs mobiles. Source : *ITU*[4].

Selon l'ARCEP (Autorité de Régulation des Communications Électroniques et des Postes)[5], on compte aujourd'hui plus de 58 millions d'abonnements, ce qui correspond à

[2] http://www.apple.com/

[3] http://www.itu.int/newsroom/press_releases/2008/29.html

[4] ITU-*International Telecommunication Union* - http://www.itu.int/newsroom/press_releases/2008/29.html

[5] http://www.arcep.fr/index.php?id=35

environ 90% de la population française (source mars 2009). Les dispositifs mobiles, notamment les téléphones mobiles, sont les machines dotées de capacité de calcul le plus utilisées au monde [Raento *et al.*, 2005]. L'essor des technologies mobiles (en termes de capacités des dispositifs et de réseaux) a fait évoluer également les usages de ces dispositifs depuis une simple communication vocale, jusqu'à la navigation sur le Web.

Le terme « dispositif mobile » désigne généralement les appareils électroniques dotés de capacités informatiques. Ces appareils présentent un certain niveau de transportabilité, grâce à un poids et une taille réduits. De plus, ils disposent d'une autonomie d'utilisation assurée par leur propre source d'énergie (telles qu'une batterie ou une pile) [Chen *et al.*, 2000][Barnes *et al.*, 2002]. Nous pouvons classer les dispositifs par catégories : les ordinateurs portables et ultraportables, les Tablet PC, les *e-books* (livres électroniques), les assistants personnels (en anglais, *Personal Digital Assistant*), les baladeurs numériques, les consoles de jeux portables (Nintendo DS, PSP, etc.), les navigateurs GPS et les téléphones portables. Dans la Figure 3, nous rapportons une classification de ces dispositifs en fonction de leur portabilité proposée dans [Ganneau, 2009]. Dans le cadre de cette thèse, nous nous intéressons principalement aux dispositifs interactifs de petites tailles, tels que les assistants personnels et les téléphones mobiles, qui offrent à leurs usagers une portabilité maximale et la possibilité d'activer des applications en situation de déplacement (par exemple, lors d'un circuit touristique dans une ville). Nous présentons ensuite un rapide historique de ces dispositifs mobiles et de l'évolution de leurs usages.

Figure 3 - Illustration de la classification de dispositifs mobiles selon leur portabilité [Ganneau, 2009].

2.1 L'évolution des assistants personnels

Les premiers dispositifs mobiles de « petite taille » ont vu le jour au cours des années 80. Les organiseurs numériques de la marque Psion[6] ou encore les calculatrices étendues produites par Casio[7] et Hewlett-Packard[8] offrent alors, sur un même dispositif, des fonctionnalités basiques bureautiques (agenda, calculatrice, liste de contacts, etc.). Par exemple, le dispositif CASIO IF-8000 lancé en 1987 dispose déjà d'un écran tactile noir et

[6] http://www.psionteklogix.com/

[7] http://www.calctech.org/

[8] http://www.hp.com/

blanc qui permet l'interaction avec les applications à l'aide d'un stylet [Cdec, 2009]. Les capacités de calcul de ces dispositifs mobiles demeurent limitées à une dizaine de Mhz et la mémoire disponible pour l'exécution des applications est à l'époque inférieure à un 1Mb. Un aperçu des dispositifs mobiles des années 80 et la description de leurs caractéristiques matérielles sont disponibles sur le site du *Pocket Museum*[9] .

En 1993, le terme assistant personnel numérique (en anglais, Personal Digital Assistant – PDA) est utilisé la première fois lors du lancement du dispositif Newton Message Pad par Apple. Ce dispositif est doté de trois caractéristiques remarquables pour l'époque : un grand écran tactile, la synchronisation avec un ordinateur, et l'installation d'applications tierces. La majorité des modèles de PDA qui vont suivre reprendront ces trois caractéristiques. La Figure 4 présente le modèle Apple Newton Message Pad 100 à côté du dernier dispositif mobile fabriqué par Apple : l'iPhone. L'écran tactile du Newton prend quasiment toute la place du dispositif et le clavier est remplacé par un clavier virtuel qui est affiché quand l'utilisateur touche l'écran avec le stylet. Le Newton offre la synchronisation des données de ses applications (agenda, contacts, etc.) avec une application desktop. L'objectif est de permettre une sauvegarde des informations sur l'ordinateur et de proposer une éventuelle saisie d'informations à partir du desktop. La connexion entre l'ordinateur et le dispositif est assurée par le biais d'un câble serie ou par l'infrarouge (protocole IrDA). L'Apple Newton dispose également d'un système d'exploitation : le Newton OS. Il permet la mise à jour des applications déjà existantes et l'installation de nouvelles applications développées en Newton Script (une plate-forme de programmation C++).

Figure 4 - Deux dispositifs mobiles conçus par Apple : l'Apple Newton (1993) et l'iPhone (version 2007). Source de la figure : www.engadget.com.

Au cours des années 90, les avancés techniques permettent de réduire progressivement la taille et le poids des assistants personnels, parfois renommés ordinateurs de poche. De plus, ces dispositifs vont voir l'accroissement de leurs capacités de calcul et de mémoire, ainsi que de leur temps d'autonomie. Les ordinateurs de poche fabriqués à cette époque par Palm (les *Palmtops*) et par Ipaq (les *Pocket PC*) proposent beaucoup de nouvelles fonctionnalités (l'écran en couleur, la reconnaissance de l'écriture en langage Graffiti, le support à la lecture de documents Microsoft Office, etc.). En conséquence, l'intérêt suscité pour les dispositifs mobiles pour une utilisation à titre personnel ne fait que croître tout au long des années 1990. L'augmentation des capacités de calcul et principalement de stockage, encourage également

[9] http://pocket.free.fr/index_f.html

le développement d'applications mobiles industrielles, destinées à des entreprises dont les employés sont mobiles (des entreprises de ventes à domicile, des compagnies de distribution et maintenance, etc.) [Hazan, 2008]. Le principe de ces applications est de charger sur l'ordinateur de poche les données nécessaires (telles que les prix des produits) avant le départ de l'utilisateur nomade sur le terrain. Une fois sur le lieu, l'utilisateur consulte les données pré-chargées et ajoute de nouvelles informations (tel que l'enregistrement d'une vente). Une fois de retour au sein de la société, les données saisies par l'utilisateur sur le terrain sont transférées à un ordinateur, à l'aide d'une fonction de synchronisation. Une application *desktop* lit les nouveaux registres et les insère dans le système d'information de la société [Viana *et al.*, 2004].

2.2 L'histoire du téléphone mobile

Parallèlement aux avancées techniques des assistants personnels, l'adoption de la téléphonie mobile connaît aussi un essor certain. La première génération de réseaux commerciaux de téléphonie sans fils émerge ainsi dans les années 1980 au Japon (NTT), aux États-Unis (AMPS), et plus tardivement en Europe (TACS/ETACS) [Adachi, 2001]. Une ville est alors divisée en cellules de couverture, chacune assurée par une antenne différente. Le téléphone mobile, à l'époque appelée téléphone cellulaire (en anglais, *cellphone*), communique directement avec l'antenne par un canal de fréquence radio. L'antenne garantit la connexion au central téléphonique en utilisant une infrastructure câblée conventionnelle. Au départ, les téléphones mobiles sont uniquement disponibles dans les voitures à cause de la taille et du poids de leurs batteries et de leurs récepteurs/émetteurs radios. Le modèle DynaTAC de Motorola reçoit la première licence d'opération pour un téléphone vraiment mobile (750 g et 25 cm de taille) en 1983.

À l'instar des assistants personnels, le poids, la taille et le prix des téléphones mobiles vont être réduits tout au long des années 1980 et 1990. Initialement, exclusivement dédiés à la communication vocale, le téléphone gagne de plus en plus en fonctionnalités et en popularité. Parmi les fonctionnalités au succès escompté, nous pouvons citer : i) l'envoi de messages de texte courts (le service SMS proposé par le système de deuxième génération GSM à partir de 1993), ii) l'accès à l'internet en utilisant le protocole WAP (*Wireless Application Protocol*) en 1999, et iii) la prise de photos à l'aide d'un appareil photo embarqué (lancement du modèle SHARP J-SH04 en 2001).

La fin des 1990 connaît une explosion de la téléphonie mobile dans les pays développés. Au début de l'année 2000, par exemple, 50% de la population japonaise a déjà un téléphone portable [Adachi, 2001]. Cette adoption massive couplée à l'évolution matérielle de ces dispositifs dope le développement d'applications destinées exclusivement aux téléphones mobiles. Au départ, les premiers téléphones portables sont des systèmes fermés où les applications mobiles disponibles demeurent limitées à celles développées/distribuées par les constructeurs de dispositifs. L'arrivée des systèmes d'exploitation (Symbian, Wince, etc.) et la mise à disposition de machines virtuelles (J2ME, Brew, etc.) permet l'installation d'applications tierces sur ces dispositifs. Ces applications sont alors développées à l'aide de kits de développement fournis par les constructeurs de téléphones portables (tels que le Nokia Series 40 SDK). Les jeux et les services qui exploitent le protocole SMS sont les applications les plus prisées à cette période [Harmer, 2003].

2.3 La convergence

En 2000, la série Nokia Communicator marque le début de la convergence tant annoncée auparavant entre les assistants personnels et les téléphones mobiles. Le premier Nokia Communicator possède comme principales caractéristiques : un écran plus grand, la possibilité d'accès à l'internet, la synchronisation des données avec un ordinateur, et l'intégration des applications présentes sur les assistants personnels (lecteur PDF, email, agenda, etc.). Ces caractéristiques créent la tendance suivie par les modèles suivants. La Figure 5 illustre l'évolution des dispositifs mobiles et la convergence vers des téléphones plus puissants, parfois nommés « smartphones ».

Quelques années plus tard, les constructeurs d'assistants personnels (Hewlett-Packard, Palm, etc.) proposent également des PDA qui font office de téléphone (par exemple, le Treo 600), et possèdent ainsi un mini-clavier QWERTY et un écran tactile (avec l'utilisation d'un stylet tactile). La possibilité de transmettre des données de presque n'importe quel endroit (GPRS en 2003, et EDGE par la suite) bouleverse le schéma classique de synchronisation des applications mobiles industrielles. À l'aide d'un PDA comme le Treo, un utilisateur nomade peut, depuis le terrain, consulter et ajouter des données au système d'information d'une entreprise. Ceci permet une transmission de données quasiment ubiquitaire et l'utilisation quotidienne d'un seul dispositif mobile. Ces constats poussent progressivement les fabricants de PDA à arrêter la production d'assistants personnels qui ne fournissent pas la fonction téléphone. Par exemple, la production du dernier modèle de Palm de ce genre, le Lifedrive, est arrêtéE en 2007. Les fabricants de PDA privilégient alors la production de *smartphones* plus petits que les PDA d'avant, mais également plus puissants et multifonctionnels (tels que le Palm Pre illustré sur la Figure 5).

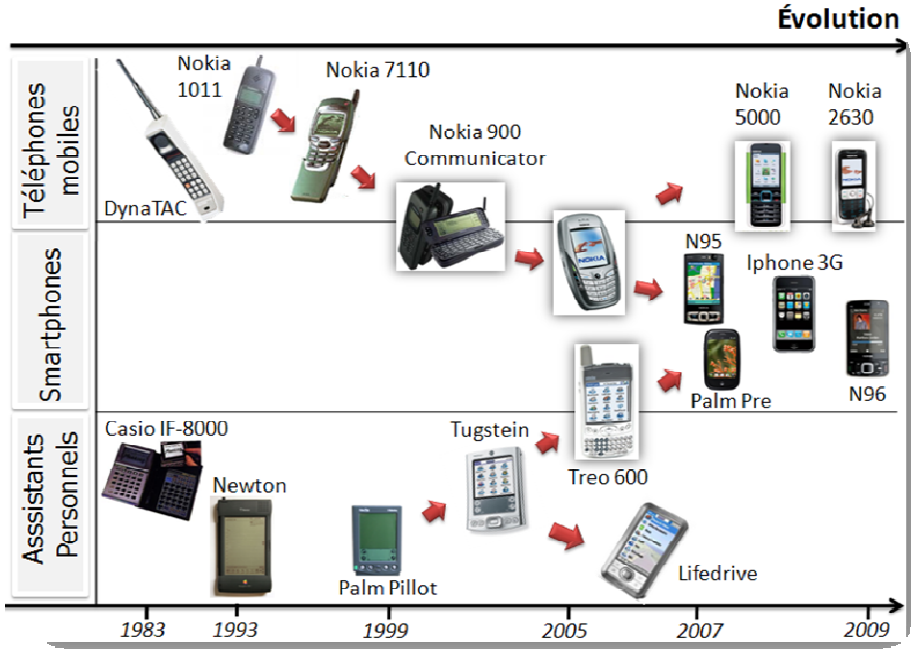


Figure 5 - Évolution de dispositifs mobiles ver la convergence aux smartphones.

À la fin des années 2000, les frontières entre les téléphones mobiles et les smartphones sont devenues très floues. En réalité, les dispositifs d'aujourd'hui sont de véritables « couteaux suisses » modernes. Ils peuvent concentrer dans un même équipement : plusieurs

technologies de communication (Bluetooth, Wi-Fi, 3G, etc.) et diverses fonctions multimédias (baladeur MP3, appareil photo et vidéo, dictaphone, audio 3D, etc.). L'exemple le plus remarquable parmi ces dispositifs est l'iPhone lancé par Apple en 2007, vendu à plus de 17 millions d'unités. Le *design* de l'iPhone supprime quasiment tous les boutons de l'appareil pour que tout soit contrôlable depuis l'écran tactile multipoints sans avoir besoin d'un stylet pour toucher le téléphone. Ceci a apporté une meilleure convivialité à l'interaction avec les applications mobiles, notamment avec le navigateur Web [Ganneau, 2009]. Cette nouvelle fonctionnalité commence aujourd'hui à être reprise par les derniers modèles (HTC Touch 3G, LG KS360, etc.). Toute la puissance de l'iPhone est perceptible en observant ses caractéristiques matérielles, particulièrement son pouvoir de stockage qui s'exprime en giga-octets. Le Tableau 1 présente une comparaison entre les caractéristiques du premier dispositif mobile conçu par Apple, le Newton message Pad et la version 3G de l'iPhone.

Outre l'adoption d'écrans tactiles multipoints, un dernier fait marquant de l'évolution des dispositifs mobiles est l'introduction du GPS. En 2007, l'intégration d'un récepteur GPS au Nokia N95 a ouvert la voie au développement de services localisés plus précis sur les *smartphones* (des cartes, le guidage). Nous pouvons voir sur le Tableau 1 que cette caractéristique a déjà été reprise sur le dernier modèle de l'iPhone.

Dispositif Mobile	Apple Newton Message Pad 100	iPhone 3G
Année d'introduction	1993	2008
Mémoire vivre	640KB	128MB
Mémoire de stockage	4MB	16GB
Connectivité	Infrarouge, par câble, extensions PCMCIA	USB, Wi-Fi (802.11 b/g), 3G, Bluetooth 2.0
Écran	336x240 (noir et blanc)	480x320 (64 millions de couleurs)
Processeur	ARM 20MHz	ARM 620MHz
Système d'exploitation	Newton OS	iPhone OS
Capteurs intégrés	_	GPS assisté, appareil photo, accéléromètre, capteur de proximité et lumière ambiante
Plate-forme de programmation	NewtonScript (C++)	Plates-formes Web, iPhone SDK
Dimensions du dispositif	18.42 cm x 11.43 cm x 1.91 cm	11.5 cm x 6.1 cm x 1.1 cm
Poids	0. 410 Kg	0.135 Kg
Entrée de données	Écran sensible au stylet tactile	Écran tactile multipoint

Tableau 1 - Comparaison des dispositifs Newton Message Pad 100 et l'iPhone 3G.

2.4 Les usages d'aujourd'hui et les perspectives pour demain

De nos jours, la communication vocale et l'envoi de SMS restent les fonctionnalités les plus utilisées sur les téléphones mobiles. Cependant, l'utilisation des fonctions multimédias progresse sensiblement. Les téléphones, même les moins chers du marché, disposent de plus en plus d'accessoires multimédias. La plupart de ces dispositifs possèdent des appareils photos numériques intégrés et font également office de baladeur MP3 et de console de jeux. Ces caractéristiques étaient présentes peu de temps avant uniquement sur les *smartphones* les plus évolués. En conséquence, l'utilisation de ces trois fonctions multimédias se généralise, surtout chez les plus jeunes [Ganneau, 2009], notamment avec l'usage de l'appareil photo intégré. Le téléphone mobile est utilisé pour prendre des photos occasionnelles ou même pour

remplacer l'appareil photo numérique classique [Sarvas *et al.*, 2005]. Par exemple, certains utilisateurs avouent n'emporter que leur téléphone pour un voyage touristique [Ames *et al.*, 2007]. Le site Flickr[10] de partage de photos donne des indices de cette évolution : le nombre de photos prises en utilisant les modèles iPhone et Nokia N95 dépasse les 14 millions.

Dès le lancement du Wap (*Wireless Application Protocol*) en 1999, l'accès au Web est possible sur les téléphones mobiles. Actuellement, rares sont les dispositifs qui ne possèdent pas un navigateur Wap ou même XHTML. Néanmoins, le Web mobile n'a pas encore eu le succès escompté. Les causes de cette non-réussite ont été principalement : l'écran réduit des premiers modèles de téléphones, la vitesse pas assez performante des réseaux sans fils, et le coût élevé des tarifs de transmission de données pratiqués par les opérateurs mobiles [Keshav *et al.*, 2005] [Saha *et al.*, 2001] . Dernièrement, cette tendance commence à être renversée. Des études montrent qu'au Japon, le nombre d'accès mobiles à l'internet est déjà quasiment équivalent à l'accès par ordinateur [ComScore, 2007]. Dans d'autres régions du monde, les débits des réseaux de troisième génération (UMTS, WCDMA, etc.) surpassent les 2 Mbit/s. Malgré les tarifs encore élevés, ces débits sont similaires à ceux de réseaux ADSL et offrent la possibilité d'un accès satisfaisant aux pages Web. Au delà de la crainte de dépenser de l'argent, le manque de connaissance des vraies capacités des navigateurs Web des dispositifs mobiles (vitesse, utilisabilité, etc.) freinent également l'accès à l'internet. Néanmoins, l'intégration des réseaux Wi-Fi sur certains terminaux mobiles commence à inciter les personnes à découvrir l'internet mobile sans imposer un coût additionnel [Cui *et al.*, 2008].

Aujourd'hui, les utilisateurs nomades naviguent sur le Web principalement lorsqu'ils sont dans un environnement clos (bâtiment, restaurant, train, etc.). La plupart du temps, les utilisateurs restent dans une position statique lors d'une navigation Web en raison de l'attention visuelle exigée pendant l'accès [Lee *et al.*, 2005]. Les navigateurs mobiles les plus avancés peuvent afficher correctement la majorité des pages Web existantes. Cependant, les sites spécialement construits pour l'accès mobile sont préférés par les utilisateurs. Les sites de webmail mobile et de lecture de flux de données (en anglais, « news feeds ») sont les plus accédés [Cui *et al.*, 2008]. Une autre catégorie d'applications Web en plein essor est celle des « widgets » mobiles. Ces applications s'exécutent à l'extérieur d'un navigateur et interagissent directement avec un service Web dédié à l'accès mobile. Les sites Web les plus répandus tels que Youtube, Amazon, Google Maps, Ebay, Flickr proposent tous des widgets mobiles à leurs utilisateurs. Cependant, ces widgets fonctionnent uniquement sur certains systèmes d'exploitation mobiles. La Figure 6 illustre un exemple d'un widget mobile développé pour l'iPhone[11]. Ce récent boom du développement des widgets mobiles est étroitement lié au succès de l'iPhone et à l'amélioration de l'interactivité proportionnée par son écran tactile.

[10] http://www.flickr.com/cameras/

[11] http://www.softpauer.com/f1app/index.html

Figure 6 - Widget mobile pour accompagner en temps réel les positions des voitures pendant une course de Formule 1.

L'évolution des dispositifs mobiles présentée tout au long de cette section montre que fréquemment les caractéristiques des derniers modèles de dispositifs sont au fur et à mesure reprises par les modèles suivants et, en cas de succès avéré, deviennent même standard (par exemple, l'écran en couleur, l'appareil photo, le SMS, l'accès au Web, le Bluetooth). Ainsi, si l'écran tactile suit cette logique et si les tarifs d'accès baissent, le Web mobile deviendra incontournable. Un autre exemple de perspective à moyen terme est l'intégration progressive de capteurs aux dispositifs mobiles. Par exemple, l'iPhone 3G dispose déjà d'un accéléromètre et d'un GPS intégré. La Figure 7 illustre un prototype construit par le laboratoire de Nokia au Japon suréquipé de capteurs [Yamaba *et al.*, 2005]. Avec la transformation ces dispositifs en véritables studios mobiles, les capteurs pourront offrir des informations suffisamment riches pour décrire les documents multimédias produits par les utilisateurs nomades (photo, vidéo, audio, etc.). La possibilité d'aller vers des dispositifs mobiles multifonctionnels, multi-capteurs et connectables sur le Web est étudiée et exploitée tout au long de cette thèse.

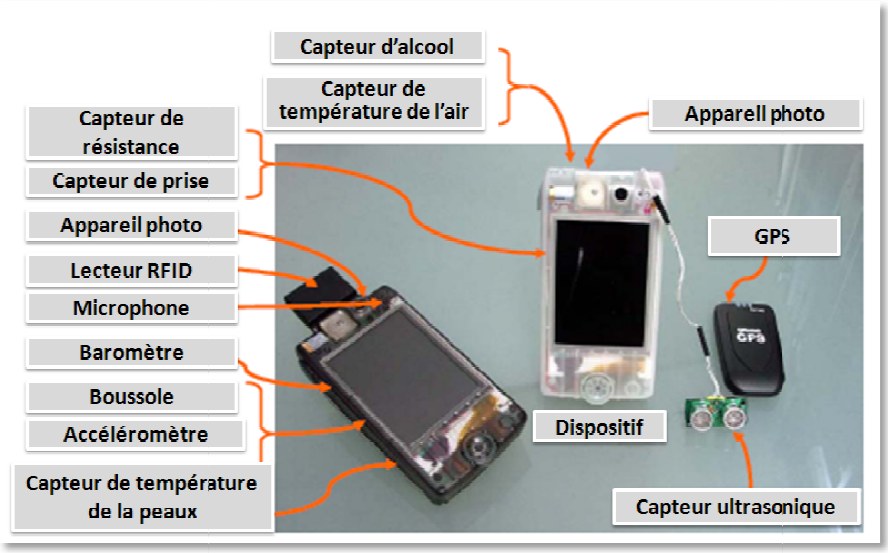


Figure 7 - Les téléphones mobiles suréquipés de capteurs. Source de la figure : Citron [Yamaba *et al.*, 2005].

3 GESTION DE DOCUMENTS MULTIMEDIAS PERSONNELS

Dans ce chapitre nous présentons la notion de *document multimédia personnel* et les nouveaux usages de ces documents dans les environnements Web 2.0. Nous passons également en revue les technologies d'organisation, d'annotation et de partage de documents multimédias. Nous nous intéressons particulièrement aux approches qui s'appuient sur l'utilisation de dispositifs mobiles.

3.1 Le document multimédia personnel et ses usages

Un document numérique désigne un ensemble de données organisées selon un certain nombre de règles et exploitables par un ordinateur [Roisin, 1999]. Le terme *document multimédia* a été initialement utilisé lorsqu'un document numérique regroupait divers moyens de diffusion. L'utilisation simultanée du son et du visuel sur une vidéo ou le regroupement d'images et textes sur un même document SMIL[12](*Synchronized Multimedia Integration Language*), sont des exemples de documents multimédias selon cette définition [Hardman *et al.*, 1993]. Aujourd'hui, l'acception du terme est désormais plus large. On utilise le mot « multimédia » pour désigner toute application utilisant ou servant à travailler sur au moins un média spécifique tel qu'une image, une vidéo, un texte ou un fichier audio. Ainsi, un document multimédia peut faire référence à un document numérique contenant un ou plusieurs de ces médias.

La création et l'édition d'un document multimédia possèdent diverses finalités allant d'une simple photo pour enregistrer un moment en famille à la création d'une superproduction hollywoodienne. Cet usage ultime du document multimédia guide la construction des outils destinés à son organisation et influe sur les besoins en termes de mécanismes de recherche. Dans cette thèse, nous étudions plus particulièrement les documents multimédias personnels, c'est-à-dire les documents produits par les utilisateurs eux-mêmes, sans un objectif, *a priori*, professionnel [Lehikoinen *et al.*, 2007]. La diversité sémantique des multimédias personnels est généralement beaucoup plus importante que sur un groupe de fichiers multimédias d'un domaine d'application spécifique (par exemple, les images des tableaux du Musée du Louvre) [Shevade *et al.*, 2007]. Cette caractéristique des documents multimédias personnels influence

[12] http://www.w3.org/AudioVideo/

également la modélisation des outils de gestion de ces multimédias. Nous examinons dans la suite comment les documents multimédias personnels sont crées, stockés et organisés par les utilisateurs. Nous présentons les approches de gestion de documents multimédias qui tentent d'automatiser les processus d'organisation, et de faciliter la récupération et la navigation au sein d'une large collection de documents.

3.1.1 Les nouveaux comportements dans la création de documents multimédias personnels

Avant la création du stockage numérique des multimédias, les personnes hésitaient beaucoup plus au moment d'enregistrer une vidéo ou de prendre une photo à tout moment. Les coûts des films argentiques, des cassettes d'enregistrement ou du développement des photos étaient des paramètres souvent pris en compte. La banalisation des appareils photos numériques, depuis la fin des années 1990, a bouleversé ce comportement lors de la création des contenus multimédias. Ces appareils permettent de prendre des photos et d'enregistrer de vidéos directement sur un format numérique. Leur simplicité d'usage et la baisse du coût de stockage des données provoquent la croissance quasi exponentielle de la quantité de photos et de vidéos numériques sur nos ordinateurs personnels [Monaghan *et al.*, 2006]. Si le coût de la création et du stockage ne pose plus de problèmes majeurs, la grande quantité de documents multimédias rend les processus d'**organisation**, de **recherche** et de **visualisation** plus ardus [Matellanes *et al.*,2005] [Monaghan *et al.*, 2006].

Les nouveaux modèles de téléphones mobiles contribuent également à l'augmentation du nombre des multimédias personnels [Ames *et al.*, 2007]. La qualité des images, des vidéos et des fichiers audio produites par ces dispositifs est en constante amélioration. Par conséquent, ces fonctionnalités sont de plus en plus utilisées. La Figure 8 illustre la progression du nombre des dispositifs mobiles dotés d'un appareil photo numérique. Le nombre de ces terminaux mobiles est déjà supérieur à la quantité d'appareils numériques conventionnels.

Plus qu'une alternative à ces appareils traditionnels, les dispositifs mobiles sont à portée de leurs utilisateurs quotidiennement, et offrent ainsi la possibilité de créer des contenus multimédias à tout moment et d'enregistrer des scènes inespérées [House et *al.*, 2006] [Sarvas *et al.*, 2005] .

Les dispositifs mobiles étendent et modifient le comportement de création de multimédias. Avant leur apparition, les photos et les vidéos personnelles étaient plutôt destinées à enregistrer un événement important (vacances, jours de fête, ..) [Matellanes *et al.*, 2005][Naaman *et al.*, 2004]. Aujourd'hui, à l'aide des dispositifs mobiles, les personnes enregistrent des photos et des vidéos sur leurs vies mondaines (par exemple, pour la construction d'un journal intime, des blogs, en images et vidéos) [House et *al.*, 2006]. Ces multimédias sont aussi utilisées pour la communication entre individus et pour l'auto-expression (tels que les photos humoristiques et artistiques de soi-même) [Ames *et al.*, 2007] [House et *al.*, 2006]. De fait, les services de visionnement de photos et de vidéos, le MMS (envoi de photos, de vidéos et de textes à d'autres téléphones), ainsi que l'accès à l'internet offert par les dispositifs mobiles, contribuent à ce changement de comportement des utilisateurs pour lesquels l'action de partager le contenu est parfois plus importante que la qualité du multimédia elle-même.

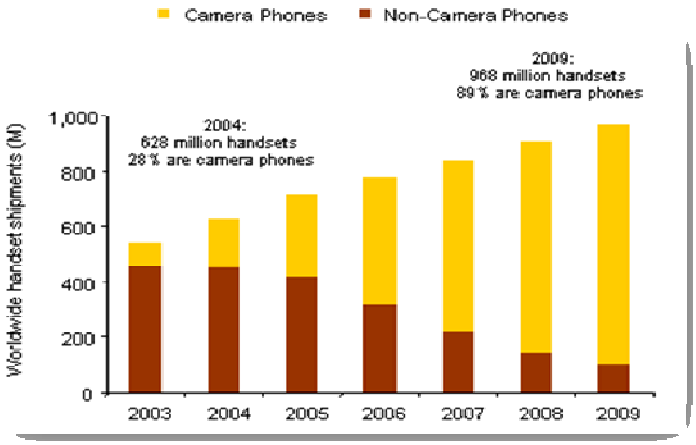


Figure 8 - Comparaison, fournie par InfoTrend[13], du nombre de téléphones mobiles dotés d'appareils photos numériques, et le nombre d'appareils conventionnels.

Un autre phénomène responsable de ce changement est le succès des sites Web de médias sociaux (en anglais, *Social Media* ou *Social Web* [Gruber, 2007]). Le Web 2.0, la nouvelle génération du Web, est en marche depuis quelques années , et le Web social est un aspect majeur [O'Reilly, 2005]. Le Web est devenu une plate-forme de services plus interactifs grâce à des technologies telles que CSS, AJAX, et RSS. Ces services sont aussi réutilisables plus aisément à travers des mécanismes de *mashup*[14] (par exemple, la bibliothèque Google Maps Api). Cependant, la caractéristique la plus remarquable de ce nouveau Web est l'intense contribution et collaboration des utilisateurs qui sont passés de consommateurs passifs d'informations à des producteurs et évaluateurs de contenu [Hendler, 2008]. Cette contribution collective est illustrée par les sites de médias sociaux. Elle est effective lorsque des personnes écrivent gratuitement des articles sur Wikipedia, partagent des photos sur Flickr et évaluent la qualité d'un hôtel sur Booking.com [Gruber, 2007].

Le **partage**, l'**annotation,** les évaluations, et les **réseaux sociaux** sont les piliers des sites de médias sociaux. En effet, un nombre important de photos et de vidéos produites par des particuliers est partagé sur le Web sur des systèmes tels que Yahoo! Flickr, Twitter[15], TripperMap, Facebook, Daily Motion et Youtube. La plupart de ces systèmes du Web 2.0 fournissent aux utilisateurs des interfaces pour décrire les contenus partagés en utilisant des mots-clés (en anglais, *tag*). L'objectif principal de cette annotation est de permettre une classification du contenu afin de faciliter ultérieurement la navigation et la recherche de ce contenu à un autre utilisateur [Ames *et al.*, 2007]. Les utilisateurs peuvent également attribuer des notes à chaque contenu visionné. Ces évaluations sont exploitées lors du classement du résultat d'une recherche.

Une partie de ses systèmes de médias sociaux (tels que Twitter, Flickr, Facebook, Skyrock[16]) repose sur des réseaux sociaux définis par les utilisateurs. Chaque utilisateur possède un profil virtuel qui peut être relié à des profils d'autres utilisateurs du système. Ces liens représentent « l'amitié » dans le monde virtuel et sont exploités dans la construction de

[13] http://www.infotrends.com/public/Content/INFOSTATS/Articles/2005/03.29.2005.html

[14] Le principe du mashup Web est la réutilisation des contenus ou des services provenant de d'autres applications Web pour créer un site ou un service nouveau. Un exemple est l'usage des cartes fournis par Google Maps pour l'affichage de contenu géoréférencé d'une entreprise.

[15] http://www.twitter.com

[16] http://www.skyrock.com/blog/

communautés virtuelles et dans la définition de droits d'accès à un contenu. Par exemple, une photo d'un utilisateur publiée sur Facebook est accessible seulement aux membres de son réseau social.

La Figure 9 illustre un de ces sites de partage de contenu : le Yahoo Flickr ! Les utilisateurs, en plus de publier leurs photos et vidéos, peuvent étiqueter leurs documents multimédias avec des informations spatiales telles que des coordonnées géographiques et la description du lieu de la prise de vue. La Figure 9 présente le résultat de la recherche des photos prises dans la ville de Paris (plus de 706.539 de photos géoréférencées à la fin de 2008).

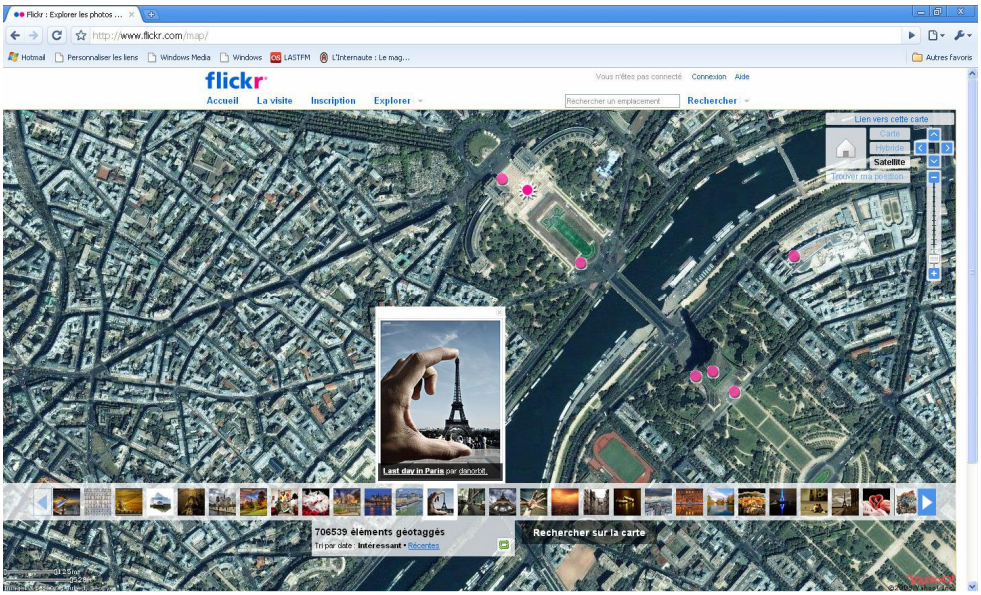


Figure 9 - Une vue satellitaire des photos prises à Paris disponibles sur le site Web 2.0 de partage de photos et vidéos Flickr.

3.1.2 Les outils desktop d'aide à la gestion de multimédias personnels

Après la création des multimédias sur leurs téléphones mobiles ou sur leurs appareils numériques conventionnels, les utilisateurs transfèrent ces multimédias, spécialement les photos et les vidéos, vers leurs ordinateurs de bureau. Afin de mieux les organiser, la majorité des utilisateurs modifie le nom des documents pour aider les recherches futures. Ils rangent leurs multimédias dans des répertoires hiérarchiques qui symbolisent des périodes (par exemple, été 2006), des événements (un voyage, une conférence) et des endroits différents (photos du Mont Blanc) [Matellanes *et al.*,2005][Naaman *et al.*, 2004]. Les personnes créent également des catégories de contenus directement liées aux sujets concernés par les documents (par exemple, des vidéos d'animaux). Néanmoins, ce type d'organisation rudimentaire n'évite pas le problème de la recherche lorsque le nombre de documents multimédias augmente de manière conséquente. Il est souvent difficile de trouver une photo ou une vidéo spécifique lorsqu'on veut la montrer ou la partager avec une autre personne.

Plusieurs logiciels de gestion de documents multimédias personnels proposent des mécanismes d'organisation de ces fichiers afin d'en faciliter la recherche et la visualisation. Le prototype proposé par Nars [Nars, 2005] et les outils commerciaux Apple Iphoto, Picasa, Adobe Photoshop Album et ACDSee sont les plus remarquables. Ces outils exploitent tous les informations sur la date de création des multimédias pour établir un ordre chronologique et génèrent des regroupements temporels afin de rendre plus facile la consultation des documents. Les principales fonctionnalités présentes sur ces outils sont : i) la création

d'albums thématiques (des collections de multimédias) ; ii) l'ajout de mots-clés pour l'indexation ; iii) des méthodes simples de traitement d'image et iv) l'intégration avec des sites Web de partage de contenus. Par exemple, la Figure 10 illustre l'ajout manuel d'un mot-clé (« conférence ») à une photo sélectionnée sur le logiciel Picasa. Cette application de gestion de multimédias est fournie gratuitement par Google. Lors d'une recherche par mots-clés, Picasa permet d'afficher, par exemple, seulement les images contenant des visages. Picasa offre également la possibilité de géoréférencer **manuellement** les documents multimédias à l'aide de Google Earth et de les rendre disponibles sur le Web à travers le site Picasa Web Album.

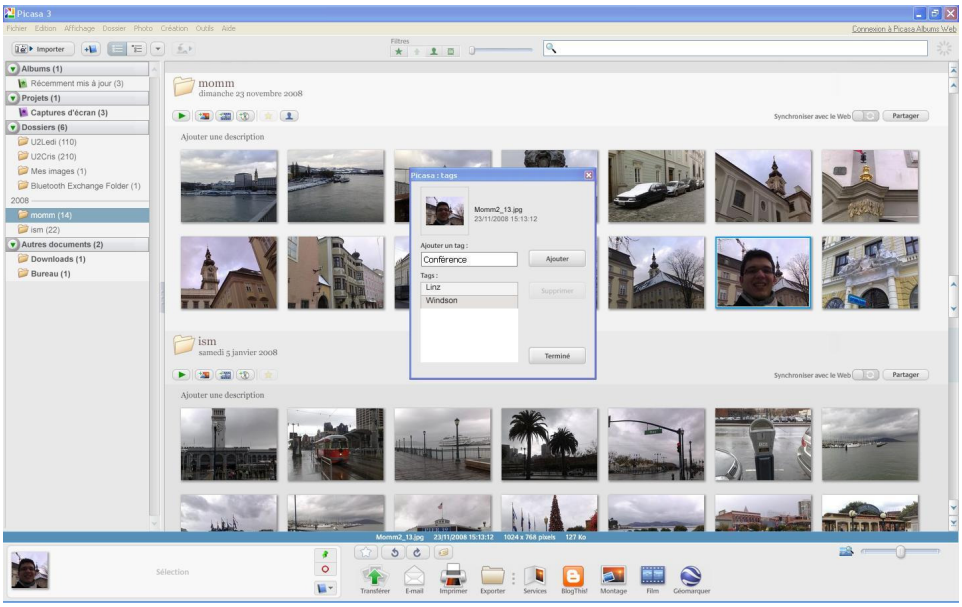


Figure 10 - Prise de vue de l'outil PICASA.

L'ajout manuel de mots-clés et l'usage de métadonnées (telles que la date de création) associées aux documents multimédias sont fondamentaux dans le fonctionnement et dans la performance de ces outils de gestion de multimédia. Ces outils reposent ainsi sur l'usage de l'annotation afin de faciliter l'organisation et le processus de recherche des documents multimédias.

Nous étudions plus en détail les mécanismes d'annotation de documents multimédias dans les prochaines sections. Nous exposons également quelques inconvénients liés à l'usage de l'annotation manuelle dans les outils de gestion de multimédias personnels.

3.2 L'annotation sémantique de documents multimédias

L'annotation englobe toutes les métadonnées associées à un document multimédia par les utilisateurs ou par des logiciels. Au delà d'un simple mécanisme d'indexation, l'annotation exerce une fonction de rappel à l'utilisateur en décrivant la situation et le contenu affichés par le multimédia. Par exemple, en lisant l'annotation d'une photo, l'utilisateur peut se rappeler des informations sur la scène capturée par cette image telles que l'endroit où il était lorsqu'il a pris cette photo. La présentation de ces métadonnées à l'utilisateur accroît l'importance rôle des documents multimédias dans l'enregistrement de souvenirs de lieux visités, d'événements, de situations occasionnelles et de personnes [Schweer *et al.*, 2007][Alexander, 2005].

Les mécanismes d'annotation de documents multimédias existants peuvent être classés selon le sujet décrit dans l'annotation. Ainsi, on trouve l'annotation sur le contenu qui retrace les objets et les personnes présentes sur une photo ou une vidéo. On trouve également l'annotation sur le contexte de création du document (le lieu de prise d'une photo, l'intervalle

de temps d'enregistrement d'une vidéo, etc.). L'annotation peut être différenciée enfin selon la manière dont elle est produite (manuellement, automatiquement ou semi-automatiquement) et selon son format de représentation.

3.2.1 Les mots-clés

L'usage de mots-clés est le système d'annotation le plus habituel. Un mot-clé est un terme non-hiérarchique employé pour décrire et étiqueter le contenu multimédia avec une information textuelle [Kuo *et al.*, 2007]. Les mots-clés peuvent être des informations descriptives du contenu ou de son contexte de création, par exemple « l'auteur », « les objets retracés », « l'endroit ». Cela peut être en plus des informations très particulières pour un utilisateur (telles qu'un sentiment, une association avec une autre situation vécue).

Dans les systèmes *desktop* et Web de gestion de multimédias, les mots-clés sont employés pour faciliter la navigation et l'accès à un document. Les mots-clés sont utilisés pour la création de nuages d'indexation (en anglais, *cloud tags*). Ces nuages sont un regroupement de mots-clés dans lequel les index possèdent diverses tailles de polices [Sinclair *et al.*, 2008]. La Figure 11 illustre un exemple de nuages de mots-clés du site Flickr. La taille de la police indique la popularité du terme. Concrètement, plus un mot-clé est ajouté à la collection des documents multimédias d'un site, plus il apparaîtra en gros dans le nuage d'indexation. Le nuage offre ainsi un aperçu des termes les plus utilisés et est également un fil conducteur pour la navigation de la collection de multimédias. Par exemple, sur le site Flickr, en cliquant sur l'un de ces mots-clés, le système affiche une page contenant toutes les photos indexées par le mot-clé.

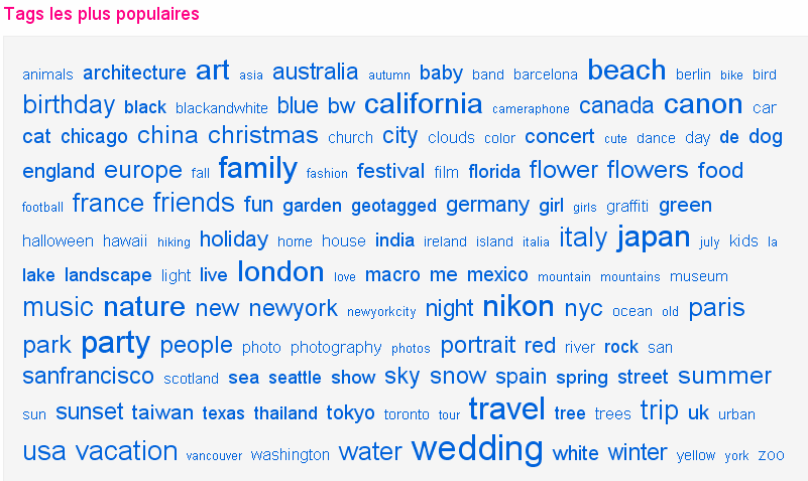


Figure 11 - Nuage des mots-clés les plus populaires du site Flickr.

3.2.2 Les inconvénients de l'annotation dans les outils de gestion de multimédias

La qualité des résultats des recherches sur les outils de gestion de documents multimédias et sur les sites Web 2.0 de partage de contenu est fortement dépendante de l'usage de l'annotation manuelle. L'ajout de mots-clés à une photo sur Flickr ou la géolocalisation d'une vidéo sur Picasa sont des exemples de cet usage. Cependant, hormis certains utilisateurs (tels que les photographes professionnels), la majorité des personnes ne sont pas prêtes à consacrer du temps à indexer et annoter correctement chacun de leurs documents multimédias [Nars, 2005]. Ainsi, seulement une partie des multimédias personnels est véritablement annotée et partagée. Il peut également exister un écart considérable entre l'instant de la création du document multimédia et le moment de l'annotation. Cet intervalle de temps peut compliquer

la tâche de l'utilisateur qui doit alors se rappeler des informations associées au document (Où cette photo a-t-elle été prise ? Qui était avec moi ce jour là ?) [Naaman *et al.*, 2004].

Au delà de l'aspect répétitif de la tâche d'annotation manuelle, deux autres inconvénients notoires sur les mécanismes d'annotation des outils de gestion de multimédias [Monaghan *et al.*, 2006] [Ames *et al.*, 2007] sont :

- **L'expressivité de l'annotation**. Les mots-clés attachés aux documents multimédias sont dépourvus de sens. Les systèmes traitent ces annotations comme de simples termes d'indexation et méconnaissent la relation sémantique entre le mot-clé et le document multimédia. Lors d'une recherche, ces systèmes réalisent à peine une comparaison syntaxique entre les mots-clés et les termes de la requête des utilisateurs. Cette comparaison syntaxique n'évite pas le problème des termes ambigus. Par exemple, le mot-clé « Paris » peut faire référence à plusieurs concepts : la capitale de la France, la ville aux États Unis, l'équipe de football de Paris Saint Germain et l'actrice américaine Paris Hilton. Une recherche avec des mots-clés ambigus produit une liste de documents multimédias contenant tous les sens des mots-clés et oblige l'utilisateur à réaliser lui-même le tri.
- **L'interopérabilité et la mobilité de l'annotation.** La plupart des systèmes de gestion de multimédia et de partage de contenu possède leur propre mécanisme de représentation et stockage de l'annotation. Par conséquent, un moteur de recherche différent de celui fourni par l'outil ne peut pas accéder à ces annotations. Divers modèles de représentation d'annotation rendent plus difficile l'interopérabilité entre les systèmes de gestion de multimédias. Un problème similaire apparaît lorsqu'un document multimédia est transposé d'un site de partage de contenu à un autre. Sur ces systèmes, l'annotation est majoritairement stockée séparément du document multimédia et, ainsi, ne sera pas transposée avec le document.

Nous décrivons dans les prochaines sections les différents formats existant pour la représentation de l'annotation d'un document multimédia. Nous nous intéressons également aux approches d'annotation automatique et semi-automatique qui tentent de minimiser les inconvénients de l'annotation manuelle présentés ci-dessus.

3.2.3 La représentation de l'annotation

L'enrichissement de documents multimédias avec des informations sémantiques fournit l'apport nécessaire pour le développement d'outils de gestion de contenus personnels plus performants. Cette amélioration passe également par la création automatique de ces métadonnées, et par l'utilisation des technologies de représentation plus expressives que de simples mots-clés. Dans cette section, nous étudions les diverses technologies pour la représentation de l'annotation des documents multimédias. Ces techniques de représentation diffèrent selon leur pouvoir d'expression, le vocabulaire fourni pour la description d'un document, et leur type de stockage. Tout d'abord, nous présentons les approches de sémantique interne DCMES[17], EXIF, ID3, et IIM qui stockent les métadonnées à l'intérieur des documents multimédias. Ensuite, nous présentons plusieurs ontologies d'annotation [Schreiber *et al.*, 2001], [Hollink *et al.*, 2004], [Halaschek *et al.*, 2005], [Naphade *et al.*, 2006], [Chai *et al.*,2008] et le standard MPEG7 d'annotation de documents multimédias. Ces dernières approches stockent les métadonnées dans un document externe au contenu multimédia, et offrent un pouvoir d'expression plus significatif que celui des approches de sémantique interne.

[17] http://dublincore.org/about/

3.2.3.1 Dublin Core Metadata Element Set (DCMES)

Le Dublin Core Metadata Initiave[18] est une organisation ouverte dédiée à la création et la diffusion de vocabulaires destinés à la description des métadonnées sur des ressources numériques. L'objectif de ces vocabulaires est de faciliter l'échange de documents entre des organisations et d'améliorer le fonctionnement des méthodes de gestion et de recherche de ces documents.

Son vocabulaire le plus adopté est la norme Dublin Core Metadata Element Set (DCMES) [19]. Elle comprend 15 termes dont la sémantique a été établie par un consensus international de professionnels provenant de diverses disciplines telles que la bibliothéconomie et l'informatique. Ces termes décrivent : i) le contenu du document (« description », « format », « relation », « source », « subject », « title »), ii) la propriété intellectuelle (« contributor », « creator », « publisher », « rights ») et iii) le document numérique lui-même (« date », « identifier », « language », « type »). Plusieurs formats peuvent être utilisés pour la représentation du vocabulaire DCMES, tels que XML, ou même une simple liste d'attributs-valeurs à l'intérieur du document. Grâce à sa simplicité et à sa généralité, le DCMES est fréquemment incorporé à d'autres schémas d'annotation pour la description de vidéos, textes, images et fichiers audio. Ce vocabulaire possède, par exemple, une large adoption au sein des standards du W3C.

3.2.3.2 IIM, ID3 et EXIF

Hormis Dublin Core, d'autres organismes proposent des vocabulaires de métadonnées pour la description de document numériques, principalement, des vocabulaires destinés à l'annotation d'images. Parmi ces vocabulaires, les plus diffusés sont IIM et EXIF.

IIM - Information Interchange Model (IIM)

IIM[20] est un vocabulaire de métadonnées proposé par l'IPTC (*International Press Telecommunications Council*) pour l'annotation de plusieurs types de médias. Néanmoins, il est plus réputé pour son utilisation dans la description d'images au sein du logiciel Adobe Photoshop. Très similaire au DCMES, l'IIM permet la représentation d'informations de propriété intellectuelle et des données sur le créateur de l'image (adresse, email, téléphone, etc.). Des mots-clés et une description textuelle peuvent être ajoutés manuellement au fichier de l'image. Adobe Photoshop et d'autres logiciels offrent des interfaces graphiques de visualisation et de saisie de métadonnées IIM.

ID3

ID3[21] est une spécification de métadonnées amplement utilisée pour l'annotation d'un fichier audio, principalement, au format MP3. Il permet de spécifier des informations sur le titre de la chanson, de l'album, le nom de l'artiste, le genre de musique et sa date de création. Ces métadonnées sont stockées à la fin du fichier audio afin d'éviter des problèmes avec les décodeurs. Cette spécification est destinée au partage de fichiers musicaux, et peut s'avérer insuffisante pour l'annotation de fichiers audio personnels.

EXIF- EXchangeable Image File

La majorité des fabricants d'appareils photos numériques adoptent un vocabulaire commun pour la distribution de métadonnées relatives aux images prises par leurs équipements : *EXchangeable Image File* -EXIF[22]. Ce vocabulaire est une partie de la recommandation

[18] http://dublincore.org/about/

[19] http://dublincore.org/documents/dces/

[20] http://www.iptc.org/cms/site/index.html?channel=CH0100

[21] http://www.id3.org/

[22] http://exif.org/

Design rule for Camera File system (DCF) crée par le *Japan Electronics and Information Technology Industries Association* (JEITA)[23] afin d'encourager l'interopérabilité entre dispositifs de capture et systèmes de gestion de photos.

Lors de la prise d'une photo, l'appareil photo numérique ajoute automatiquement à l'en-tête de l'image une description au format EXIF. Cette description possède des métadonnées sur l'image (nom, taille, la date de création), sur l'appareil photo (modèle, fabricant, résolution maximale), et sur les réglages utilisés pour prendre la photo (l'intensité du flash, l'ouverture du diaphragme). L'archive EXIF permet aussi l'ajout des coordonnées GPS au descriptif de l'image. L'EXIF est largement adopté et exploité par les outils de gestion de multimédia qui profitent, par exemple, de la valeur de la date de prise des images dans leurs méthodes d'organisation. Néanmoins quelques modèles de dispositifs mobiles ne génèrent pas ces métadonnées au moment de la prise d'une photo.

3.2.3.3 MPEG7

Les vocabulaires présentés auparavant s'intéressent très peu à la description du contenu d'un document multimédia. La description du contenu est limitée à l'attachement d'un texte ou de quelques mots-clés. Le standard MPEG-7 possède une toute autre préoccupation. Issu des efforts de standardisation du groupe de travail *Moving Pictures Expert Group* (MPEG), MPEG-7 vise la description sémantique des ressources multimédia, spécialement les fichiers audio et vidéo. À la différence des autres standards MPEG plutôt destinés à l'encodage des documents audiovisuels, MPEG-7 régit la description des métadonnées sur le contenu afin de faciliter la recherche, la navigation, et le filtrage sans le besoin de décompresser le contenu encodé.

MPEG-7 fournit un vocabulaire de description de contenu qui comprend : des descripteurs (D), des schémas de description (DS) et un langage de définition des descriptions (DDL). Un descripteur représente une caractéristique visuelle de bas niveau (telle qu'une couler, une texture) ou auditive (telle qu'un timbre, une mélodie) d'un document multimédia. Les *schémas de description* offre une représentation d'informations de plus haut niveau du contenu (objets, événements, interactions entre les objets, etc.). Le vocabulaire MPEG-7 peut être étendu en utilisant le langage DDL qui définit un schéma pour la création de nouveaux descripteurs et de schémas de description.

MPEG7 est largement utilisé pour l'annotation des vidéos et images. Par exemple, l'outil Vanotea [Schroeter *et al.* , 2003], développé par l'université de Brisbane, permet aux utilisateurs de collaborer dans l'ajout de métadonnées à des vidéos et à des images en utilisant MPEG-7 pour le stockage des métadonnées.

MPEG-7 permet également de segmenter un document en parties qui correspondent à des segments spatiaux, temporels ou spatio-temporels. À chaque segment créé, un ensemble d'annotations peuvent être attaché.

3.2.3.4 Les ontologies d'annotation de documents multimédias

L'interopérabilité sémantique entre applications et l'ajout de métadonnées pour enrichir la description des ressources font partie également des objectifs du Web Sémantique [Berners-Lee *et al.*, 2001]. L'idée principale est d'étendre le Web actuel à l'aide de représentations formelles des métadonnées sur les ressources Web (pages Web, images…), sur les services (applications, services Web…) et sur les utilisateurs (profils, réseau social …). Ces métadonnées formalisées facilitent le traitement automatique par des logiciels de recherche et d'agrégation d'information. De plus, ces logiciels peuvent activer des mécanismes d'inférence

[23] http://www.jeita.or.jp/english/

et d'intégration de données afin de dériver des nouvelles informations sur les ressources décrites et, ainsi, mieux répondre aux besoins des utilisateurs.

Le W3C Media Annotations Working Group[24] (MAWG) suggère l'utilisation des technologies du Web Sémantique pour la représentation de métadonnées des fichiers multimédias. La formalisation des métadonnées de multimédias, et l'usage de vocabulaires standard peut garantir l'augmentation d'expressivité des annotations, et améliorer l'interopérabilité des systèmes de gestion de multimédia. Le W3C est l'acteur majeur dans la spécification des langages formels de représentation des métadonnées du Web Sémantique. Le consortium propose deux standards pour leurs descriptions : RDF (Ressource Description Framework) et OWL (Ontology Web Langage).

RDF- Ressource Description Framework

RDF décrit des ressources Web en permettant des affirmations (« statements ») sur ces ressources qui sont identifiées par des URI (Uniform Resource Identifier). Chaque affirmation correspond à un triplet « ressource, propriété, valeur » qui peut être interprétée comme « le sujet, le prédicat et l'objet ». Une représentation possible de RDF est l'usage de graphes directionnels étiquetés dont les arcs font référence aux propriétés et les nœuds correspondent aux ressources et aux valeurs. Le langage RDFS (ou RDF Schema) offre les moyens de définir des schémas de métadonnées RDF. Il est possible de définir des classes de ressources, spécifier des hiérarchies de concepts, et associer des contraintes aux valeurs des propriétés.

La Figure 12 illustre une proposition du MAWG de description de métadonnées d'une photo en utilisant RDF sur la forme d'un document XML (selon la spécification XML/RDF). La ressource principal (*rdf :about*) est l'image à décrire appartenant à une collection personnelle et disponible sur le Web. Le MAWG défend la réutilisation de plusieurs schémas RDFS pour la description des métadonnées sur des images. L'exemple de la Figure 12 fait usage de deux schémas RDFS : Dublin Core RDF et FOAF. Le Dublin Core RDF[25]est une représentation en RDFS du DCMES présenté dans la section 3.2.3.1. FOAF (Friend Of A Friend) est un vocabulaire pour la description d'un utilisateur Web (nom, site Web, email, image personnelle, etc.) et de son réseau social (l'URI des profils de ses connaissances). L'exemple de la Figure 12 utilise le vocabulaire du Dublin Core pour exprimer la description de la photo, son type de fichier et le créateur du contenu. Contrairement à l'usage simplifié du DCMES, le créateur n'est pas décrit par une chaine de caractères. Il fait, en revanche, référence à un profil FOAF.

OWL (Ontology Web Langage)

OWL est le standard du W3C pour l'expression d'ontologies. Selon Gruber [Gruber, 2008], une ontologie définie formellement des concepts, leurs relations et d'autres distinctions (par exemple, des attributs) pour la modélisation d'un domaine. Les ontologies sont essentiellement exploitées de deux manières :

- *Pour la définition des taxonomies*. Une ontologie peut définir une hiérarchie de concepts d'un domaine. Dans ce cas, l'ontologie exprime des catégories (des classes) de concepts et la subsomption est utilisée pour représenter l'hiérarchie. Par exemple, une ontologie décrivant les mammifères peut contenir un concept racine *Mammifère*, un sous-concept *Primate*, et un concept *Humain* qui est lui un sous-concept de Primate. Toutes les relations de Mammifère sont valides sur les Humains (telle qu'avoir quatre membres)

- *Pour la formalisation d'un vocabulaire et les relations entre les termes*. L'objectif de ce type d'ontologie est de spécifier un vocabulaire commun d'un domaine et de l'exprimer dans une représentation compréhensible par une machine. Ces ontologies sont utilisées principalement

[24] http://www.w3.org/2008/WebVideo/Annotations/

[25] http://purl.org/dc/elements/1.1/

pour assurer l'interopérabilité entre systèmes hétérogènes. Par exemple, le vocabulaire FOAF définit formellement le profil d'une personne sur le Web. Une ontologie du type vocabulaire commun peut également faire référence à des concepts décrits dans une taxonomie représentée par une autre ontologie.

```
  <!—Exemple de description d'une photo en utilisant RDF et les
vocabulaires
 Dublin Core et FOAF -->
<rdf:Description
rdf:about="http://www.w3.org/2005/Incubator/mmsem/XGR-image-
annotation/Personal.jpg">
    <dc:type rdf:resource="http://purl.org/dc/dcmitype/Image"/>
    <dc:description>Photo of Katerina Tzouvara during Vacations in
Thailand</dc:description>
    <dc:creator>
      <foaf:Person>
        <foaf:familyname>Stabenaou</foaf:familyname>
        <foaf:firstname>Arne</foaf:firstname>
      </foaf:Person>
    </dc:creator>
    <dc:date>2002-12-04</dc:date>
```

Figure 12 - Exemple d'utilisation de RDF pour la représentation de métadonnées d'une photo (adapté du site du MAWG).

Les ontologies peuvent être exprimées en plusieurs langages comme UML, XML Schema, RDF Schema et OWL. Par exemple, la Figure 13 illustre une ontologie d'annotation de photos exprimée en UML [Schreiber *et al.*, 2001]. Cette ontologie permet la description du photographe, du moment de la prise de la photo et des « agents » participants de la scène.

Bien que plusieurs représentations des ontologies existent, OWL est beaucoup plus expressif en termes sémantiques que ces autres langages. OWL considère entre autres, les relations entre les classes et/ou instances, leur cardinalité, les caractéristiques des propriétés (symétrie, transitivité). Le langage OWL permet également d'exprimer des équivalences entre des classes et de créer de nouveaux concepts à l'aide des opérations ensemblistes telles que la disjonction, l'union, etc. OWL compte trois dialectes à l'expressivité croissante (OWL Lite , OWL DL et OWL Full). OWL DL, par exemple, repose sur la logique de description et peut être utilisé dans des processus d'inférence de connaissances. Ils existent des nombreux moteurs d'inférences (tels que Race, Jess, Pellet) pour raisonner sur la logique de description qui acceptent des fichiers OWL DL comme entrée du processus. Ces moteurs d'inférence peuvent déduire de nouvelles connaissances (tels que des propriétés entre les instances de l'ontologie) à partir des « faits » décrits sur l'ontologie. Un langage à base de règles, le SWRL (*Semantic Web Rule Langage*), est également disponible pour étendre le raisonnement d'ontologies OWL à travers l'usage des variables et d'un moteur des règles d'inférence.

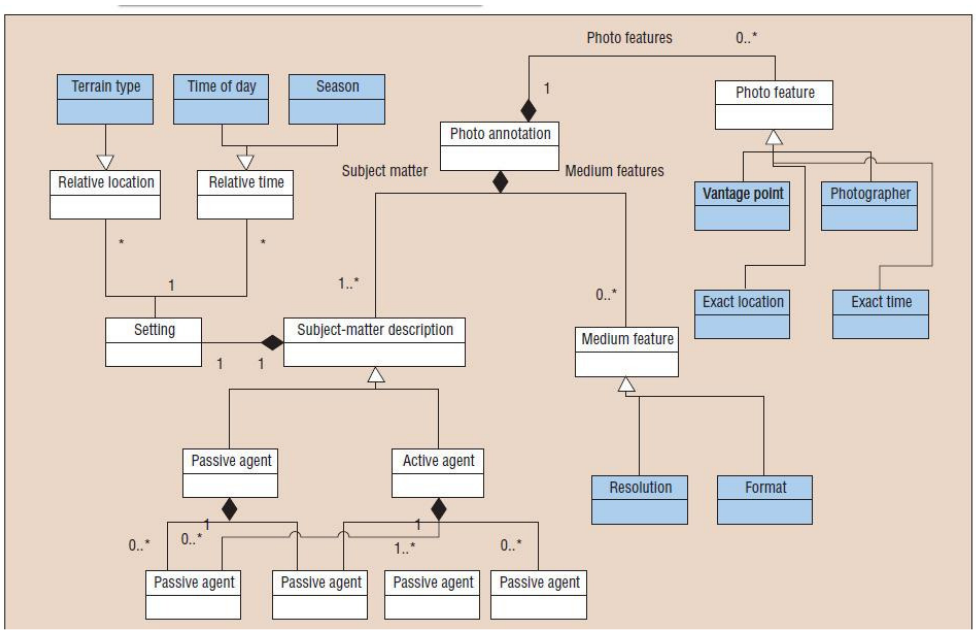


Figure 13 - Diagramme UML de l'ontologie d'annotation de photo proposée par Schreiber [Schreiber *et al.*, 2001]

Nous présentons par la suite d'autres propositions d'ontologies d'annotation de multimédias qui utilisent les langages RDF et OWL.

OntoAlbum

OntoAlbum [Chai *et al.*, 2008] propose l'usage d'ontologies OWL pour l'amélioration de la gestion de photos personnelles, spécialement, pour la résolution du problème d'expressivité des annotations. OntoAlbum offre deux ontologies pour l'annotation : une ontologie pour la description d'une photo et une taxonomie des membres d'une famille (père, mère, femme, enfant, etc.). La première ontologie est illustrée sur la Figure 14.

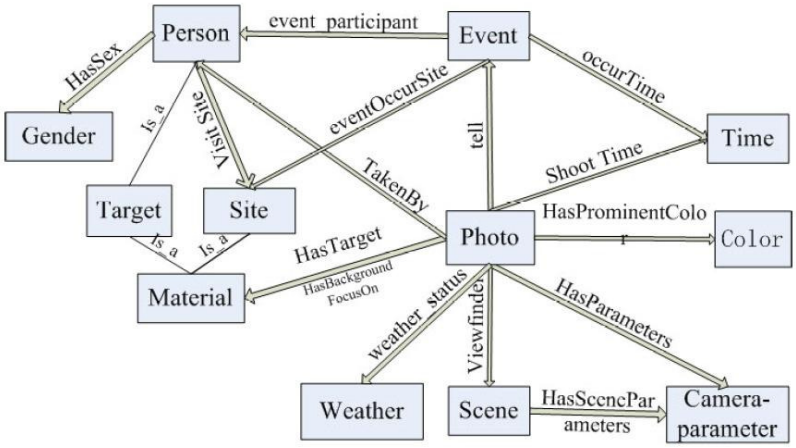


Figure 14 - Ontologie OWL d'annotation de photos personnelles [Chai *et al.*, 2008].

Les auteurs modélisent une photo comme un évènement qui peut avoir plusieurs participants. Le concept « Person » peut être relié à la taxonomie exprimée par la deuxième ontologie (Figure 15).

Les deux ontologies sont représentées en OWL-DL, la version OWL reposant sur la logique de description. Dans l'outil OntoAlbum, une interface graphique permet aux utilisateurs de se servir des concepts de deux ontologies dans la mise en œuvre de

l'annotation. Par exemple, on peut exprimer manuellement que le mot-clé « Karol » décrit le nom de l'épouse du propriétaire de la photo et que cette personne est un des participants de la scène capturée. L'arbre hiérarchique des concepts des ontologies est utilisé pour la navigation de la collection. Chaque concept fonctionne comme une catégorie dans les outils *desktop* de gestion de multimédias de la section 3.1.2.

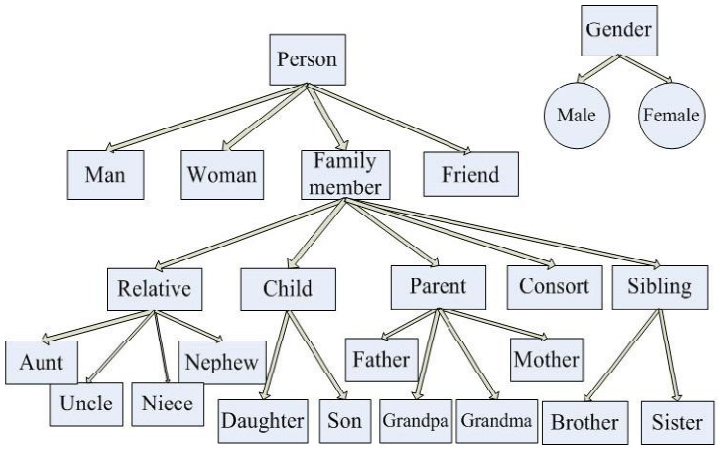


Figure 15 - Taxonomie des personnes.

XMP- Extensible Metadata Platform[26]

XMP est une plate-forme ouverte de création de vocabulaires de métadonnées proposé par Adobe et repose sur une représentation RDF de ces informations. XMP offre également un vocabulaire de base qui incorpore des termes des vocabulaires EXIF, DCMES et IIM. Les métadonnées XMP peuvent être insérées à l'intérieur de plusieurs formats de fichiers tels que PNG, TIFF, HTML, PostScript et JPEG. Cependant, XMP est principalement adopté pour la description textuelle de métadonnées sur un document PDF et des fichiers images des dernières versions des logiciels Adobe.

3.2.3.5 Ontologies pour l'annotation de segments d'un document multimédia

Les ontologies décrites auparavant considèrent indivisible le contenu du document multimédia. D'autres approches, telles que [Halaschek *et al.*, 2005] [Garnaud *et al.*, 2006], proposent des vocabulaires pour l'annotation de segments d'un document multimédia.

PhotoStuff [Halaschek *et al.*, 2005], par exemple, offre une ontologie OWL et un outil d'annotation manuelle qui permet à l'utilisateur de découper l'image en régions et de les annoter avec l'association d'une instance d'un concept de l'ontologie au segment sélectionné. Garnaud *et al.*, [Garnaud *et al.*, 2006] ont construit un outil similaire pour l'annotation de vidéos en utilisant l'ontologie LSCOM [Naphade *et al.*, 2006]. Cette ontologie est une taxonomie de concepts relatifs au domaine de vidéos d'informations télévisées et contient plus de 800 termes (tels que « Bank », « Person », « Thief »). L'utilisateur peut associer manuellement des segments d'une vidéo à ces concepts.

Les approches [Hollink *et al.*, 2004] [Bloehdorn *et al.*, 2005] proposent, de plus, de détecter automatiquement des segments des images qui représentent d'objets ou des personnes. L'objectif est de réduire une partie de l'effort d'annotation des utilisateurs. Par exemple, la Figure 16 illustre le découpage généré par le logiciel proposé par Hollink *et al.* [Hollink *et al.*, 2004] qui utilise des filtres de Gabor appliqué aux couleurs pour la détection

[26] http://www.adobe.com/products/xmp/

des régions. Une ontologie spatiale permet à l'utilisateur d'exprimer des relations topologiques entre les régions détectées (« la région Dog est à gauche de la région Adult_Female »).

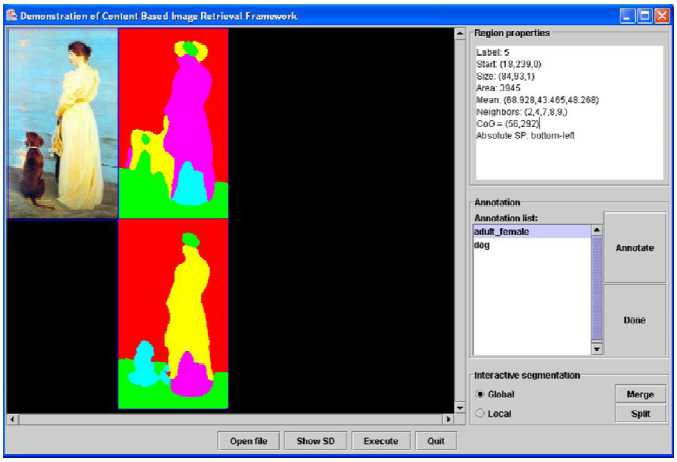


Figure 16 - Découpage d'une image en zones pour aider l'annotation sémantique sur le contenu

3.2.4 Génération automatique de métadonnées

Une fois définis le vocabulaire et le langage de représentation des métadonnées, il convient de décider des mécanismes de complétion de ces informations. La plupart des outils d'annotations présentés auparavant reposent sur l'ajout manuel de ces métadonnées. Nous nous intéressons dans cette section aux propositions qui offrent des méthodes de génération automatique des métadonnées.

3.2.4.1 La classification de documents multimédiaS au niveau signal

Une des approches d'organisation de multimédia la plus ancienne est la classification automatique du contenu en utilisant des informations au niveau signal [Haralick *et al.*, 1973]. Ces approches, principalement appliquées aux images et vidéos, exploitent les propriétés visuelles de bas niveau (texture, couleur, luminosité) afin de catégoriser le contenu. Les catégories sont principalement la détection de coucher/lever du soleil [Vailaya *et al.*, 2001] et des visages [Tan *et al.*, 2006], ou la classification en photos de paysages [Szummer *et al.*, 1998]. Les méthodes de classification sont également utilisées pour la génération d'annotations. Par exemple, les approches MediaAssist [O'Hare *et al.*, 2006] et PhotoCopain [Tuffield *et al.*, 2006] annotent automatiquement des images avec des mots-clés symbolisant la catégorie détectée (telle que « indoor » et « outdoor »).

Ces techniques ont progressé énormément avec l'usage de méthodes puissantes provenant de l'intelligence artificielle telles que le SVM (Support Vector Machine) et les réseaux Bayésiens. Cependant, son applicabilité reste réduite au sein des outils de gestion de multimédias personnels. Les difficultés d'adoption de ces approches sont liées à la diversité sémantique des photos d'une collection personnelle. Ceci rend plus difficile l'entraînement de certains algorithmes d'apprentissage utilisés pour la classification [Shevade *et al.*, 2007]. De plus, à l'exception de la détection de visages, les classifications possibles aujourd'hui (telle que la classification en photo de paysage), ne se trouvent pas parmi les critères d'annotation et de recherche les plus demandés par les utilisateurs. Ces critères qui sont plutôt liés à la reconnaissance d'objets et des personnes et des critères concernant le contexte de création de documents [Matellanes *et al.*, 2005][Naaman *et al.*, 2004]. Nous détaillons ces critères de recherche dans la section 3.3.

3.2.4.2 La suggestion d'annotations à partir des mesures de similarité au niveau signal

La suggestion ou propagation d'annotation repose sur l'utilisation d'un ensemble initial de documents multimédias pré-annotés. À l'arrivée d'un nouveau document, une mesure de similarité [Liu *et al.*, 2006] est appliquée afin de trouver un ou plusieurs documents similaires. Les annotations des documents similaires sont ainsi suggérées pour l'annotation du nouveau document. La mesure de similarité correspond à une fonction mathématique qui établit une distance entre deux documents dans un espace vectoriel de plusieurs dimensions. Les propriétés des documents utilisées par ces fonctions de similarité varient selon les propositions. Les critères bas niveau visuelle (texture, histogrammes de couleur, etc.) sont les plus utilisées pour la comparaison d'images et de vidéos. D'autres approches utilisent la proximité temporelle dans la suggestion de l'annotation en utilisant des méthodes de regroupement (*clustering*, en anglais). Si un document est annoté par l'utilisateur son annotation est propagée aux documents appartenant au même regroupement.

Elliot *et al.* proposent l'utilisation de multicritères et d'ontologies dans le calcul de la mesure de similarité [Elliott *et al.*, 2008]. La Figure 17 illustre cette proposition de suggestion d'annotation pour des photos personnelles.

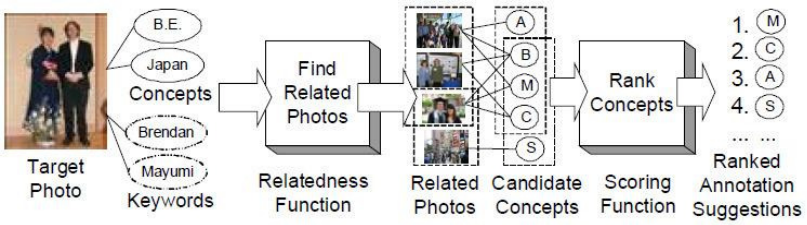


Figure 17 - Processus multicritère de suggestion d'annotation proposé en [Elliott *et al.*, 2008].

Lors de l'insertion d'une photo à un album, l'utilisateur peut attacher manuellement des concepts provenant d'une ontologie et des mots-clés. Une combinaison de mesures de similarité est appliquée en utilisant les critères : i) de partage des mêmes mots-clés, ii) d'appartenance à un même album et iii) de possession des mêmes concepts ou synonymes. Une combinaison linéaire des mesures guide le filtrage des photos « similaires » et l'ordre final des annotations suggérées. Les mots-clés les plus utilisés sont également injectés dans ce classement. Malgré les mauvais résultats lorsqu'aucune photo d'un nouvel album a été annotée (seule 30% des annotations suggérées sont acceptées par les utilisateurs), cette proposition montre une alternative aux approches de suggestion d'annotation uniquement basées sur le critère visuel qui ne correspond pas toujours aux résultats attendus par les utilisateurs [Wang *et al.*, 2007].

3.3 La sensibilité au contexte et la gestion de documents multimédias

La notion de contexte est utilisée par plusieurs domaines de recherche qui l'appréhendent sous différentes perspectives. Par exemple, dans le domaine des Systèmes d'Information sensibles au contexte (en anglais, *context-aware systems*), la notion de contexte fait référence principalement à la caractérisation de la situation de l'utilisateur lors de l'accès à un système. D'après Dey [Dey *et al.*, 2000], *le contexte est construit à partir de tous les éléments d'information qui peuvent être utilisés pour caractériser la situation d'une entité. Une entité correspond ici à toute personne, tout endroit, ou tout objet (en incluant les utilisateurs et les applications eux-mêmes) considéré(e) comme pertinent(e) pour l'interaction entre l'utilisateur et l'application*. Le qualificatif de « sensible au contexte » est donc associé aux systèmes qui guident leur comportement selon leur *contexte d'utilisation*. La plupart des

auteurs considèrent la sensibilité au contexte comme la capacité de percevoir la situation de l'utilisateur en plusieurs aspects, et d'adapter en conséquence le comportement du système.

Concernant le domaine de la gestion de documents multimédias, la notion de contexte, et principalement, son exploitation, sont légèrement différentes. En fait, les auteurs Naaman *et al.* [Naaman *et al.*, 2004] et Mattelanes *et al.* [Matellanes *et al.*, 2005] ont étudié le comportement des utilisateurs lors de l'organisation et de la recherche de photos personnelles. Une grande partie des informations évoquées par les personnes à propos de souvenirs concernant leurs images est constituée des aspects liés au *contexte d'utilisation* de l'appareil photo numérique lors de la prise de la photo («quand », « où », « avec qui », « prise par quoi », etc.). La caractérisation du contexte dans lequel l'action de création de la photo se produit donne ainsi du sens à la photo car ces informations contextuelles sont également perçues par les personnes comme étant fondamentales lors d'une recherche à posteriori de ces documents. Les personnes étudiées par ces auteurs défendent que les informations à propos du *contexte de création* des documents les aident à trier plus facilement un ensemble de documents multimédias.

La Figure 18 illustre le classement de plusieurs aspects contextuels (tels que l'année, la date, la météo, la saison, la localisation, les personnes proches) attribué par les utilisateurs pour se rappeler d'une photo (ligne normal) et pour sa recherche effective (ligne pointillée avec des bulles). La localisation et la période de la journée sont parmi les aspects les mieux classés.

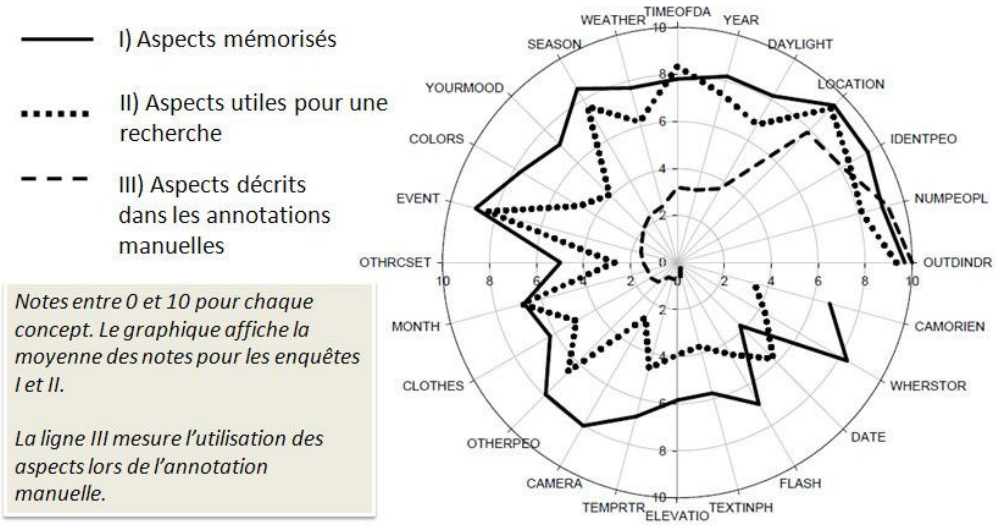


Figure 18 - Comparaison entre les aspects perçus comme étant importants pour une recherche et ceux effectivement décrits par les utilisateurs [Naaman *et al.*, 2004].

Malgré l'importance attribuée à ces aspects, ces auteurs ont observé que la plupart des informations contextuelles n'était pas informée par les utilisateurs (ligne pointillée) lors de l'annotation manuelle. En réalité, l'annotation manuelle demande aux utilisateurs de consacrer du temps à annoter leurs documents. Lors du moment de la description, il est difficile pour l'utilisateur d'évaluer quels seront les bénéfices futurs apportés par l'annotation et, ainsi, elles se limitent à seulement quelques aspects [Naaman *et al.,* 2004] [Ames *et al.*, 2007].

Ces études montrent l'importance des méthodes pour la dérivation automatique de ces aspects contextuels et pour leur utilisation lors de l'annotation et de l'organisation de documents multimédias.

Nous examinons à présent des approches d'organisation et d'annotation qui tentent de satisfaire cette nécessité de production et d'utilisation de métadonnées décrivant le *contexte de création* de documents multimédias. D'autres notions de contexte et approches sensibles au contexte pour l'adaptation d'applications mobiles seront étudiées plus profondément dans le chapitre 5.

3.3.1 Le temps

La date et l'heure de création d'un document multimédia sont les informations contextuelles les plus faciles à exploiter grâce à leurs disponibilités au sein des fichiers multimédias (par exemple, EXIF sur les photos JPEG, et des métadonnées sur les fichiers AVI et MP3). Ces informations sont largement exploitées par les outils de gestion de multimédia pour établir un ordre chronologique des documents (comme dans Picasa) et pour la génération de regroupements temporels qui identifient les événements possibles (comme dans Adobe PhotoShop Album[27] et dans Windows Vista). Malgré l'utilité pour la navigation, la date n'est pas forcément l'attribut le plus adopté lorsqu'une personne souhaite chercher une photo de façon temporelle. À l'exception des dates commémoratives, les personnes ne se souviennent pas des dates précises de création de leurs documents. D'autres attributs temporels, tels que le moment de la journée, le mois, l'année, se montrent plus pertinents pour la recherche (Figure 18).

3.3.2 L'espace

La localisation du lieu de création de documents multimédias est un autre aspect important des informations contextuelles. Initialement, cette information était acquise à l'aide d'un synchronisme entre des fichiers de trajets capturés par un équipement GPS et une collection de multimédias. Des logiciels tels que EXIFtool[28] associent au fichier EXIF d'une photo les coordonnés géographiques capturées par le GPS dans le même instant de capture de l'image. Ce type de synchronisation est également disponible pour des vidéos et des fichiers audio en utilisant d'autres formats de représentation de métadonnées (tels que XMP pour les vidéos et ID3 pour les fichiers audio).

WMMX [Toyama *et al.*, 2003] est une des premières approches d'organisation de multimédias à explorer la possibilité d'accéder aux données spatiales d'une photo. WMMX est un projet de recherche de Microsoft dont l'objectif est de construire un outil de visualisation des images à l'aide d'un SIG (Système d'Information Géographique). WMMX utilise une base de données spatiale pour indexer des photos personnelles géoréférencées. WMMX offre plusieurs interfaces de visualisation avec des cartes fournies par MapPoint[29].

La difficulté du synchronisme manuel et les prix des équipements GPS ont freiné le succès de ce type d'approche à leur commencement. Cependant, ces dernières années, la popularisation de services d'affichage de cartes géographiques (tel que Google Maps et Yahoo ! Maps) a augmenté l'intérêt et la familiarisation du grand public avec les informations de type spatial. De nombreuses approches d'annotation géoréférencée et manuelle (*geotagging*, en anglais) ont ainsi émergé. Panoramio[30], TriperMap et FlickrMap sont des exemples de succès du *geotagging* des photos sur le Web. Panoramio est une application Web

[27] http://www.adobe.com/

[28] http://www.sno.phy.queensu.ca/~phil/exiftool/

[29] http://mappoint.msn.com/

[30] http://www.panoramio.com/

dont l'objectif est de créer une base de photos de paysages et de monuments des villes partout dans le monde. Des personnes mettent en ligne gratuitement leurs photos et les positionnent sur une carte de la Terre. Un service fait un tri pour enlever les photos qui n'illustrent pas le lieu (par exemple, la photo d'une personne) et les photos sélectionnées sont affichées sur des couches Google Earth et, récemment, sur Google Maps. Le succès de Panoramio a incité d'autres outils à rendre disponibles des interfaces de *geotagging* manuel, notamment Flickr et Picasa, vus dans la section 3.1.2.

3.3.3 Le contexte de création

Les approches d'annotation reposant sur la localisation explorent à peine les coordonnées géographiques dans l'organisation de multimédias. Cependant, ces informations de positionnement peuvent être combinées à d'autres sources d'information afin d'élargir le nombre de métadonnées sur le contexte de création des documents. Cette perspective est exploitée dans l'outil PhotoCompas [Naaman et *al.*, 2004] qui utilise des Services Web pour dériver des informations sur l'espace et le temps (telles que la saison, la température, la météo et le période de la journée par rapport au lever et au coucher du soleil). La Figure 19 illustre l'interface de navigation des collections de multimédias proposée par PhotoCompas qui offre de multiples manières de parcourir les collections.

PhotoCompas transforme par exemple les coordonnées GPS en une hiérarchie de localités (avec les niveaux Pays, Département, Ville) et offre une interface en forme d'arbre de navigation en utilisant cette hiérarchie. Les attributs de l'annotation générée (tels que l'altitude, la saison) peuvent être également utilisés afin de parcourir les collections d'images.

D'autres propositions d'outils desktop de gestion de photos tels que MediaAssist [O'Hare *et al.*, 2006][O'Hare *et al.*, 2007], PhotoCopain [Tuffield *et al.*, 2006] et, plus récemment, PhotoGeo [Lacerda *et al.*, 2008] reprennent et étendent les idées de PhotoCompas pour l'organisation et l'annotation de collection d'images personnelles.

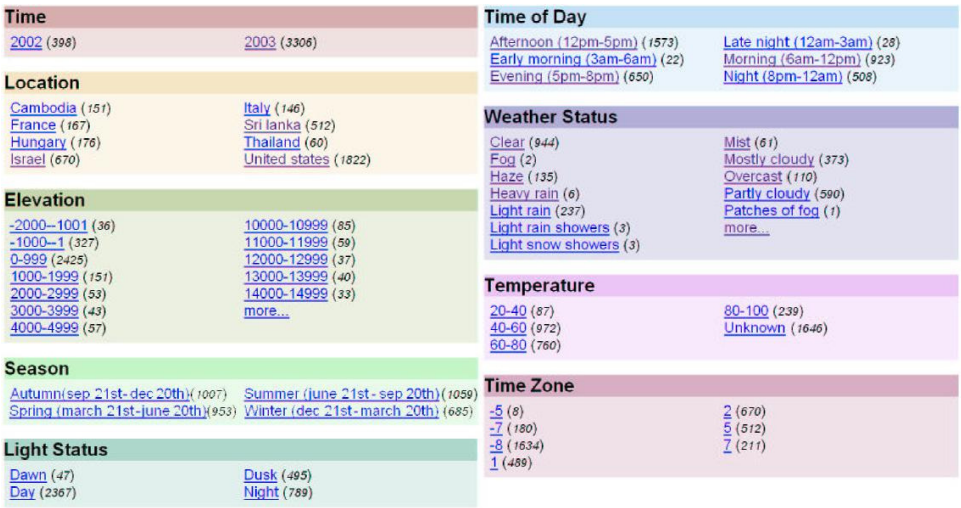


Figure 19 - Interface d'organisation automatique de photos de l'outil PhotoCompas [Naaman *et al.*, 2004].

MediAssist [O'Hare *et al.*, 2006][O'Hare *et al.*, 2007], proposée par le *Centre for Digital Video Processing* de l'Université de Dublin[31], exploite la date et les coordonnées géographiques disponibles sur les métadonnées EXIF des images pour la dérivation

[31] http://www.cdvp.dcu.ie/

d'informations sur le contexte de création identiques à celles proposées dans l'approche PhotoCompas (l'adresse, la saison, etc.). De plus, MediaAssist utilise des méthodes de classification d'images reposant sur les informations de niveau signal pour la détection de bâtiments (bâtiments présents ou non sur la photo) et de visages des personnes sur la photo. Toutes ses informations sont suggérées comme des annotations des photos et peuvent être modifiées sur l'interface de l'application (Figure 20).

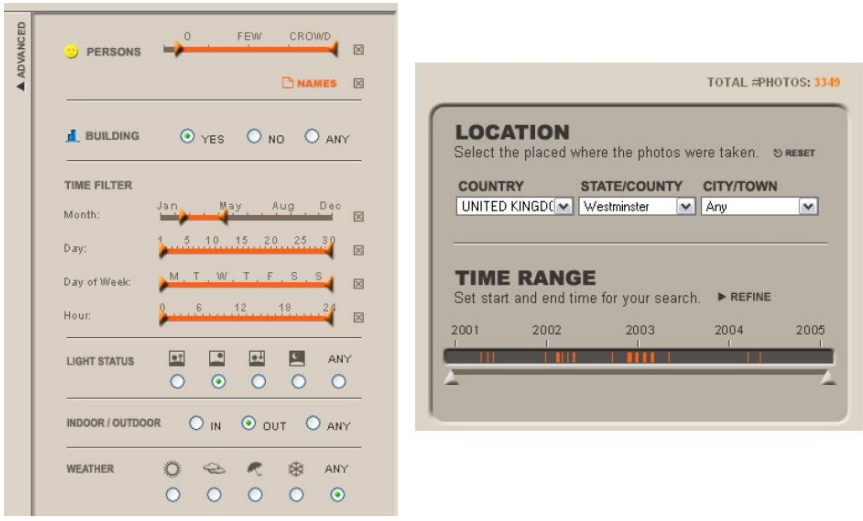


Figure 20- Les interfaces de validation d'annotation de MediAssist [O'Hare *et al.*, 2006][O'Hare *et al.*, 2007].

PhotoCopain [Tuffield *et al.*, 2006] est un système d'annotation sémantique des images reposant sur une représentation RDF de la norme EXIF. La proposition combine des métadonnées EXIF (telles que la date et les coordonnées géographiques), des données de l'agenda de l'utilisateur, et un *gazetteer* pour dériver des annotations contextuelles (par exemple, la ville où la photo a été prise). PhotoCopain intègre également des méthodes de traitement d'image afin de classifier les images dans les catégories portrait ou paysage et pour détecter la présence de visages (comme dans Picasa). L'annotation produite est représentée dans le format IIM qui permet l'usage de l'outil AKtiveMedia [Chakravarthy et *al.*, 2006] pour visualiser les métadonnées des images et pour les valider.

Contrairement aux autres approches, l'intérêt de PhotoGeo [Lacerda *et al.*, 2008] est d'exploiter la date et les coordonnées géographiques disponibles sur les métadonnées EXIF d'images pour la détection d'événements (une fête, une visite touristique, etc.). PhotoGeo, proposé par l'Université de Paraíba au Brésil, utilise la date pour acquérir sur l'agenda des utilisateurs (en utilisant Google Calendar) le nom des événements prévus. À partir des coordonnées géographiques, le logiciel consulte une base de données géographique peuplée par les contours des villes brésiliennes et dérive ainsi le nom des villes et des départements où les photos ont été prises. PhotoGeo applique la méthode DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) [Ester *et al.*, 1996] pour la génération de regroupements spatio-temporels des photos supposées représenter des événements réels. Chaque regroupement calculé est transformé dans une collection d'images. Les informations dérivées (nom d'événement et nom de ville) sont combinées avec la date de début et la durée de chaque regroupement pour générer le nom de la collection (tel que « ACM Symposium on Applied Computing – Vila Galé Hotel – Fortaleza – Ceará – Brasil – 05/16/2008 -5 jours »).

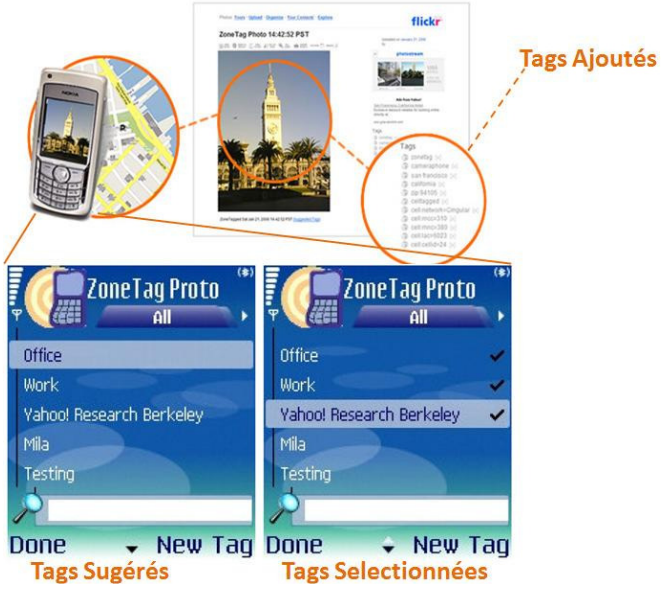
3.3.4 La mobilité et la sensibilité au contexte pour l'annotation et l'organisation de multimédias

La disponibilité des fonctions de production de contenu et de communication avec le Web sur les dispositifs mobiles et l'ajout progressif de capteurs (tel que le GPS) à ces terminaux ont accéléré le développement de propositions mobiles d'annotation et d'organisation de documents multimédias.

Par exemple, le fabricant Nokia offre dès 2008 l'application Location Tagger[32] de *geotagging* automatique qui fonctionne sur des téléphones disposant d'un GPS intégré. Lors de la prise de photo avec ce logiciel, les coordonnées géographiques sont stockées dans les métadonnées EXIF de l'image. Ces coordonnées peuvent être utilisées ultérieurement pour localiser la photo sur un plan. Location Tagger permet de transmettre les photos sur Flickr, Picasa ou Google Earth, pour qu'elles soient visualisées. Sa dernière version permet en plus d'afficher la position d'une photo sur le téléphone depuis Nokia Maps.

Un autre exemple est le système MMM Image Gallery [Sarvas *et al.*, 2004a] qui propose une application d'annotation semi-automatique pour les appareils Nokia 3650. Lors d'une prise de photo, l'application capture l'identificateur de la cellule GSM, l'identité de l'utilisateur et l'instant de prise de la photo. L'image est envoyée à un serveur avec son annotation. Le serveur exécute un processus de manipulation sur l'image pour en déduire des informations sur son contenu (si la photo a été prise à l'extérieur ou à l'intérieur) et sur sa localisation (adresse et le nom de l'endroit de prise). Ensuite, l'utilisateur emploie un navigateur mobile supportant XHTML afin de lire et valider l'annotation produite. L'imprécision de l'identification du relais cellulaire GSM couvrant la zone GSM et le long temps de réponse du serveur pendant les processus de téléchargement d'image ont été pointés par les utilisateurs comme les problèmes principaux d'utilisabilité de MMM. Cependant, l'avancée des capacités des derniers dispositifs mobiles pourra diminuer considérablement ces inconvénients.

Zonetag [Ames *et al.*, 2007] est un projet de recherche Yahoo ! pour la suggestion d'annotations d'images prises par des téléphones mobiles. Zonetag possède un logiciel mobile dont l'objectif est d'associer des informations géographiques à une photo prise, avant de les envoyer sur le service Flickr (cf. Figure 21).



[32] http://www.nokia.com/betalabs/locationtagger

Figure 21 - La suggestion d'annotation sur le service ZoneTag.

ZoneTag peut associer à la photo le numéro d'identification du relais cellulaire GSM ou les coordonnées géographiques dans le cas de la présence d'un GPS. Le serveur de Zonetag utilise des informations sur l'utilisateur et sa localisation afin de recommander des mots-clés d'annotations. Par exemple, la correspondance relais cellulaire/ville ou quartier est sauvegardée sur des serveurs de sorte que les prochaines photos prises dans la zone de couverture du relais en question seront automatiquement tagguées avec le nom du lieu. De plus, l'application utilise une méthode de regroupement spatiale afin de produire une liste de mots-clés les plus utilisés pour cette zone. Cette liste est enrichie avec les mots-clés les plus utilisés par l'utilisateur lui-même. La liste finale est renvoyée au dispositif mobile permettant à l'usager de sélectionner les mots-clés à rajouter. Plus il y aura d'utilisateurs et plus cette approche de suggestion sera efficace. Parmi les dispositifs supportés actuellement : Nokia N73, N95 et les Smartphones S60, 3eme édition. L'utilisateur peut utiliser les fonctionnalités de Flickr pour visualiser et rechercher les images publiées.

L'utilisation des dispositifs mobiles, pour l'annotation semi-automatique du nom des personnes appartenant à une photo, est étudiée par [Monaghan *et al.*, 2006]. La reconnaissance biométrique de visages et l'identification restent encore un défi, en particulier, lorsque les photos contiennent plusieurs visages, des vues de profil ou possèdent des conditions de lumières difficiles comme c'est souvent le cas sur les photos d'appareils mobiles [Davis *et al.*, 2005]. [Monaghan *et al.*, 2006] proposent de contourner ces difficultés en combinant les méthodes de détection de visage et l'annotation semi-automatique. La Figure 22 illustre le processus d'annotation proposée.

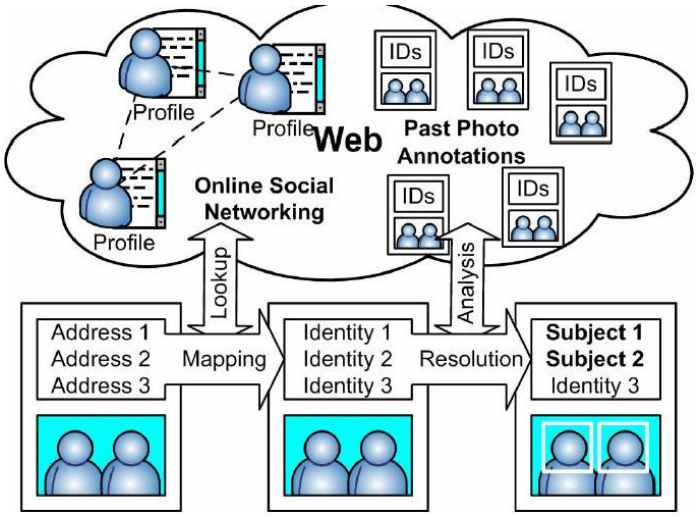


Figure 22 - Proposition d'identification des personnes sur une photo à l'aide de Bluetooth [Monaghan *et al.*, 2006].

Ce travail suggère la création d'une application mobile qui capture les adresses Bluetooth des dispositifs environnants chaque fois qu'une photo est prise. Un registre des utilisateurs d'un site de médias sociaux serait capable de récupérer l'identité des utilisateurs à partir de ces adresses. Dans le cas d'un échec, une méthode d'analyse des annotations des photos précédentes est mise en place. L'objectif est de sélectionner des photos dont l'adresse Bluetooth a été également retrouvée aux environs et de proposer l'annotation ajoutée préalablement à ces photos. [Monaghan *et al.*, 2006] suggère aussi l'utilisation de méthodes de détection de visages telles que celles disponibles sur la bibliothèque OpenCV[33] afin de mettre en évidence sur l'image des régions contenant un visage. Une interface permettra à

[33] OpenCV (Open Source Computer Vision), disponible sur: http://opencv.willowgarage.com/wiki/

l'utilisateur d'associer à chaque région les noms des personnes détectées par le Bluetooth. Malgré l'absence d'une implémentation de validation et la suggestion des noms de personnes que l'utilisateur ne connait même pas, la proposition de combiner la détection de visages et la capture d'adresses Bluetooth nous semble très prometteuse.

3.4 Recherche de documents multimédias

Les mécanismes d'organisation et de navigation des multimédias personnels peuvent s'avérer insuffisants lorsqu'un utilisateur essaie de retrouver un multimédia dans une grande collection de documents. Les fonctionnalités de recherche occupent ainsi un rôle central au sein des outils de gestion de documents multimédias personnels. Deux principales approches de recherche d'information ont été explorées pour aider les utilisateurs dans cette tâche : la recherche par le contenu et la recherche à base de propriétés.

Dans cette section, nous étudions brièvement des propositions de ces deux approches. Nous nous intéressons particulièrement aux approches de recherche à base de propriétés qui reposent sur les mécanismes d'annotation de documents multimédias présentés auparavant.

3.4.1 Recherche par le contenu

La recherche de documents par le contenu repose sur la comparaison de propriétés bas-niveau ou niveau signal de multimédias. La recherche d'image par le contenu (*Content Based Image Retrieval*, en anglais), par exemple, est une technique visant à effectuer des recherches d'images à l'aide de requêtes portant sur les caractéristiques visuelles des documents : texture, histogrammes de couleur, formes d'objet. Lors de la recherche par contenu d'un fichier audio, d'autres critères de similarité sont exploités tels que la hauteur ou le degré d'une note, l'amplitude, et la fréquence des séquences audio. Les approches de recherche de vidéo utilisent typiquement une fusion des critères bas-niveau d'audio et des images tout en ajoutant une dimension temporelle.

Les requêtes types de cette approche sont des « requête par l'exemple » (*query by example*, en anglais). L'utilisateur propose un document multimédia ou un morceau et le système répond avec des documents similaires. Un exemple de succès de ces approches est l'application mobile Shazam[34]. En enregistrant simplement à l'aide de ce logiciel une séquence audio de 30 secondes d'une émission de la radio, Shazam est capable de suggérer le nom de la musique et de proposer d'acheter une version complète à partir du téléphone.

Malgré les intenses études de plus d'une décennie, trois inconvénients majeurs persistent et limitent l'usage de ces approches pour la recherche de documents multimédias personnels [Wang *et al.*, 2006] [Wang *et al.*, 2007]: i) le fossé sémantique ; ii) l'effort computationnel d'indexation de large collections de documents et iii) la difficulté de l'utilisateur lors du choix ou de la construction des exemples.

Le fossé sémantique concerne principalement les méthodes de recherche d'images et des vidéos puisque ces propositions présentent encore un écart sémantique entre les documents suggérés et ceux attendus par les utilisateurs. En fait, la similarité visuelle n'est pas équivalente à la similarité sémantique. Lorsque deux images possèdent la même texture ou la présence massive d'un même dégradé de couleurs, elles sont considérées visuellement similaires par ces approches. Cependant, il y a une divergence entre ces caractéristiques visuelles de bas niveau et l'information sémantique représentée par une photo ou une séquence d'images. Deux photos visuellement similaires peuvent afficher des scènes complètement différentes.

[34] http://www.shazam.com/music/web/home.html

De plus, les approches de recherche par le contenu exigent une indexation de propriétés bas niveau des documents et les méthodes de comparaison lors d'une recherche peuvent s'avérer très lents et gourmands en termes de ressources. La construction des requêtes exemples présente également des difficultés. Il est difficile par exemple, d'exprimer les besoins de recherche de photos avec seulement une image exemple ou un dessin schématique [Gong *et al.*, 2005].

3.4.2 Recherche à base de propriétés

Afin de contourner les limitations de la recherche par le contenu, la plupart des approches fournissent la recherche à base de propriétés sur lesquelles les utilisateurs expriment les requêtes quasiment en langage naturel à partir de leur compréhension du contenu recherché. Les documents multimédias sont indexés par des termes extraits des métadonnées associés aux documents (les annotations manuelles, la date, etc.). Les utilisateurs spécifient leurs besoins dans une requête à texte libre qui fonctionne comme une combinaison logique de plusieurs termes (par exemple, une combinaison de mots-clés).

Dans le cas de l'usage d'une base de données traditionnelle, les documents indexés par tous les termes sont retournés et un critère informé par l'utilisateur définit le classement de résultats (tel qu'un ordre par la date de création). Cependant, l'usage de ce type d'approche est moins courant.

Généralement, l'indexation des documents multimédias repose sur des méthodes provenant du domaine de la recherche d'information tel que le modèle vectoriel et ses variations [Salton *et al.*, 1975]. L'idée centrale est d'associer à un document un vecteur de termes et d'attribuer une pondération qui exprime l'importance de chaque terme par rapport au document. Le vecteur de poids possède une taille équivalente au nombre de termes du corpus. Ce poids a une influence directe sur sa puissance de discrimination : plus le poids d'un terme dans le vecteur d'un document est grand, plus pertinent est le classement de ce document dans une recherche par ce terme. La requête de l'utilisateur est également représentée par un vecteur de termes dont la pondération est maximale pour les termes informés par l'utilisateur. Un processus de mise en correspondance (*matching*, en anglais) est effectué afin de renvoyer une liste ordonnée des documents les plus semblables à la requête. La similarité dans ce cas-ci est mesurée par la distance entre les vecteurs des documents et le vecteur de la requête dans un espace vectoriel multidimensionnelle [Zobel *et al.*, 2006].

La Figure 23 illustre un schéma générique d'implémentation du modèle vectoriel sur un moteur de recherche d'images par texte [Wang *et al.*, 2006].

La requête de l'utilisateur « plages brésiliennes » est transformée en quatre autres requêtes en utilisant un dictionnaire de synonymes et des mécanismes de cooccurrence (étape I de la figure) [Castells *et al.*, 2007]. L'implantation la plus courante d'indexation est la liste inversée de documents : chaque terme ou combinaison de termes est associé à une liste de documents qui attribuent un poids supérieur à zéro à ce terme. Une fois la liste de documents sélectionnée (étape II), une pondération finale de discrimination est calculée pour chacune des requêtes (étape III). La mesure statistique TF-IDF (de l'anglais, *term frequency-inverse document frequency*) est la plus utilisée :

$TF-IDF(t_j, d_i) = \log \frac{|C|}{|C(t_j)|} \bullet w(t_j, d_i)$, où $|C|$ est la taille du corpus de documents, $C(t_j)$ est l'ensemble de documents appartenant à liste inversée du terme t_j et $w(t_j, d_i)$ est le poids de relevance du terme t_j par rapport au document d_i.

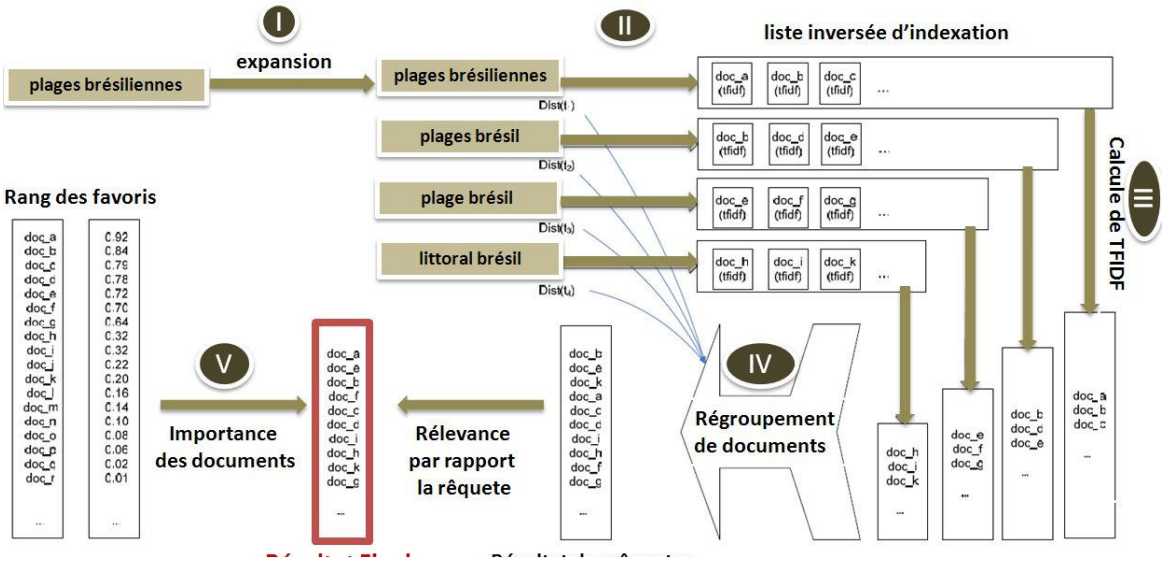


Figure 23 - Schéma général du processus de mise en correspondance en utilisant le modèle vectoriel, adapté de [Wang *et al.*, 2006].

Les documents de quatre requêtes sont regroupés dans une seule liste (étape IV), en utilisant pour le classement la mesure TF-IDF et un poids additionnel attribué à chaque requête (par exemple, les documents provenant de « littoral brésil » possèdent une pénalité sur leur poids en relation aux documents provenant de la requête initiale de l'utilisateur). Le processus de regroupement produit une liste de documents qui répondent à la requête dont le classement indique la relevance des documents par rapport aux termes de la requête de l'utilisateur. Avant de présenter la liste finale à l'utilisateur, le moteur de recherche peut modifier le classement des documents à l'aide d'un poids d'importance (un rang des documents favoris, une mesure de qualité du document) attribué auparavant à chaque document par l'utilisateur lui-même ou par une communauté d'usagers (étape V).

Ce schéma générique peut être appliqué à différents types de documents multimédias car ce mécanisme d'indexation considère à peine les termes extraits des métadonnées du document et ne manipule pas directement son contenu. Le défi demeure sur l'établissement de la meilleure pondération de chaque terme et sur la mesure de qualité appliquée aux documents.

Dans les sous-sections suivantes nous étudions brièvement les approches existantes de recherche de documents multimédia à base de propriétés.

3.4.3 Recherche par texte

Nous distinguons deux familles de systèmes de recherche de multimédia par texte:

- *Les moteurs web de recherche d'images et de vidéos*, tels que Google (plus spécifiquement Google Image et Google Videos) et PicSearch[35]. Google Image, par exemple, indexe les images du Web en employant les propriétés de l'image (par exemple, le nom, l'URL), les données textuelles présentes autour de l'image dans le document HTML, et les liens Web vers ce document [Gong *et al.*, 2005]. Une requête dans ces outils contient quelques mots-clés combinés et, de temps en temps, quelques options de filtrage (tel que le type de fichier, la taille, l'élimination de contenu

[35] http://www.picsearch.fr/

offensif). Bien que ces outils contiennent un nombre considérable d'images répertoriées, les résultats des recherches présentés aux utilisateurs contiennent fréquemment des images non désirées [Tuffield *et al.*, 2006] puisqu'il est encore difficile d'extraire des textes qui ont un lien sémantique effective avec les images.

- *Les moteurs de recherche basés sur des collections*, comme Flickr, Picasa et Youtube ont des résultats de recherche qui présentent moins de problèmes de précision en comparaison de ceux présentés par les moteurs de recherche du web [Wang *et al.*, 2007]. La différence se produit parce que, généralement, ces outils indexent leurs images en employant des métadonnées EXIF (par exemple, le nom de l'appareil-photo) et de mots-clés (ajoutés manuellement par des utilisateurs) plutôt que l'utilisation du contenu HTML « possiblement » lié aux documents.

Les moteurs de recherche des outils de gestion de multimédias personnels reposent sur les mêmes principes que les *moteurs de recherche basés sur des collections.* La qualité des résultats (le rappel et la précision) possède une forte dépendance vis-à-vis de l'annotation, qui la plupart du temps, est manuelle et accompagnée des inconvénients présentés auparavant. Généralement, ces approches proposent une mise en correspondance simplement syntaxique qui présente les problèmes de polysémie et de synonymie inhérents [Gruber *et al.*, 2007]. En fait, les moteurs de recherche des outils de gestion de documents multimédias ne considèrent pas les relations sémantiques qui existent entre les documents et ses termes d'indexation.

L'usage d'ontologies à la fois lors de l'indexation des documents [Castells *et al.*, 2006][Garnaud *et al.*, 2006] et lors de la mise en correspondance [Koskela *et al.*, 2007] [Popescu *et al.*, 2007] améliore le rappel et la précision des résultats, notamment par le biais de l'augmentation de l'expressivité des requêtes et des annotations. L'intégration de l'ontologie linguistique WordNet est l'approche la plus répandue [Popescu *et al.*, 2007]. Les relations de synonymie, hyperonymie et hyponymie sont exploitées automatiquement pour l'expansion des termes d'une requête, ceux qui permet d'élargir la liste de résultats. Par exemple, l'outil RetrieveOnto [Popescu *et al.*, 2007] utilise un segment de Wordnet (1113 *synsets*) lié au concept *placental mammals* pour améliorer la précision des requêtes d'images Web à propos d'animaux. RetrieveOnto combine des requêtes par texte étendues à l'aide de WordNet à des mécanismes de traitement d'image pour filtrer des images non désirées.

Des ontologies typiquement construites pour le domaine de multimédia sont également exploitées lors de l'interrogation telles que les ontologies OntoAlbum [Chai *et al.*,2008] et LSCOM [Naphade *et al.*, 2006] présentées dans la section 3.2.3.4. Les requêtes dans ces propositions sont exprimées en langages standards d'interrogations d'ontologies (RDQL et SPARQL).

3.4.4 Recherche à l'aide d'informations contextuelles

Les approches de recherche de documents multimédias présentées jusqu'à ce point reposent sur une indexation des aspects liés au contenu de document multimédia et aux annotations manuelles ajoutées. En fait, les outils de gestion de multimédia basées sur les informations contextuelles (présentées dans la section 3.3) ne prennent pas en compte les métadonnées contextuelles lors du processus de recherche de documents multimédias. Les attributs sont majoritairement exploités pour l'organisation et pour la navigation au sein de la collection (à travers les séquences temporelles ou les cartes géographiques).

Seul le système MediAssist [O'Hare *et al.,* 2006] utilise l'annotation contextuelle dans la phase d'interrogation. Il offre une interface de recherche par texte et l'annotation dérivée d'une image est transformée en mots-clés d'indexation [O'Hare *et al.,* 2007]. La mise en correspondance est simplement syntaxique en utilisant la méthode de classement Okapi

BM25 [Robertson *et al.*, 1994]. L'inconvénient principal de cette approche est la non considération de la relation sémantique entre les photos et les index lors de la phase d'interrogation. Il a y donc une perte de la sémantique des index. Or, celle-ci peut s'avérer utile pour supprimer certaines ambiguïtés de mots telles que « April » (la personne ou le mois ?) ou « Paris » (la ville ou l'équipe de football de Paris Saint Germain ? Si c'est la ville, dans quel pays : la France, les États-Unis ou le Canada ?).

Plus généralement, puisqu'il peut s'écouler un certain laps de temps entre la prise et le moment de la recherche d'un document multimédia, les utilisateurs peuvent oublier certaines informations sur le contexte. De plus, lorsqu'ils se souviennent de certaines informations sur le contexte de création du contenu, ces informations peuvent être partiellement incorrectes ou différentes de celles automatiquement produites par le système d'annotation. Par exemple, si une photo était prise la fin de l'hiver, les utilisateurs peuvent demander cette photo, en pensant qu'elle a été prise au printemps.

Un autre aspect de **requêtes imprécises** est lié au manque de connaissance des utilisateurs par rapport aux attributs du contexte de création du document. Par exemple, un touriste qui prend une photo au Stade de France peut confondre le lieu de prise (Saint-Denis) avec la ville la plus connue des environs (Paris). Dans ce cas, l'utilisateur pourra essayer de rechercher cette photo avec le terme « Paris ». Par conséquent, il obtiendra des photos non désirées, puisque dans les systèmes d'annotation contextuelle de photos, l'endroit exact « Saint-Denis » est extrait en employant les coordonnées de GPS et les services « gazetteer » du web.

3.5 Partage de documents multimédias et d'informations contextuelles

Les personnes partagent leurs documents multimédias par e-mail, à travers des sites Web de médias sociaux (Facebook, Youtube, etc.), en imprimant leurs photos ou simplement en reproduisant des vidéos sur un écran LCD [Sarvas *et al.*, 2004b]. Une fois que les documents sont téléchargés sur les ordinateurs, les outils de gestion de multimédias personnels (tels que Apple Iphoto et Picasa) fournissent des services qui rendent plus aisées ces activités de partage.

La création de documents multimédias à l'aide de dispositifs mobiles bouleverse cet enchaînement de distribution de contenu car les utilisateurs nomades peuvent désormais partager ces documents directement à partir de leurs terminaux mobiles. Des applications commerciales et des projets de recherche n'ont pas tardé à apparaître pour profiter des avantages de communication de ces dispositifs afin d'élargir les services de partage de contenu multimédia [Sarvas *et al.*, 2004][Ames *et al.*, 2007][Raento *et al.*, 2005][Cemerlang *et al.*, 2006][Freyne *et al.*, 2007].

Dans cette section, nous nous intéressons à ces approches de distribution de contenu reposant sur l'usage de dispositifs mobiles et sur des informations concernant le contexte de création des documents. Nous présentons également des services mobiles pour lesquels le partage du contexte actuel de l'utilisateur est le but majeur.

3.5.1 Mobilité et sensibilité au contexte pour le partage de documents multimédias

Les dispositifs mobiles d'aujourd'hui possèdent davantage de moyens de distribution de contenu personnels qui stimulent le comportement de création de contenu (exclusivement) pour leur partage (*create-to-share behavior*, en anglais) [Sarvas *et al.*, 2004][Ames *et al.*, 2007]. Le protocole SMS sert de support à la publication de textes courts sur des blogs et des microblogs dont le site Twitter est le plus connu. Le MMS, le Bluetooth, l'infrarouge et, plus récemment, l'e-mail sur les mobiles, facilitent la distribution pair-à-pair d'images et de vidéos. Enfin, l'accès à l'internet par ces dispositifs permet la publication de contenus sur le

Web en peu de clics. Par exemple, l'application mobile ShoZu[36] (Figure 24) s'intègre à la majorité des sites de médias sociaux (Blogger, Facebook, Flickr, Youtube, etc.) en permettant l'envoi de texte, photos et vidéos directement à partir du téléphone. D'autres applications commerciales telles que l'Iphone Mobile Galery fournissent des services similaires.

Figure 24 - Interface de partage de multimédias du widget mobile ShoZu.

Bien que ces systèmes rendent plus facile le partage de contenus, les destinataires peuvent rencontrer des difficultés pour organiser les documents reçus. Par exemple, une image reçue par MMS ne contient pas d'annotations et n'appartient pas à une collection. Le projet de recherche MobShare [Sarvas *et al.*, 2004b] propose de résoudre ce problème en intégrant toutes les phases du partage de photos (de l'envoi à la réception) au sein d'une même outil. Par exemple, l'envoi d'une image par MMS est mis en œuvre par une application mobile qui transmet l'image en même temps au destinataire et à l'outil d'organisation de multimédias. Le répertoire du destinataire reçoit l'image et ses annotations ajoutées par l'expéditeur. MobShare offre en plus un mécanisme de notification par SMS qui communique à un utilisateur la publication d'une photo sur l'application Web d'un membre de son réseau social.

3.5.1.1 Sensibilité au contexte pour l'enrichissement du contenu partagé

La caractérisation du contexte d'utilisation de l'application mobile (localisation, temps, objets et personnes à proximité, etc.) est exploitée par plusieurs approches de partage de contenu [Sarvas *et al.*, 2004b][Ames *et al.*, 2007][Raento *et al.*, 2005][Cemerlang *et al.*, 2006][Hwang *et al.*, 2007]. Les projets de recherche Zonetag [Ames *et al.*, 2007] et MMM [Sarvas *et al.*, 2004a] utilisent les informations de localisation pour la suggestion d'annotations lors du partage d'une photo (tel que nous l'avons déjà présenté dans la section 3.3). D'autres approches reposent sur le contexte de l'utilisateur afin d'enrichir le contenu partagé. Par exemple, le projet Aware[37] remplace l'application d'envoi de MMS des téléphones Nokia par une autre version (ContextMedia) qui s'exécute au dessus d'une plate-forme sensible au contexte nommée ContextPhone (Figure 25) [Raento *et al.*, 2005].

Cette plate-forme mobile offre des bibliothèques pour l'interrogation de capteurs de localisation (capteur GPS couplé par Bluetooth ou découverte de l'identificateur de la cellule GSM) et pour la création (prise de photo, enregistrement de vidéos) et partage des multimédias (SMS, MMS, Jabber, etc.). L'application ContextMedia ajoute automatiquement à chaque MMS envoyé par l'utilisateur des informations sur sa localisation (une adresse dérivée à partir de l'identificateur de la cellule GSM et d'une base de données du serveur de ContextPhone) et sur les dispositifs Bluetooth détectés à proximité. La Figure 25 illustre un exemple de MMS généré. L'application utilise le nom publié par les dispositifs Bluetooth (le

[36] www.shozu.com

[37] http://aware.uiah.fi

friendlyname) et sa pile de protocoles afin de le catégoriser en téléphone mobile d'une personne ou en ressource dotée d'une interface de communication Bluetooth (telle qu'une imprimante).

Le projet de recherche Mobilog [Cemerlang *et al.*, 2006] est plutôt centré sur l'usage des informations contextuelles pour la génération semi-automatique de messages multimédias (photos et textes) sur un blog à propos de Singapour. L'objectif est de réduire le temps d'écriture des messages par les utilisateurs en leurs proposant sur leurs téléphones mobiles des phrases générées automatiquement. Mobilog utilise le système de SnapToTell [Chevallet *et al.*, 2005] afin de détecter sur la photo de l'utilisateur des monuments touristiques de la ville de Singapour. En cas de la détection d'un monument de Singapour, la phrase « I went to DetectedObject » est ajoutée automatiquement au Blog. La localisation de l'objet détecté est ensuite utilisée pour l'agrégation d'autres informations telles que l'adresse du monument et la météo de la journée.

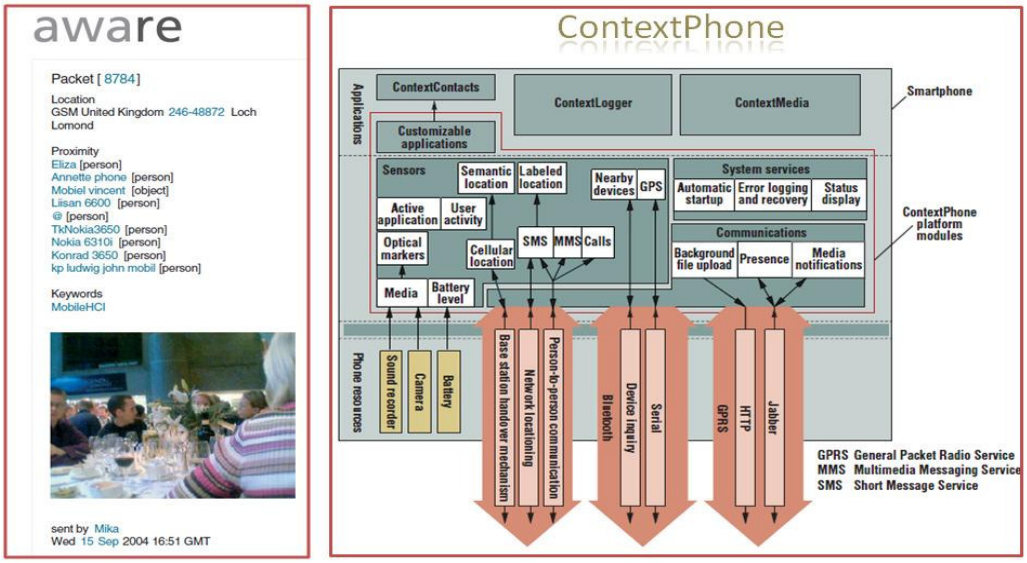


Figure 25 - Exemple de MMS enrichi par des informations automatiques sur le contexte de l'utilisateur capturées à l'aide de la plate-forme ContextPhone [Raento *et al.*, 2005].

3.5.1.2 Partage sensible au contexte de documents multimédias

L'information du contexte de l'utilisation peut étendre également le processus de partage lui-même car les informations sur le contexte des expéditeurs et des destinateurs peuvent être exploitées pour l'établissement de nouveaux canaux de communication et pour la définition de règles de distribution de contenu [Hwang *et al.*, 2007] [Freyne *et al.*,2007].

Par exemple, le projet de recherche Zurfer [Hwang *et al.*, 2007] de l'Université de Berkeley offre une application mobile de recommandation de photos du site Flickr. Le logiciel mobile capture la localisation de l'utilisateur et lui-montre les éléments photographiés par d'autres personnes lorsqu'elles sont passées près de cet endroit. Pour arriver à cette recommandation, Zurfer sélectionne les photos géoréférencées de Flickr prises à proximité, et il démarre ensuite une méthode d'analyse de contenu afin de supprimer les photos contenant des visages.

Le logiciel CAMM (*Contexte-Aware Mobile Message*) est une architecture Java pour la diffusion de messages sensibles au contexte (MMS, SMS) [Freyne *et al.*, 2007]. CAMM fournis aux utilisateurs un niveau augmenté de contrôle de leurs messages. L'application mobile permet de définir des conditions basées sur le contexte des destinataires pour la réception d'un message. Par exemple, un expéditeur de MMS décide que seulement d'autres utilisateurs de CAMM situés au même endroit que lui peuvent recevoir le message à ce

moment précis. Le serveur de CAMM calcule alors quels utilisateurs remplissent la condition de diffusion. La localisation des utilisateurs est capturée par un balayage des dispositifs Bluetooth voisins.

3.5.2 Partage d'informations contextuelles

Le partage instantanée d'informations sur les situations actuelles des personnes (où ils sont, quelles activités ils exercent, à quoi ils pensent, etc.) est la nouvelle tendance des services de médias sociaux dont les microblogs Jaiku et Twitter sont les plus connus. Malgré l'impression, à première vue, d'attentent à la vie privé, ces applications ont pour but d'augmenter la communication et le rapprochement d'individus qui possèdent des intérêts similaires [Barkhuus *et al.*, 2008].

Une grande partie des messages partagés sur ces sites fait référence à des informations sur le contexte des utilisateurs et peuvent, ainsi, être dérivées automatiquement à l'aide des dispositifs mobiles élargis par des capteurs variés [Barkhuus *et al.*, 2008]. Connecto [Barkhuus *et al.*, 2008] et ContextWatcher [Koolwaaij *et al.*, 2006] sont exemples de ce genre de proposition.

Connecto [Barkhuus *et al.*, 2008], développé au sein de l'Université de San Diego, offre la possibilité d'informer l'humeur (heureux, triste, etc.), les activités (en réunion, en silence, en train de manger, etc.) et une information de localisation de haut niveau sémantique (au travail, à la maison, etc.). La Figure 26 illustre les interfaces de l'application mobile. L'utilisateur peut attribuer un nom à sa localisation courante (en fait, sa cellule GSM) qui sera réutilisée chaque fois qu'il sera localisé dans cette même région. Les informations sur l'activité sont dérivées à partir de son agenda ou renseignées manuellement.

Figure 26 - Le système CONNECTO de partage de location et d'activités [Barkhuus *et al.*, 2008].

ContextWatcher [Koolwaaij *et al.*, 2006] est l'application mobile pour la capture et le partage des informations contextuelles la plus répandue. L'objectif est d'acquérir et de décrire avec une grande précision la situation actuelle de l'utilisateur. Le contexte inclut la localisation (i.e., coordonnées géographiques, altitude et adresse), la vitesse de déplacement, l'humeur, les fréquences des battements cardiaques et les conditions météorologiques de l'environnement de l'utilisateur. ContextWatcher offre des services de partage de cette information contextuelle avec d'autres utilisateurs. Par exemple, un utilisateur peut voir la localisation des membres de sa famille ou si ses invités sont proches de chez lui.

ContextWatcher a été développé à l'aide du *framework* de gestion de contexte : le MobiLife CMF[38]. Ce *framework* permet la découverte semi-automatique des fournisseurs

[38] http://www.lab.telin.nl/~koolwaaij/showcase/crf/

d'information contextuelle dans un environnement pervasif (par exemple, plusieurs capteurs dans une salle). Le *framework* contient un protocole de communication basé sur le standard SOAP qui permet la mise en œuvre, la diffusion et le partage des données des capteurs entre différents fournisseurs (l'accès aux données contextuelles d'un autre utilisateur). Une application utilisant MobiLife CMF peut contrôler l'accès à un fournisseur de contexte et elle peut définir dynamiquement l'agrégation entre les données de plusieurs fournisseurs afin de dériver des données contextuelles plus complexes (la météo de l'endroit où se trouve un ami).

Le système ContextWatcher est développé sur la plate-forme Python et peut être déployé sur les dispositifs Nokia Séries 60. L'application utilise les métadonnées contextuelles pour permettre la génération automatique des blogs journaliers (e.g. http://koolwaaij.blogspot.com/). ContextWatcher capture automatiquement la localisation (i.e., les coordonnées GPS, l'ID de la cellule GSM) et génère un rapport avec le pourcentage du temps que l'utilisateur a passé sur un endroit ou une ville. De plus, il permet à l'utilisateur d'associer des mots-clés aux dispositifs *Bluetooth* reconnus dans l'environnement et, par la suite, il génère un graphique illustrant combien de fois chaque dispositif a été repéré.

L'utilisateur configure manuellement les informations contextuelles qui seront capturées et partagées avec son réseau social. Par exemple, l'utilisateur peut partager avec ses amis son humeur, sa localisation, la météo (voir Figure 27). Ces informations peuvent aussi être exportées en plusieurs formats de partage (e.g., KMZ, XML, EXIF) et être transmises à un serveur afin d'être visualisées sur des applications Web 2.0 telles que Google Earth et Flickr.

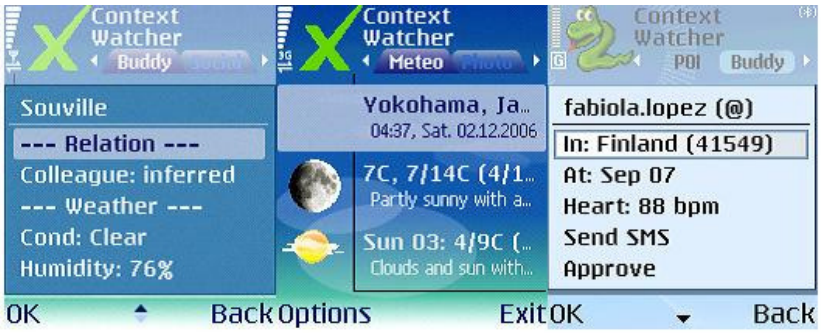


Figure 27 - Partage d'informations contextuelles.

Depuis 2008, ContextWatcher est relié au site IYOUIT[39] qui offre une interface de visualisation Google Maps enrichie avec les données partagées par la communauté MobiLife. Par exemple, des messages textuels, les utilisateurs, la météo, les dispositifs Bluetooth détectés peuvent être géoréférencés sur la carte de la terre.

3.6 Synthèse

Dans ce chapitre, nous avons présenté un large panorama des propositions de gestion de contenus multimédias en mettant en évidence quatre aspects : **organisation**, **annotation**, **recherche** et **partage**. Une attention spéciale aux approches reposant sur la mobilité a été également prêtée. La Figure 28 illustre un nuage des termes les plus utilisés dans ce chapitre. Les principales sections du chapitre traitent de la représentation et de la découverte automatique de métadonnées relatives à la fois au contenu et au contexte de création des documents multimédias.

[39] http://www.iyouit.eu/portal/

Concernant les techniques de représentation des métadonnées, les ontologies semblent mieux répondre aux problèmes d'expressivité et d'interopérabilité des approches d'annotations classiques (telles que l'usage des mots-clés). Cependant, les ontologies destinées au domaine de gestion de documents multimédias n'offrent quasiment pas de description dédiée au contexte de création des documents multimédias personnelles. Elles sont principalement destinées à la caractérisation du contenu et sont exploitées pour une annotation extensive et manuelle des documents.

Figure 28 - Nuage des termes de ce chapitre généré à l'aide du logiciel Wordle[40].

Pour l'organisation et l'annotation automatique de documents multimédias, nous semble que les propositions reposant sur la caractérisation du contexte de création des documents sont plus prometteuses. Les métadonnées contextuelles peuvent être plus facilement découvertes, et les études d'utilisabilité, présentées dans la section 3.3, montrent les avantages de l'usage de ces métadonnées pour la gestion de documents multimédias.

Le Tableau 2 présente une synthèse des principales approches de gestion sensible au contexte de documents multimédias. La plupart des solutions repose sur l'utilisation de métadonnées EXIF générées par des appareils photos numériques. Certaines approches (telles que ZoneTag, Aware et ContextWatcher) proposent néanmoins des applications mobiles pour capturer davantage d'informations associées au contexte de création de documents. Bien que ces outils offrent plus de métadonnées que ces présentes sur les fichiers EXIF, ces solutions s'appuient très souvent sur des informations contextuelles quasiment symboliques (telles que l'ID de la cellule GSM) ou renseignées manuellement par les utilisateurs (telles que leurs activités). La capture et la gestion de l'information contextuelle sont directement intégrées au code des applications. Ceci limite également leur extension et leur portabilité sur des modèles de dispositifs mobiles différents.

Concernant la représentation des métadonnées des approches du Tableau 2, nous pouvons constater qu'EXIF est utilisé uniquement comme paramètre d'entrée des outils de gestion en raison de son vocabulaire limité en termes de description des annotations. Dans le Tableau 2, nous pouvons voir que la majorité des propositions emploie des représentations internes aux outils de gestion de multimédias afin d'exprimer les nouvelles informations inférées, rendant plus difficile, l'interopérabilité de ces métadonnées. Malgré l'existence aujourd'hui d'un nombre important d'approches de gestion sensible au contexte de multimédias, les métadonnées contextuelles sont peut exploitées lors de l'activité de recherche textuelle des documents. Seul MediaAssist et Zonetag considèrent ces attributs lors de la phase

[40] http://www.wordle.net

d'interrogation. Cependant, les relations sémantiques entre les annotations contextuelles et les documents sont perdues durant le processus d'indexation de ces approches.

Exploiter une représentation sémantique et enrichie du contexte de création dans le processus d'indexation et d'interrogation peut s'avérer utile pour la réduction des problèmes d'ambigüité et d'imprécision des requêtes formulées par les utilisateurs.

Tableau 2 - Synthèse des approches de gestion sensible au contexte de documents multimédias.

Approche	Métadonnées	Stockage de Métadonnées	Acquisition de Métadonnées	Exploitation
Photo Compas	Date, localisation, météo, saison et température	Représentation interne	• Capture : Fichier EXIF • Enrichissement : Web Services	• Organisation
MédiaAssist	Date localisation, météo, saison, température, informations sur le contenu	Représentation interne	• Capture : Fichier EXIF • Enrichissement : Web Services et classification du contenu	• Organisation • Recherche textuelle par comparaison syntaxique
ZoneTag	Mots-clés	Représentation interne	• Capture : application mobile • Enrichissement : Mesures de similarité appliquées à des images annotées préalablement	• Suggestion d'annotations • Recherche textuelle par comparaison syntaxique
PhotoCopain	Date, localisation, informations sur le contenu	RDF	• Capture : Fichier EXIF • Enrichissement : Gazetteers et classification du contenu	• Annotation des images
PhotoGeo	Date, localisation, et nom d'événement	Représentation interne	• Capture : Fichier EXIF • Enrichissement : Gazetteers, Google Calendar, et méthodes de regroupement	• Organisation par événements
Context Watcher	Date, localisation, bâtiments cardiaque, météo, dispositifs Bluetooth à proximité	Représentation interne	• Capture : application mobile et information manuelle • Enrichissement : Web Services et Gazetteers	• Organisation • Partage de photos et d'information contextuelle
OntoAlbum	Personnes présentes sur le document	OWL	• Capture : information manuelle • Enrichissement : règles d'inférence	• Organisation
Aware	Date, localisation et dispositifs Bluetooth à proximité	Représentation interne	• Capture : application mobile et information manuelle	• MMS enrichi

4 L'INFORMATIQUE MOBILE

Les dispositifs mobiles modernes présentent de nombreuses avantages pour devenir des plates-formes idéales de création de multimédias personnels [Naaman *et al.*, 2004][Raento *et al.*, 2005]. Ces terminaux sont portés quotidiennement et disposent de nombreuses fonctionnalités d'enregistrement, d'édition et de reproduction de contenus (image, vidéo et audio). Les dispositifs mobiles, malgré des limitations de ressources (capacité de calcul et de mémoire, durée de vie de la batterie, etc.), ont l'avantage d'intégrer un nombre croissant de capteurs et de sources de données sur l'usager (agenda, contacts, …), comme décrit dans le chapitre 2. Également, ces dispositifs mobiles acceptent l'installation de nouvelles applications. Il y a donc la possibilité de développer de logiciels qui tirent profit des sources d'information et des fonctionnalités multimédias pour l'amélioration de la gestion de contenus multimédias personnels dans toute sa globalité (organisation, annotation, recherche et partage).

Cependant, en réalité, l'hétérogénéité des dispositifs mobiles et les contraintes imposées par leurs environnements d'exécution rendent difficiles la conception et le déploiement d'applications multimédias et multi-capteurs. Surtout, lorsque ces applications mobiles (i.e. applications susceptibles d'être activées sur un dispositif mobile) sont utilisées sur un grand nombre de modèles de dispositifs.

Ce chapitre propose un état de l'art du développement et du déploiement d'applications pour ces dispositifs, permettant de juger des difficultés et limites actuelles.

Tout d'abord, nous présentons les principales plates-formes de programmation d'applications mobiles. Le chapitre décrit particulièrement les plates-formes destinées à la création d'applications de type client lourd dont la majorité du code est mise en œuvre localement sur le dispositif mobile. Ensuite, nous nous intéressons aux étapes du processus de déploiement d'applications mobiles sous ces plates-formes. Un aperçu des obstacles engendrés par l'hétérogénéité de dispositifs mobiles est également présenté.

4.1 Développement sur les plates-formes mobiles

Cette section introduit le développement d'applications destinées aux dispositifs mobiles. À l'instar du développement d'applications pour les ordinateurs de bureau, plusieurs environnements de programmation sont disponibles aujourd'hui pour la création

d'applications mobiles. Ces environnements de programmation visent principalement la création de deux catégories de logiciels : les applications purement Web, et les applications mobiles d'exécution locale.

Nous décrivons les outils utilisés pour le développement de ces deux catégories d'applications tout au long de cette section qui se découpe en trois parties. Initialement, nous présentons le développement d'applications mobiles Web. Ces applications s'exécutent à l'intérieur d'un navigateur Web installé sur les dispositifs mobiles et sont codifiées généralement à l'aide de langages informatiques de balisages (tels que XHTML et WML). Nous détaillons notamment les raisons pour lesquelles cette catégorie d'application n'est pas la plus adapter au développement d'applications multimédias et multi-capteurs envisagées dans notre proposition de thèse.

Dans la seconde partie de la section, nous nous intéressons à la deuxième catégorie d'applications dites d'exécution locale. Ces applications ne dépendent pas de la présence d'un browser, même si, éventuellement, elles peuvent accéder à des contenus et des services disponibles sur le Web. En réalité, la majorité de leur code s'exécute directement sur des plates-formes logicielles intégrées aux dispositifs mobiles. Ces plates-formes mobiles sont, soit les propres systèmes d'exploitation des terminaux (tels que Windows Mobile et Symbian OS), soit des machines virtuelles embarquées (tel que la machine virtuelle de Java Mobile). Nous présentons les principales plates-formes mobiles existantes et les outils disponibles pour le développement d'applications destinées à ces plates-formes.

4.1.1 Les applications Web pour les dispositifs mobiles

Après le lancement de la technologie WAP (Wireless Application Protocol) en 1999, le développement d'applications Web adressées exclusivement aux dispositifs mobiles n'a fait que croître. L'intégration des capacités de communication dans les terminaux mobiles permet désormais l'accès à des applications Web codées en langages à balises tels que WML (Wireless Markup Langage), VoiceXML, et plus récemment, XHTML et Adobe Flash Lite. Des émulateurs de navigateurs mobiles sont fournis par les fabricants de dispositifs et offrent la possibilité de tester le comportement des pages sans avoir besoin d'un terminal mobile en mains. Les langages à balises utilisés pour développer des applications mobiles sont plus limités que les langages Web traditionnels. Ils possèdent des limitations principalement au niveau des scripts et des feuilles de styles. Cependant, certains langages (tels que WML) offrent des méthodes d'accès aux fonctionnalités spécifiques des téléphones mobiles. Par exemple, un lien en WML peut déclencher un appel téléphonique ou un envoi de SMS [Shresta, 2007]. Des navigateurs Web sont installés en usine par les constructeurs sur les dispositifs mobiles. Ces navigateurs agissent comme de simples clients légers de l'application en interprétant le code des pages. La partie lourde du code métier des applications est mise en œuvre du côté du serveur Web du système d'information.

Les développeurs des pages Web mobiles sont confrontés à deux difficultés majeures : i) le support hétérogène offert par les navigateurs mobiles envers les langages de script et envers les contenus multimédias (audio, image, animation) ; et ii) la diversité des tailles d'écran et des modes d'interaction (écran tactile, curseur, stylet, clavier, etc.) des dispositifs mobiles [Gajos *et al.*, 2004] [Viana *et al.*, 2008a]. Afin de contourner le problème de l'hétérogénéité des dispositifs, les développeurs sont fréquemment obligés de créer une version d'une page Web quasiment pour chaque modèle de terminal mobile. Au début du Web mobile, ces inconvénients limitaient la quantité de pages produites spécialement pour l'accès nomade en raison du coût de développement qui était très élevé par rapport au nombre d'utilisateurs potentiel [Buttler *et al.*, 2002].

Nombreux sont les travaux de recherche qui ont vu le jour depuis la naissance du WAP autour du développement de pages Web et de documents multimédias susceptibles d'être accédés par les dispositifs mobiles [Puerta, 2005][Paterno *et al.*, 2004][Lemlouma, 2004][Chu *et al.,* 2004][Gajos *et al.*, 2004][Chaari *et al.*, 2005][Bilasco *et al.*, 2005][Muller *et al.*, 2005][Viana *et al.*, 2008a]. L'objectif principal de ces approches est de rendre aux utilisateurs nomades les documents Web et des contenus multimédias (images, audio, etc.) les plus adaptés possibles aux caractéristiques logicielles et matérielles de leurs dispositifs d'accès. Ces propositions tentent également de réduire le besoin de création manuelle de multiple versions des ressources des applications mobiles (telles que les pages et les images). Nous pouvons classer ces propositions en deux catégories : i) *les approches de génération de versions* ; et ii) *les approches d'adaptation par proxy*. La Figure 29 illustre des schémas récurrents de flux de données au sein de ces deux catégories de propositions.

Généralement, *les approches de génération de versions* possèdent des langages à balises génériques pour la description d'une page Web. Ces langages sont indépendants d'une technologie spécifique et permettent une description des interfaces dissociée de leurs présentations finales. De plus, les interfaces sous ces langages peuvent facilement être transcodées en d'autres langages de balisages. UIML (User Interface Markup Language)[41] , XIML [Puerta, 2005], XML AWD (Abstract Window Description) [Chaari *et al.*, 2005] sont des exemples de ces langages génériques. *Les approches de génération de versions* offrent également aux développeurs des outils de génération de documents XHTML, WML et VoiceXML à partir des interfaces initialement décrites sous les langages génériques [Puerta *et al.*, 2005][Gajos *et al.*, 2004]. Certaines approches telles que TERESA [Paterno *et al.*, 2004] et XMobile [Viana *et al.*, 2008a] permettent en plus de prendre en compte les caractéristiques des dispositifs cibles (taille d'écran, format des images acceptés, etc.) pour un réglage plus fin du processus de génération. À partir d'une unique définition de l'interface d'une application, ces propositions génèrent plusieurs versions, chacune adaptée à une classe de dispositifs. Par exemple, une version en WML est générée pour l'affichage sur tous les dispositifs qui possèdent une taille d'écran inférieur à 180x220 et n'offrent pas le support à la technologie XHTML (tel que le dispositif K750i de Sony Ericsson).

Un mécanisme de classification des dispositifs est mis en place (voir Figure 29-I) lors de l'accès à un site Web créé à l'aide de ces approches génératives, [Paterno *et al.*, 2004] [Muller *et al.*, 2005]. Une réquisition WAP ou HTTP contient dans son entête des métadonnées qui indiquent le nom du navigateur Web et quelques informations sur le dispositif d'accès. Un des attributs de cette réquisition est l'entête « User Agent » qui indique le nome du navigateur et le type du dispositif d'accès. Le navigateur du dispositif K750i envoie, par exemple, «SonyEricssonK750i/R1A Browser/SEMC-Browser/4.2 Profile/MIDP-2.0 Configuration/ CLDC-1.1 » comme valeur de cette métadonnée. Le serveur d'une application utilise cette information pour chercher la description du dispositif dans un registre de profils de terminaux mobiles (voir Figure 29-I). Diverses technologies existent pour la mise au point de ce genre de registre de profils telles que CC/PP, UAProf et WURFL. Ces technologies seront étudiées plus en détails dans le chapitre suivant. Une fois que le profil du dispositif est identifié, le serveur exploite les caractéristiques matérielles et logicielles du dispositif afin de le classer (par exemple, dispositif WML doté d'un mini écran). Le serveur peut ensuite sélectionner la version de la page Web créée spécialement pour cette catégorie de dispositif (sur la Figure 29-I, la version 1 a été choisie) [Muller *et al.*, 2005].

[41] http://www.uiml.org/

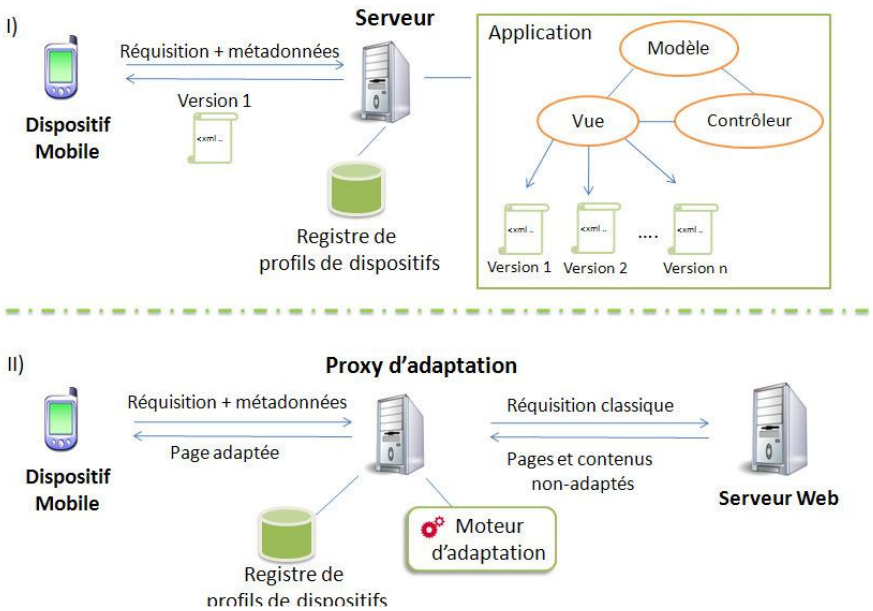


Figure 29 - Schémas de flux de données lors d'accès à une page Web : I) au sein des approches de générations de versions; II) au sein des approches de proxy d'adaptation.

Les *approches d'adaptation par proxy* n'exigent pas la création de versions d'une application Web avant sa mise en ligne. Dans ces approches, le navigateur mobile obtient les pages Web par l'intermédiaire d'un proxy (Figure 29-II). Les objectifs du proxy sont d'une part de découvrir les caractéristiques du dispositif d'accès, et, d'autre part, d'adapter le rendu des pages et les éventuels contenus multimédias (images, animation, etc.) aux contraintes de diffusion du dispositif d'accès [Lemlouma, 2004][Lum *et al.,* 2003]. À l'instar *des approches de génération de versions*, ces propositions s'appuient sur un registre de profils pour la découverte des caractéristiques des dispositifs. Les métadonnées des connexions WAP ou http sont également utilisées pour l'identification du modèle du terminal d'accès.

Après avoir reçu une demande d'une page ou d'un contenu multimédia, le proxy d'adaptation utilise le registre afin d'identifier le dispositif. Ensuite, le proxy entame un mécanisme de négociation avec le détenteur du contenu ou le serveur de la page Web. Le but de cette opération est de trouver un consensus entre les ressources disponibles sur le serveur et les contraintes d'affichage du terminal mobile. Le proxy demande au serveur des versions des contenus (image, vidéo, etc.) et de la page Web qui respectent les contraintes du dispositif. Dans l'absence de ces versions, le proxy démarre des processus d'adaptation de la page et des contenus déjà mis à disposition par le serveur. Le proxy peut exploiter plusieurs techniques d'adaptation telles que la transformation du contenu original, la suppression des balises du document, la compression des images et la dégradation du contenu [Lemlouma, 2004]. Nonobstant la surcharge du processus de réquisition d'une page Web, les *approches d'adaptation par proxy* offrent la possibilité aux utilisateurs nomades d'accéder à des sites Web qui n'étaient pas initialement prévus pour un accès mobile.

Malgré les progrès apportés par ces deux catégories d'approches, les applications mobiles les plus abouties restent encore celles qui s'exécutent directement sur les plates-formes mobiles intégrées aux dispositifs mobiles. De plus, ces applications n'ont pas besoin d'un navigateur internet comme intermédiaire. Les technologies Web les plus utilisées par les navigateurs mobiles n'offrent pas encore des méthodes d'accès aux fonctionnalités multimédias, par exemple pour prendre une photo ou enregistrer un message audio. L'acquisition d'informations qui proviennent des capteurs intégrés aux dispositifs est également très limitée. Par exemple, il n'existe pas une fonction WML ou JavaScript qui donne la localisation de l'utilisateur. Seuls certains opérateurs mobiles ajoutent des

métadonnées de positionnement géographique dans l'entête des connexions WAP et HTTP. Néanmoins, l'accès à ces informations est limité aux applications fournies par l'opérateur.

L'utilisation de plates-formes Web pour développer des applications multimédias et multi-capteurs est un choix de second ordre. Cependant, l'intérêt d'avoir étudié ces plates-formes et leurs extensions réside dans le fait que leurs langages de description de modèles de terminaux mobiles et leurs méthodes d'identification d'un profil peuvent être réutilisés dans une solution visant à améliorer le déploiement d'une application multimédia sur un grand nombre de modèles hétérogènes de dispositifs mobiles.

4.1.2 Les plates-formes logicielles mobiles et leurs kits de développement

L'arrivé des plates-formes logicielles ouvertes aux terminaux mobiles a permis l'installation d'applications tierces sur ces dispositifs et, par conséquent, a incité la création d'applications mobiles à exécution locale. Ces logiciels sont développés à l'aide de nouvelles plates-formes de programmation qui augmentent l'interactivité entre les utilisateurs et les applications, puisque ces plates-formes offrent davantage de ressources pour la conception d'interfaces et l'accès aux nouvelles fonctionnalités multimédias des terminaux (actionner la prise de photos ou de vidéos, le MMS, le GPS, le MP3, etc.). Il existe de nombreux outils pour développer, déboguer et tester de telles applications. Ces outils sont rassemblés dans des kits de développement (en anglais, *SDK - Software Development Kit*). Ces SDK permettent une complète implémentation des applications sur des ordinateurs de bureau aussi bien que la simulation des applications en utilisant des émulateurs de dispositifs mobiles réels. Nous identifions deux classes d'applications conçues à l'aide de ces kits : les applications natives et les applications indépendantes d'un système d'exploitation.

Les *applications locales natives* sont des logiciels créés à l'aide de plates-formes de programmation étroitement liées à un unique système d'exploitation de dispositifs mobiles. En réalité, chaque système d'exploitation fournit son propre SDK qui rend possible le développement d'applications natives. Ces applications s'exécutent exclusivement sur les terminaux mobiles qui possèdent le système d'exploitation cible. Parmi les systèmes d'exploitation existants, Symbian OS, iPhone OS, Windows Mobile, Android, Linux et le Web OS (ancien Palm OS) sont les plus répandus.

Les *applications* (théoriquement) *indépendantes du système d'exploitation* sont des logiciels créés sur des plates-formes de programmation qui reposent sur les principes des machines virtuelles. Les compilateurs des kits de développement ne génèrent pas un binaire spécifique à un système d'exploitation mobile. Ainsi, une application codée dans ce genre de plate-forme pourra fonctionner sur divers systèmes d'exploitation. Il suffit qu'ils disposent d'une machine virtuelle installée capable d'interpréter le code de l'application. J2ME MIDP/CLDC, DOJA[42], TotalCross (ancien Superwaba)[43], et Satellite Forms[44] sont des exemples de plates-formes mobiles qui fonctionnement sur plusieurs systèmes d'exploitation.

Nous avons étudié plus en détail les cinq principales plates-formes mobiles : J2ME MIDP/CLDC, Symbian OS, iPhone OS, Windows Mobile et Android. Cette étude est présentée dans l'Annexe 1. Nous présentons également un tableau récapitulatif des caractéristiques de ces cinq plates-formes logicielles.

[42] http://www.nttdocomo.co.jp/english/service/imode/make/content/iappli/

[43] http://www.superwaba.com.br

[44] http://www.satelliteforms.net/

4.2 Synthèse du développement d'applications mobiles

Dans le but de développer une application mobile, le choix de la plate-forme de programmation influe directement sur sa performance future (vitesse d'affichage et d'exécution), sur les fonctionnalités susceptibles d'être développées, et sur le nombre de modèles de dispositifs dans lesquels l'application pourra être exécutée. Par exemple, Android et Symbian OS proposent plus de contrôle sur les propriétés bas niveau du dispositif (telles que l'accès à la carte réseau Wi-Fi), et une vitesse d'exécution légèrement supérieure à celle proposée par J2ME MIDP. Cependant, la quantité de modèles de dispositifs et de terminaux mobiles sur le marché qui disposent de Symbian OS et Android est nettement inférieure à celle des terminaux dotés de J2ME MIDP (plus de 8000 modèles et plus d'un milliard de dispositifs vendus[45]).

Du point de vue du développeur, les plates-formes mobiles possèdent également des dissemblances au niveau de la courbe d'apprentissage de l'environnement de programmation, et au niveau de la quantité disponible d'outils et de documentations (émulateurs, livres, tutoriels, forums, etc.) [Wesson *et al.*, 2005]. Par exemple, l'apprentissage d'Android et de J2ME MIDP est facilité par la connaissance de Java par la majorité de développeurs. Néanmoins, Windows Mobile offre des environnements de programmation plus aboutis que ceux proposés pour J2ME MIDP et Android.

Malgré les différences entre ces cinq plates-formes, toutes, dans une certaine mesure, permettent la création d'applications multimédias et multi-capteurs. Les bibliothèques de ces plates-formes proposent, par exemple, l'accès à l'appareil photo/vidéo ainsi que la lecture du positionnement géographique et l'orientation spatiale de l'utilisateur. Les grands inconvénients du développement d'applications mobiles d'exécution locale restent [Chu *et al.,* 2004][Jiang *et al.*, 2006][Preuveneers *et al.*, 2007] :

- *Les capacités de calcul et de mémoire encore limitées*. Bien que les nouveaux dispositifs mobiles sont mieux armés en termes de capacités que les premiers terminaux mobiles, les nouvelles fonctionnalités attendues par les utilisateurs ont une consommation de ressources supérieure à celle des premières applications mobiles. Par exemple, la manipulation de multimédia (ouvrir une photo, reproduire une vidéo, etc.) et l'acquisition en boucle des données de capteurs peuvent facilement entraîner une occupation importante de mémoire et/ou ralentir l'exécution d'une application [Jiang *et al.*, 2006]. De plus, les plates-formes telles que J2ME, iPhone OS et Windows Mobile ne libèrent pas toute la mémoire du dispositif pour une seule application. Un développeur non averti peut ainsi avoir l'impression trompeuse qu'un dispositif ayant beaucoup de mémoire pourra supporter la charge de son application.

- *La consommation excessive de batterie*. L'écran en couleur, la communication à travers le réseau, les fonctions multimédias (flash de l'appareil photo, baladeur numérique) et les capteurs de positionnement (GPS, A-GPS et Wi-Fi), sont gros consommateurs d'énergie. Par exemple, des expérimentations d'usage d'un GPS connecté via Bluetooth au dispositif Sony Ericsson K750i[46] réalisées qui nous avons réalisé ont montré qu'en moyenne la batterie est déchargée en moins de trois heures d'utilisation. L'activation des capteurs, combinée à l'utilisation de fonctions multimédias, peut entraîner une surchauffe des terminaux mobiles[47]. Les développeurs doivent ainsi

[45] http://java.sun.com/javame/index.jsp

[46] http://www.sonyericsson.com/cws/products/mobilephones/overview/k750i

[47] http://www.bestofmicro.com/actualite/26985-iphone-3G-S.html

éviter au maximum d'activer ces fonctionnalités simultanément [Preuveneers *et al.*, 2007].

- *L'hétérogénéité des modèles de dispositifs*. L'hétérogénéité des dispositifs mobiles peut être caractérisée par la différence entre les propriétés matérielles (la mémoire, la vitesse du processeur, les capteurs intégrées, etc.) et les propriétés logicielles (le système d'exploitation, les bibliothèques disponibles, la version des machines virtuelles) [Chu *et al.,* 2004]. Les applications multimédias et multi-capteurs sont les plus touchées par l'hétérogénéité car elles accèdent à des fonctions et à des ressources qui existent sous diverses formes sur les dispositifs (par exemple, certains appareils sont équipés de GPS, d'autres non).

Parmi ces trois inconvénients vis-à-vis la création d'applications mobiles, l'hétérogénéité est celui qui impacte le plus sur le coût de développement. La dissemblance entre les modèles de terminaux est déjà remarquable même au sein de la plate-forme iPhone OS qui compte seulement quatre modèles de dispositifs (Tableau 3).

Tableau 3 - Un comparatif entre les terminaux mobiles disposant de l'iPhone OS .

Dispositif	GPS	Capture de vidéo	Prise de photo	Accéléromètre	MMS
iPod Touch				X	
iPhone			X	X	
iPhone 3G	X		X	X	
iPhone 3G S	X	X	X	X	X

La dissemblance entre les modèles de terminaux n'est pas propre à l'iPhone OS. Nous pouvons constater (voir Annexe 1) que les plates-formes de programmation d'applications mobiles sont des environnements très hétérogènes, pour lesquels le support des bibliothèques (classes, API, méthodes) change selon entre les dispositifs. La présence de certaines fonctionnalités dépend de la version du système d'exploitation ou de la machine virtuelle, et des caractéristiques matérielles du terminal mobile cible. Ce support hétérogène des plates-formes rend difficile la conception d'applications exécutables par une grande diversité de dispositifs. À l'instar du développement d'applications mobiles purement Web, lutter contre l'hétérogénéité exige de la part des développeurs la construction de différentes versions d'une application mobile pour chaque couple potentiel (modèle de dispositif, plate-forme). Cela n'est clairement pas envisageable alors que le nombre de modèles croît constamment. Dans le prochain chapitre, nous exposons des propositions de recherche qui tentent d'alléger la tâche du développement d'applications mobiles à l'aide de techniques d'adaptation.

4.3 Déploiement d'applications sur les plates-formes mobiles

Le terme « déploiement » est fréquemment utilisé en informatique comme un synonyme d'installation, principalement lorsque les logiciels à mettre à disposition reposent sur des ressources multiples et complexes (base de données, serveur Web, etc.). Plus exactement, le processus de déploiement fait référence à toutes les phases du cycle de vie observable pour une application après son développement, qui incluent son installation, sa maintenance et son évolution [Benamar *et al.*, 2008]. Initialement, le déploiement d'applications tierces sur les téléphones mobiles était inexistant. Les systèmes d'exploitation de l'époque étaient fermés et aucune application ne pouvait être déployée. Aujourd'hui, les plates-formes mobiles (présentées dans la section précédente) offrent plusieurs manières de déployer des applications tierces et de contrôler leur cycle de vie. Dans cette section, nous survolons les mécanismes de déploiement fournis par les principales plates-formes mobiles.

4.3.1 Les étapes du processus de déploiement d'applications

Selon Carzaniga [Carzaniga *et al.*, 1998], les étapes du processus de déploiement recouvrent entre autres : la configuration, le packaging, l'installation, l'activation, l'adaptation (ou reconfiguration), la mise à jour, la désactivation et la désinstallation. La Figure 30 illustre un processus générique de déploiement et les interactions entre les différentes étapes du cycle de vie d'une application.

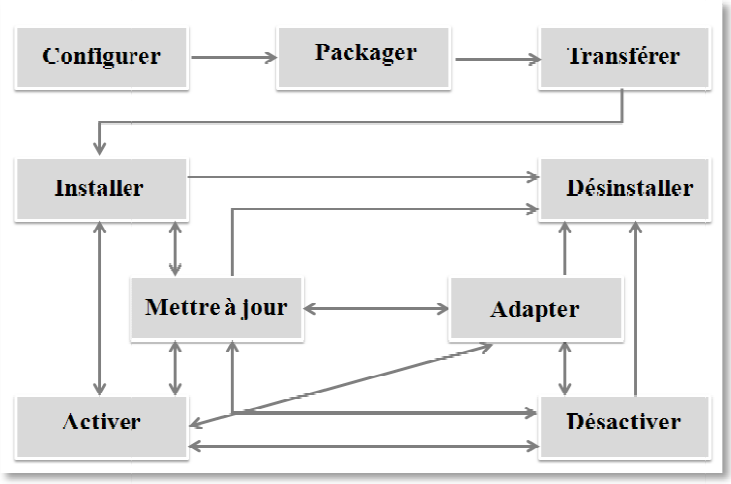


Figure 30 - Un processus générique de déploiement [Lestideau *et al.*, 2001].

Ce schéma comporte neuf étapes présentées ci-dessous :

- **Configuration**. Mise en œuvre après la compilation du code, cette étape comporte la définition des paramètres de l'application à déployer, et la description de ses ressources (composants, images, bases de données, etc.). Sur les plates-formes mobiles, la configuration est souvent décrite à l'aide de documents textes ou XML (*eXtended Markup Language*). Par exemple, pour une application mobile de prise de photo développée en J2ME MIDP, le fichier de configuration doit contenir : les noms des dossiers de l'ordinateur dans lesquels les classes de l'application se trouvent, le nom de la classe principale (le nom du MIDlet), les versions des icônes à utiliser, le nom du dossier sur le téléphone mobile où les photos seront stockées, les paramètres par défaut pour l'usage de l'appareil photo, etc. Le système iPhone OS (version 3.0) permet de décrire les fonctionnalités minimales du terminal mobile requises pour l'exécution d'une application. Par exemple, l'exigence d'une fonction d'envoi de SMS, la présence d'un GPS ou encore la nécessité d'une connexion Wi-Fi. Ces métadonnées seront exploitées lors du transfert afin d'éviter l'installation d'une application sur un dispositif Apple dont une de ces fonctionnalités est absente. Des outils de développement rapide d'applications mobiles (NetBeans, Eclipse, Microsoft Studio, Xcode, etc.) offrent des formulaires sous forme de « Wizards » afin d'aider les développeurs dans la mise en œuvre de ce processus de configuration. La description de la configuration créée est utilisée à la fois pour la génération de l'application finale, et pour la définition du comportement initial de l'application lors de son exécution.
- **Packaging**. Afin de faciliter la distribution et l'installation d'un logiciel, les développeurs génèrent un seul *package* contenant toutes les ressources nécessaires pour l'installation d'une application [Lestideau *et al.*, 2001]. Les documents issus de la phase de configuration sont repris afin d'accomplir cette tâche. Généralement, les *packages* d'installation possèdent des métadonnées pour indiquer leur contenu (tels que l'arborescence des dossiers, les noms de fichiers). Le type et le format de ces fichiers

varient selon les plates-formes mobiles. Les fichiers *.cab* sur la plate-forme Windows Mobile, les fichiers *.jar* sur J2ME MIDP, et les fichiers *.apk* de Android, sont des exemples de « packages » générés après cette phase de déploiement. Les kits de développement de plates-formes mobiles offrent des outils qui assistent l'activité de packaging (par exemple, le Wirelless Toolkit fourni par la plate-forme J2ME MIDP). Une autre particularité des plates-formes mobiles est la possibilité de certifier les applications lors du packaging. La certification du package d'installation atteste de l'authenticité de l'application et de l'absence d'un code malicieux (virus, trojan, etc.).

- **Transfert.** Cette étape du processus de déploiement fait référence à la distribution du fichier d'installation généré à la phase de packaging. Actuellement, les systèmes d'exploitation de dispositifs mobiles proposent plusieurs manières de transférer les packages d'installation. Ce transfert est mis en œuvre principalement par les utilisateurs eux-mêmes. Ils téléchargent, à partir de leurs ordinateurs de bureau, les applications disponibles sur les sites des fournisseurs. Ensuite, les fichiers d'installation sont transférés aux dispositifs mobiles à l'aide de logiciels de synchronisation (par câble USB ou par IrDA). Les technologies de communication dispositif-dispositif (tels que Bluetooth, IrDA, MMS[48]) offrent également l'échange direct d'applications entre les utilisateurs sans avoir besoin d'un ordinateur de bureau servant d'intermédiaire. Avec l'accès au Web via les dispositifs mobiles, un nouveau modèle de transfert est apparu pour les applications mobiles. Les utilisateurs peuvent désormais accéder à des portails d'applications tierces, choisir des applications, et, ensuite, les transférer directement sur leurs dispositifs mobiles. Ce modèle de déploiement supporté par la majorité des plates-formes mobiles est nommé transfert OTA (*Over - The - Air*) [Harmer, 2003]. Le processus de déploiement démarre généralement après une sélection de l'application à télécharger sur le navigateur Wap ou XHTML du dispositif mobile [Fjellheim, 2006]. Récemment, le fabricant Apple a révolutionné le modèle de transfert OTA en mettant à la disposition l'application App Store[49] sur l'iPhone OS. Contrairement au modèle OTA, les fournisseurs de services pour l'iPhone OS ne sont plus obligés d'attirer les utilisateurs sur leurs sites Web. Les entreprises qui fournissent des widgets mobiles ou des contenus multimédias (films, musiques, etc.) peuvent les rendre disponibles via le service Itunes Store[50]. Des usagers de l'iPhone et de l'iPod Touch utilisent l'application App Store pour lancer des recherches par mot-clé ou parcourir les catalogues d'applications de l'Itunes Store. Ils peuvent ainsi acheter les applications tierces et les télécharger directement à partir de leurs téléphones avec le logiciel App Store. Un filtrage des applications disponibles pour le modèle du dispositif de l'utilisateur est également mis en place en utilisant les métadonnées sur les fonctionnalités minimales requises pour l'application (e.g., exigence d'un GPS). L'objectif est d'éviter le transfert d'une application qui ne peut pas s'exécuter correctement sur le dispositif de l'utilisateur. Ce service dispose actuellement de plus de 35000 applications et a déjà dépassé la barrière du milliard de téléchargements[51]. Du point de vue des fournisseurs, ce modèle de transfert d'applications mobiles a certains inconvénients. Le déploiement est contrôlé entièrement par Apple qui, en plus de s'octroyer 30% des revenus de ventes, peut décider à tout moment de ne plus rendre

[48] Multimedia Message Service – Protocol d'envoi de messages multimédia qui peuvent contenir des textes, des images, et des vidéos.

[49] Application App Store de l'iPhone OS. Disponible sur: http://www.apple.com/fr/iphone/iphone-3g-s/app-store.html

[50] Itunes Store. Système de vente d'applications et contenu multimédia proposé par Apple. Disponible sur : http://www.apple.com/fr/itunes/

[51] http://www.itespresso.fr/app-store-dapple-un-milliard-de-telechargements-enregistres-24894.html

disponible l'application du fournisseur [Zittrain *et al.*, 2009]. Cependant, du point de vue des utilisateurs, cette approche offre la convivialité de trouver toutes les applications tierces à un même endroit et la garantie de trouver des applications de qualité certifiées par Apple. Ce modèle de téléchargement commence à être repris par d'autres constructeurs de dispositifs (tel que Nokia et HTC avec Android) qui ont lancé aussi leurs portails d'application en 2009. Ces constructeurs sont plus ouverts qu'Apple tandis que leurs dispositifs supportent également d'autres types de transfert. La Figure 31 présente un aperçu de l'application App Store sur l'iPhone 3G et de son homologue chez Nokia : le logiciel Ovi Store (exécuté sur le dispositif Nokia N97).

- **Installation**. Le processus d'installation peut commencer une fois que les packages sont transférés vers le dispositif cible d'exécution. Ce processus consiste à décompresser les fichiers d'installation et à vérifier si toutes les dépendances d'exécution de l'application sont satisfaites [Benamar *et al.*, 2008]. Ces dépendances expriment notamment les ressources nécessaires pour exécuter correctement l'application (telles que la version minimale du système d'exploitation, la présence d'une machine virtuelle ou d'une bibliothèque). Sur les plates-formes mobiles, ces dépendances sont fréquemment décrites avec des métadonnées incluses dans le fichier d'installation. Par exemple, sur la plate-forme J2ME MIDP, le fichier *.jar* du MIDletSuite possède un descripteur : le MANIFEST.MF. Ce fichier peut contenir les noms des bibliothèques optionnelles exigées pour l'exécution de l'application. En cas d'absence de ces bibliothèques, l'installation de l'application génère un message d'échec. Outre la vérification des ressources, le processus d'installation peut contrôler les certificats d'authentification des applications mobiles. Le logiciel chargé du déploiement lit ces certificats et accorde un droit d'accès aux applications en fonction des informations décrites dans les certificats. Le système d'exploitation pourra avertit l'utilisateur du danger d'installer une application, si son certificat n'est pas présent ou s'il contient une anomalie.

Figure 31 - Exemples de portails d'applications mobiles : l'Apple App Store et le Nokia Ovi Store.

- **Activation et Désactivation**. Après une installation correcte, l'application peut enfin être activée. Pour des systèmes simples, l'activation inclut uniquement l'exécution d'une commande afin de démarrer l'application (telle que le clic sur l'icône de l'application). Elle sera chargée en mémoire et pourra ainsi être utilisée par la suite. Pour des systèmes plus complexes, l'activation d'une application exige le démarrage d'autres ressources ou systèmes (des serveurs, des bases de données, etc.) avant d'activer l'application elle-même [Benamar *et al.*, 2008]. Le processus de désactivation consiste à arrêter l'exécution de l'application, et à libérer les ressources exploitées (la

mémoire, l'écran, etc.). Sur les plates-formes mobiles, les applications sont fréquemment activées et désactivées manuellement par les utilisateurs.

- **Mise à jour**. Le processus de déploiement englobe également les étapes de maintenance et de gestion de l'évolution d'une application [Ali *et al.*, 2006]. L'activité de mise à jour s'insère dans la perspective de gestion de l'évolution d'une application et de l'installation de ses nouvelles versions. La mise à jour comprend d'ailleurs la suppression de l'ancienne version de l'application, le transfert de la nouvelle version, et la mise en place de ressources. Parfois, le processus est partiel et seulement certaines parties de l'application sont remplacées. Sur les plates-formes mobiles, la mise à jour est déclenchée principalement par les utilisateurs lorsqu'ils essaient d'installer une application déjà existante sur le dispositif [Fjellheim, 2006]. Dans ce cas, le logiciel de déploiement du système d'exploitation averti l'utilisateur de la présence d'une application avec le même nom. Il pourra choisir de la remplacer tout en conservant les données et l'ancienne configuration (par exemple, les droits d'accès de l'application). Afin d'automatiser ce processus de mise à jour, les dernières versions de plates-formes mobiles offrent des API qui permettent aux applications de déclencher elles-mêmes leurs mises à jour. Ainsi, une application peut être programmée pour se connecter à un serveur lors de son activation, et vérifier si existe une version plus récente de l'application ou d'une de ses ressources (des composants, des images, etc.). Nous retrouvons ce mécanisme de mise à jour sur les plates-formes Symbian et J2ME MIDP 2.0. Le modèle de mise à jour proposé sur l'iPhone OS est encore plus puissant. L'utilisateur n'a pas besoin d'activer l'application pour être informé d'une nouvelle version. Les utilisateurs sont avertis par l'application App Store dès qu'une nouvelle mise à jour d'une de leurs applications est disponible sur le service Itunes Store. La mise à jour est déclenchée par un simple clic dans l'application App Store. Cependant, la mise à jour dans ce type de plates-formes nécessite de l'arrêt de l'application. Les plates-formes mobiles ne permettent pas de remplacer des parties d'une application. La mise à jour implique ainsi le remplacement complet de son code.
- **Adaptation ou reconfiguration**. Une modification du comportement de l'exécution d'un logiciel n'a pas forcément besoin d'une mise à jour pour être accomplie. L'activité d'adaptation (ou reconfiguration) consiste à mettre en place une modification en conservant le code disponible sur l'environnement d'exécution de l'application. Cette étape du processus de déploiement permet de changer les paramètres d'exécution et la configuration de l'application. La reconfiguration peut être déclenchée par un agent externe (un logiciel, l'utilisateur, l'administrateur), par un événement (tel que la panne d'une ressource) et par les applications elles-mêmes [Lestideau *et al.*, 2001]. Sur les plates-formes mobiles, il est rare de trouver des mécanismes qui permettent de déléguer la reconfiguration d'une application à un autre logiciel. Les applications mobiles doivent contenir des méthodes qui mettent en œuvre la modification de leur propre configuration. Seul l'iPhone OS possède une interface standard pour les changements de configurations (préférences) des applications : le mécanisme *Settings*. Un développeur peut définir sur un fichier *.pList* un ensemble de propriétés qui pourront être modifiées. Le changement de la configuration doit être effectué par l'utilisateur sur l'application *Settings* qui avertit l'application du changement de sa configuration. Concernant le changement de configuration du dispositif, la plate-forme Android offre un mécanisme de notification qui permet aux applications d'écouter des événements de changement de configuration du dispositif. Parmi ces événements, on trouve : le changement des polices, des paramètres de dates/régions, et des modifications physiques du terminaux (orientation de l'écran, l'activation d'un

clavier). Une application sur Android peut être développée pour réagir à ces changements de configuration du dispositif et modifier ainsi son mode d'exécution. Nous pouvons considérer le changement manuel des droits d'accès d'une application (tel que la découverte de la position du mobile) comme étant une forme de reconfiguration statique. Ce mécanisme est présent sur plusieurs plates-formes mobiles, et, à l'instar des mises à jour d'une application, un arrêt de l'application est nécessaire pour le changement de cette configuration.

- **Désinstallation**. Ultime étape du cycle de vie d'une application, la désinstallation d'une application comprend la suppression de son code et des ressources installées. Une mauvaise désinstallation peut compromettre l'exécution des autres applications d'un système, dans le cas d'une suppression erronée d'une ressource partagée (une bibliothèque, base de données, etc.). Ceci met en évidence l'importance de l'exécution correcte de cette activité. La désinstallation est initiée principalement par les utilisateurs sur les plates-formes mobiles. Sa mise en œuvre exige la désactivation préalable de l'application afin qu'elle soit accomplie.

4.4 Synthèse du déploiement d'applications mobiles

La Figure 32 présente une vue d'ensemble d'une partie des activités du processus de déploiement sur la plate-forme Windows Mobile. Nous constatons constater les nombreuses possibilités de transfert d'une application ainsi que les rôles centraux du logiciel d'installation du système d'exploitation (WceLoad) et du gestionnaire de configuration. Lors de l'installation d'une application sur Windows Mobile, un ensemble de dossiers et des variables d'exécution sont créés. Ces informations et les droits d'accès de l'application sont insérés dans le registre de configuration qui exerce un rôle similaire à celui du registre des versions Windows *desktop*. Les données de configuration sont enregistrées sous la forme de CSP (*Configuration Service Providers*). Lors de la désinstallation, ces informations sont effacées.

Les schémas de déploiement utilisés sur d'autres plates-formes mobiles ressemblent à celui de Windows Mobile illustré par la Figure 32. Nous observons néanmoins certaines différences, notamment sur les formats de fichiers d'installation et sur les formes de transfert d'application (voir Annexe 1). À l'exception des étapes de configuration et packaging, l'intervention des utilisateurs est considérable tout au long du cycle de vie d'une application. De plus, l'arrêt de l'application est exigée avant la mise en œuvre de certaines activités de déploiement (telles que la mise à jour, la reconfiguration, la désinstallation). La mise à jour d'une partie d'une application (telle qu'un composant) est inexistante également sur la majorité des plates-formes mobiles. Des plates-formes telles que Symbian, J2ME et Android offrent des mécanismes qui permettent à une application de démarrer d'autres applications. Néanmoins, il n'existe pas de mécanisme pour contrôler entièrement le cycle de vie de l'application démarrée. Seule la plate-forme Android offre la possibilité de déléguer l'activation et la désactivation à une autre application.

À l'instar du développement, le déploiement d'applications mobiles pâtit de l'hétérogénéité des dispositifs mise en évidence dans la section précédente. L'utilisateur qui désire télécharger une application est obligé de connaître les caractéristiques matérielles et logicielles de son dispositif. Il doit identifier les applications qui fonctionnent sur son terminal mobile, et, ensuite, choisir la version la plus adaptée à ces caractéristiques. Par exemple, pour télécharger le navigateur Opera Mini, un utilisateur doit préciser le modèle de son dispositif (Figure 33) puisqu'il existe plusieurs versions de l'application, chacune adaptée à un modèle de dispositif différent. Afin d'avoir accès à une version plus aboutie du navigateur mobile, l'utilisateur doit de plus savoir si son dispositif supporte ou non l'API JSR 75 (Java File API).

Il s'agit d'une information assez technique pour la majorité des utilisateurs. Ce problème de filtrage est partiellement allégé sur les plates-formes qui utilisent des portails d'application (tel que l'App Store). Ces applications reconnaissent le dispositif d'accès et filtrent les applications susceptibles de s'exécuter sur le dispositif de l'utilisateur. Cependant, les portails exigent que soient générées plusieurs versions d'une même application, chacune dirigée vers une configuration de dispositif.

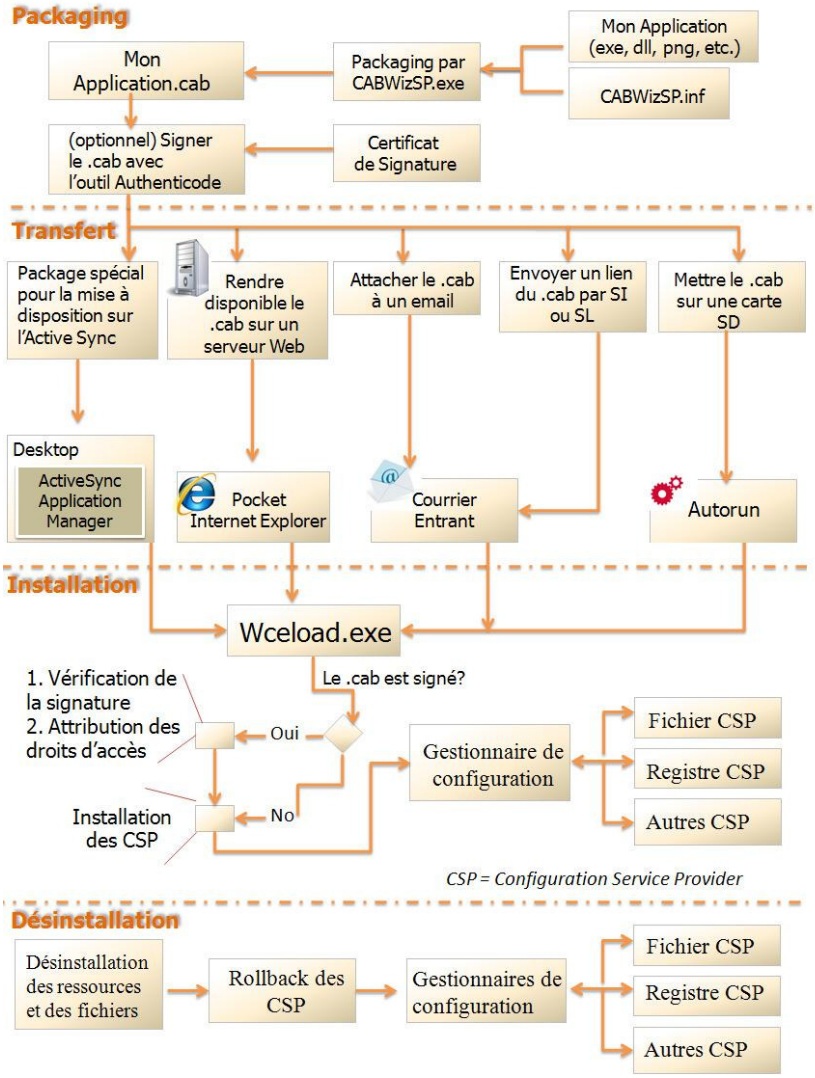


Figure 32 - Les différentes phases de déploiement d'une application mobile sur la plate-forme Windows Mobile. Source portail MSDN[52].

Dans le prochain chapitre, nous présentons des projets de recherche qui proposent des améliorations pour le déploiement d'applications mobiles afin d'éviter les inconvénients présentés ci-dessus. Leur objectif est d'alléger l'implication des utilisateurs en automatisant au maximum les étapes du déploiement, et de diminuer le coût engendré par la génération de différentes versions d'une même application.

[52] http://msdn.microsoft.com/en-us/library/aa458930.aspx

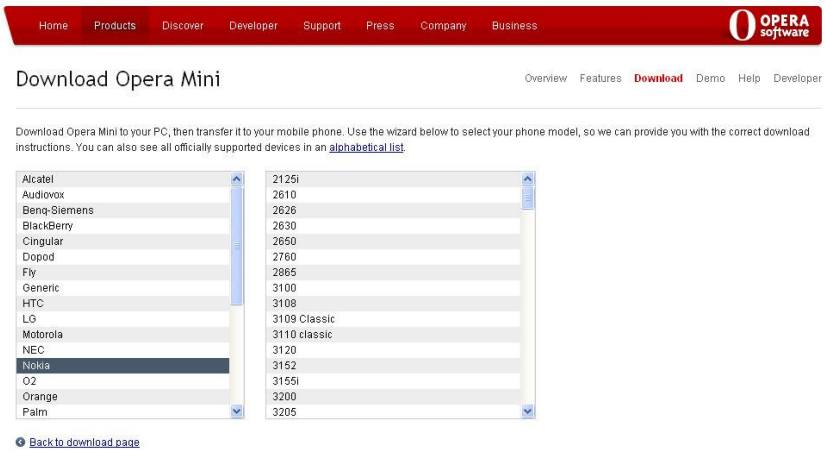


Figure 33 - Site de téléchargement de l'application Opera Mini[53]. L'utilisateur doit indiquer le modèle de son dispositif afin de télécharger la bonne version du navigateur mobile.

[53] http://www.opera.com/mini/download/nokia/nokia_2865/

5 SENSIBILITE AU CONTEXTE ET ADAPTATION D'APPLICATIONS MOBILES

Les dispositifs mobiles, malgré des limitations en termes de ressources (capacité de calcul et de mémoire, durée de vie de la batterie, etc.), ont l'avantage d'intégrer sans cesse de capteurs et de sources de données sur l'usager (agenda, contacts, …). Une application mobile peut combiner les informations produites par ces capteurs et par ces sources de données afin d'établir son contexte de l'utilisation (la localisation, l'activité de l'utilisateur, …). En s'appuyant sur la description de son *contexte d'utilisation*, l'application pourra adapter son propre comportement [Baldauf *et al.,* 2007], ou utiliser ces informations pour enrichir les métadonnées de documents multimédias produits à ce moment précis.

La sensibilité au contexte d'utilisation et l'adaptation d'une application mobile sont les sujets principaux de ce chapitre.

Nous définissons tout d'abord la notion d'adaptation et les nomenclatures utilisées pour la décrire. Ensuite, nous présentons différentes notions de contexte existantes et les principales caractéristiques d'un système sensible au contexte. Notre objectif est d'appliquer des solutions architecturelles similaires pour la gestion de documents multimédias.

Dans ce chapitre, nous décrivons particulièrement les architectures dédiées aux applications sensibles au contexte. Ces architectures de gestion de contexte offrent des modules, plus génériques, d'acquisition de contexte qui assurent généralement les processus de capture, d'agrégation, d'interprétation et d'inférence de l'information contextuelle.

L'impact de l'hétérogénéité matérielle et logicielle des dispositifs mobiles sur le coût de développement et du déploiement est abordé. Nous étudions un ensemble de solutions proposées pour réduire ce côut à l'aide de mécanismes d'adaptation des applications mobiles. Nous survolons les principaux moyens de description des caractéristiques matérielles et logicielles des dispositifs mobiles exploités par ces solutions d'adaptation. L'étude englobe à la fois des langages de description génériques tels que les standards CC/PP [W3C, 2007] et UAPROF [OMA *et al.*, 2001], et des langages spécifiques à une plate-forme de développement tels que WURFL[54] et Marjory[55].

[54] http://wurfl.sourceforge.net/

[55] http://www.j2mepolish.org/

La dernière partie du chapitre est consacrée à la description de plusieurs techniques et architectures d'adaptation d'applications mobiles. Nous nous intéressons en premier lieu aux approches de description des caractéristiques matérielles et logicielles dans un terminal mobile, qui sont essentielles pour l'achèvement des processus d'adaptation. Nous présentons ensuite des propositions d'adaptation statiques qui reposent sur des processus de génération de versions d'une application encore en phase de développement. Nous terminons le chapitre par une étude sur les approches d'adaptation dynamique et de déploiement adaptatif d'applications mobiles.

5.1 Adaptation d'applications mobiles

Les ressources limitées (batterie, mémoire vivre, vitesse du réseau, etc.) et la variabilité des dispositifs ont impulsé la proposition de mécanismes d'adaptation dans le domaine de l'informatique mobile. Ces mécanismes sont très présents dans les Systèmes d'Information en général. Initialement, la cible du processus d'adaptation de ces systèmes était les besoins de l'utilisateur. Le terme adaptation était ainsi employé pour designer l'adéquation d'une composante applicative, ou des données d'un système aux préférences et aux connaissances des utilisateurs [Villanova-Oliver, 2002]. Les processus d'adaptation exploités étaient principalement *la customisation* de l'interface réalisée par l'utilisateur (par exemple, l'ajout d'une barre d'outillage sur l'interface de Microsoft Word) ainsi que *la personnalisation* du contenu et des fonctionnalités initiée par le système lui-même [Moisuc, 2007].

La possibilité d'accès mobile aux Systèmes d'Information a augmenté le champ d'action de l'adaptation car les caractéristiques du dispositif d'accès (telles que la taille d'affichage) et de son environnement d'exécution (la mémoire disponible, la bande passante du canal de communication) ont désormais un impact sur la qualité du service reçu/aperçu par l'utilisateur [Zheng *et al.*, 2006]. Les proxys d'adaptation présentés dans le précédent chapitre sont des exemples de propositions qui prennent en compte les propriétés des terminaux mobiles et l'état du réseau lors du processus d'adaptation des pages et du contenu Web. Le processus d'adaptation peut tenir compte d'un ensemble d'informations encore plus large qui caractérise la situation de l'utilisateur lors de l'accès au système. Nous nommons cet ensemble *le contexte d'utilisation* dans les travaux de notre équipe [Moisuc, 2007][Kirsch, 2006][Carrillo Ramos, 2007]. Le *contexte d'utilisation* varie d'un ensemble simple et réduit d'informations (par exemple; le modèle du dispositif de l'utilisateur et sa position géographique) à une composition d'éléments plus riche sémantiquement et plus complexe à acquérir/inférer (telle que la relation d'amitié qui lie l'utilisateur du système et une personne à proximité). La notion de contexte et les architectures de contexte sont étudiées dans la prochaine section.

D'autres aspects sont importants dans le processus d'adaptation au delà de sa cible. Nous pouvons énumérer : le sujet de l'adaptation (c'est-à-dire, *quoi* doit être adapté), le mécanisme utilisé (*comment* l'adaptation est-elle mise en œuvre), le moment de sa mise en œuvre (*quand* le mécanisme est-il démarré), le responsable du processus (*qui* décide l'adaptation), l'objectif de l'adaptation (*pourquoi* adapter), et, enfin, la localisation de son exécution (*où* le processus est-il exécuté).

Dans le domaine de l'adaptation des Systèmes d'Information, les sujets à adapter les plus fréquemment mentionnés par les auteurs sont le contenu, la présentation et les fonctionnalités (les services) fournis par une application. Le contenu de l'application représente les informations et les fichiers multimédias (images, vidéo, etc.) présentés aux utilisateurs. La réduction, le filtrage et la transformation de contenu sont les mécanismes d'adaptation les plus considérés par les auteurs. Concernant la présentation, le processus d'adaptation permet la modification de la manière dont un contenu est perçu par les utilisateurs. La réduction de la qualité d'un vidéo pour un affichage sur un dispositif mobile possédant une mémoire vivre

limitée est un exemple de ce type d'adaptation. L'adaptation de l'interface graphique d'une application est un autre exemple d'adaptation de la présentation. D'ailleurs, le terme interface homme-machine plastique est utilisé pour désigner ces interfaces graphiques capables de s'adapter à la diversité de contextes d'utilisation tout en préservant leur utilisabilité [Thevenin, 2001].

Les applications mobiles multimédias sont caractérisées par l'utilisation des fonctions bas niveau des dispositifs, et l'accès à des bibliothèques de plates-formes mobiles qui ne sont pas toujours disponibles sur la diversité de modèles de terminaux mobiles existants. En conséquence, parmi les sujets à adapter mentionnés précédemment, nous nous intéressons en particulier à l'adaptation des fonctionnalités d'une application mobile. L'hétérogénéité des dispositifs empêche fréquemment que toutes les fonctionnalités fournies par une application soient exploitables. Le processus d'adaptation comprend ainsi l'ajout, la suppression et la modification des fonctionnalités (ou des services) fournies par l'application en accord avec les capacités du dispositif mobile et son environnement d'exécution.

Le processus d'adaptation d'une application mobile peut être réalisé en diverses phases de son cycle de vie. Une distinction importante est faite entre l'adaptation *statique* et *dynamique*. Le premier type d'adaptation suggère que le processus de création de versions d'une application ou de leurs ressources (images, données, etc.) a été entamé avant l'interaction de l'application avec l'utilisateur. Lors de l'installation de l'application, le processus d'adaptation final est à peine un choix entre les possibles versions des fonctionnalités ou des contenus qui resteront les mêmes tout au long de l'exécution de l'application.

L'adaptation dynamique, en revanche, se manifeste pendant la mise en œuvre de l'application. Les transformations sur les éléments de l'application à adapter (tels que ses services) sont effectuées à la volée, en accord avec l'environnement d'exécution de l'application. Le comportement de l'application mobile varie, ainsi, pendant son exécution. L'adaptation dynamique exige en contre partie une inspection constante des propriétés dynamiques d'un dispositif mobile (mémoire, réseau, position géographique, etc.).

Nous pouvons distinguer également l'entité responsable du démarrage et du contrôle du processus d'adaptation. Lorsque l'utilisateur lui-même intervient pour changer l'application, nous pouvons employer le terme *adaptabilité*. Dans ce cas, l'application devient *adaptable* et est susceptible d'être « customisée » par l'utilisateur. Le terme *adaptativité* est utilisé lorsque l'application décide elle-même de s'adapter et le processus d'adaptation est achevé sans aucune intervention de l'utilisateur. Les applications *adaptatives* ou *auto-adaptables* sont ainsi les logiciels capables de mettre en œuvre ce dernier modèle d'adaptation. L'adaptation dynamique et l'adaptativité sont constamment utilisées comme des synonymes [Miller *et al.*, 2005], néanmoins nous préférons souligner cette différence, car il est possible qu'un changement dynamique exige l'intervention de l'utilisateur.

Au delà de l'utilisateur et de l'application elle-même, l'adaptation d'une application mobile peut être démarrée et contrôlée par un logiciel tiers. Ceci est une particularité des approches d'adaptation d'applications mobiles qui s'exécutent au-dessus de *middlewares.* Ceux-ci surveillent l'environnement d'exécution de l'application et décident de l'adapter automatiquement en fonction des changements de cet environnement (par exemple, un *middleware* peut activer une fonction de communication d'une application mobile lorsqu'un réseau WiFi est détecté à proximité).

5.2 La sensibilité au contexte

Nous étudions dans cette section la sensibilité au contexte, les notions de contexte existantes et les usages généraux de l'information contextuelle.

5.2.1 La notion de contexte

La sensibilité au contexte est la propriété attribuée aux systèmes qui orientent leur comportement selon un *contexte d'utilisation.* Le terme « sensible au contexte » est fréquemment associé à un processus d'adaptation du système qui change son comportement ou les données qu'il fournit, conformément à l'état de l'environnement physique, et aux besoins des utilisateurs. Généralement, les systèmes sensibles au contexte exploitent ces informations dans un objectif d'adaptation de l'application et d'amélioration de l'utilisation et des performances [Baldauf *et al*, 2007]. Cependant, comme nous avons vu dans le chapitre 3, les informations sur le contexte peuvent être utilisées à d'autres fins au delà de l'adaptation.

En fait, les informations qui décrivent le *contexte d'utilisation* sont utilisées principalement pour [Baldauf *et al*, 2007] [Kirsch *et al.,*2005]:

- Adapter le comportement, les interfaces et le contenu d'un système en fonction du contexte d'utilisation (comme afficher un plan du quartier où se trouve l'utilisateur en utilisant ses préférences de visualisation et en accord aux caractéristiques de son dispositif mobile);
- Suggérer le contenu qui soit le plus approprié possible au contexte d'utilisation. Par exemple, un système de guidage touristique qui notifie l'utilisateur chaque fois qu'il est proche d'un monument historique dont la description est présente sur le système ;
- Annoter des documents à l'aide d'informations contextuelles qui peuvent être exploitées pour la réalisation *a posteriori* de requêtes ou de mécanismes de partage de contenu;
- Proposer et exécuter des services spécialisés selon le contexte (par exemple, des services proposés à l'utilisateur selon sa localisation ou son dispositif) ;

En raison de cette diversité d'usages de l'information contextuelle, différents notions de contexte ont été proposées. Dans le chapitre 3, nous avons indiqué que la définition de Dey est la plus acceptée[56]. Cette définition reste très générale : elle permet l'inclusion des informations explicitement fournies par l'utilisateur à travers une interface, et des informations capturées automatiquement par le système. Néanmoins, cette généralité induit un autre problème : l'identification des éléments qui composent la notion contexte. Pour Chaari *et al.* [Chaari *et al.*, 2005], la définition donnée par Dey ne permet pas d'identifier les données appartenant au contexte, ni de les distinguer des données propres à l'application. De plus, d'après cette définition, seule les informations qui peuvent améliorer l'interaction utilisateur-application doivent appartenir au contexte. Malgré ces critiques cette définition est désormais la plus utilisée par les travaux liés à l'informatique sensible au contexte.

Les auteurs Schilit *et al.* [Schilit *et al.*, 1994] et Ryan *et al.* [Ryan *et al.,* 1997], au contraire de Dey, proposent une notion de contexte dont les éléments sont bien définis : la localisation de l'utilisateur, l'environnement physique, l'identité des personnes présentes à proximité, et le temps. Manuelle Kirsch Pinheiro [Kirsch *et al.,*2005] tente d'intégrer ces deux notions tout en gardant la généricité de la définition de Dey, et en proposant également un groupe initial de concepts qui peuvent être étendu et est divisé en cinq catégories : temps, espace, outils, communauté et processus. Cette notion de contexte est exploitée pour l'adaptation des systèmes d'information coopératifs.

---

[56] D'après Dey [Dey et *al*, 2001], *le contexte est construit à partir de tous les éléments d'information qui peuvent être utilisés pour caractériser la situation d'une entité (personne, place ou objet), considérée comme pertinente vis à vis de l'interaction entre un utilisateur et une application.*

Une autre manière de définir le contexte est l'établissement de dimensions contextuelles [Baldauf *et al.*, 2007], par exemple, la division de contexte en deux dimensions : physique et logique [Prekop *et al.*, 2003]. La dimension physique fait référence aux éléments mesurés directement par l'accès à des capteurs (GPS, capteur de température, etc.), tandis que la dimension logique fait référence à l'utilisateur, ses préférences, son activité, et son état émotionnel. Ces dernières informations sont plus difficiles à capturer automatiquement.

Les travaux abordant la notion de contexte partagent, selon Rey [Rey, 2006], trois principes :

- Il n'y a pas de contexte sans contexte. Autrement dit, la notion de contexte se définit en fonction d'une finalité ;
- La capture du contexte n'est pas une fin en soi, mais les données capturées doivent servir à un objectif (tels que l'adaptation, l'annotation, le partage);
- Le contexte est un espace d'informations infini et évolutif. Par conséquent, il n'existe pas *a priori* de « contexte absolu », mais un espace qui se construit au cours du temps.

Une notion de contexte doit être ainsi générique et permettre également son extension afin de répondre aux spécificités du domaine qui va exploiter les informations contextuelles.

5.2.2 Modèles de contexte

La modélisation et la représentation des informations contextuelles sont des aspects centraux dans la conception d'une application ou d'une architecture sensible au contexte. Un modèle de contexte définit plus formellement la notion de contexte adoptée par un système et sa représentation indique comment les informations contextuelles sont stockées en machine [Kirsch *et al.,* 2005]. Les mécanismes d'enrichissement et d'exploitation du contexte sont directement influencés par cette représentation, car la représentation détermine les mécanismes de raisonnement et de requêtes qui peuvent être appliquées sur les informations contextuelles stockées.

À l'instar de la notion de contexte, plusieurs modèles et représentations existent. En effet, la modélisation du contexte est centrée sur un domaine ou une thématique précise qui, la plupart du temps, est lié à l'adaptation d'une catégorie de Système d'Information. Parmi les approches de représentation, nous identifions comme les plus usuels [Kirsch, 2006][Baldauf *et al.*, 2007][Chaari *et al.*, 2005] :

- **L'utilisation de paires « attribut/valeur »**. Le contexte peut être représenté simplement par un tableau des propriétés dont chaque ligne indique une propriété et sa valeur respective (par exemple, « {User="Windson", Location="Bureau", Personnes Proches="Karol"} »). Ce type de représentation est moins complexe à codifier, cependant les mécanismes de raisonnement sont quasiment inexistants sur ces approches.
- **La représentation par objets**. Plusieurs approches ont adopté UML (*Unified Modeling Langage*) pour la conception de modèles de contexte orientés objets [Kirsch, *et al.*, 2006] [Henricksen *et al.*, 2004]. Ces approches implémentent le modèle à l'aide d'un langage de programmation ou d'une base de connaissances. La Figure 34 illustre le modèle proposé par Kirsch [Kirsch, *et al.*, 2006] qui est implémenté sur un système de Représentation de Connaissances par Objets[57]. Le raisonnement repose sur l'usage

[57] http://www.inrialpes.fr/romans/arom/

de mesures de similarité entre les objets pour établir l'équivalence entre deux contextes.

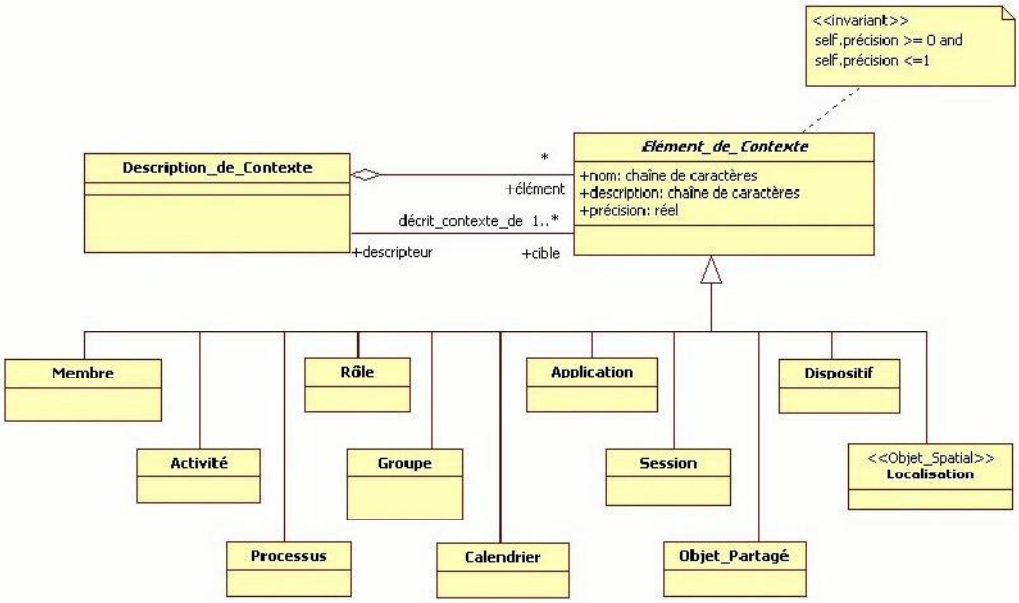


Figure 34 - Modèle objet proposé par Kirsch [Kirsch, 2006] pour la représentation du contexte de systèmes coopératifs.

- **L'utilisation du langage RDF**. À l'instar des vocabulaires de la représentation de métadonnées multimédias, le langage RDF du Web Sémantique est utilisé pour la représentation du contexte, principalement à l'aide des extensions du standard CC/PP qui sera présenté dans la section 5.3. Le RDF (Ressource Description Framework) permet la description de ressources Web en faisant des affirmations (« statements ») sur ces ressources. Chaque affirmation correspond à un triplet « ressource, propriété, valeur », et peut être représenté sous la forme d'un document XML (selon la spécification XML/RDF). Dans ces modèles de contexte, les ressources sont des éléments contextuels. L'inconvénient du RDF est sa limitation en termes de mécanismes de raisonnement associés. Cependant, son usage facilite l'échange d'informations contextuelles dans un système distribué (par exemple, un dispositif mobile qui capture le contexte et l'envoie à un serveur d'adaptation).
- **L'utilisation des ontologies OWL**. Les ontologies OWL sont la répresentation de contexte la plus utilisée ces dernières années comme en attestent les propositions COBRA [Chen *et al.*, 2004], SOCAM [Gu *et al.*, 2005], CoCA [Ejigu *et al.*, 2008], PUMAS[Carrillo Ramos, 2007], COMMANTO[Strimpakou *et al.*, 2006], et CoDaMOS [Preuveneers *et al.*, 2004]. Les ontologies OWL offrent une représentation plus formelle, et apportent aux systèmes sensibles au contexte divers mécanismes d'inférence (tels que les règles d'inférence) qui peuvent être mis en œuvre pour augmenter le niveau sémantique des informations contextuelles. La Figure 35 et la Figure 36 illustrent deux exemples d'ontologies pour la description du contexte d'utilisation. Ces deux ontologies définissent un premier niveau de concepts plus généraux, et, ensuite, ces concepts sont spécialisés pour la modélisation du contexte dans un domaine spécifique.

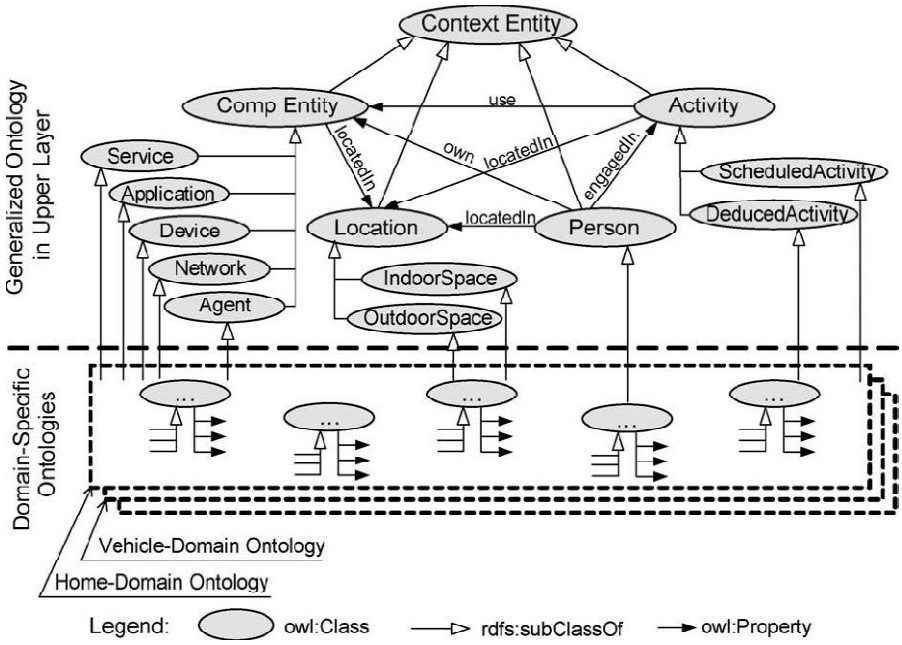


Figure 35 - Modèle de contexte de l'architecture SOCAM.

Ces modèles ne peuvent pas être directement réutilisés pour la représentation de contexte dans la gestion de documents multimédias, car la notion de contexte à laquelle ils sont liés est assez différente de celle dont nous avons besoin pour décrire le contexte de création d'un document multimédia. Cependant, ces ontologies indiquent comment l'information contextuelle peut être modélisée et segmentée. Nous pouvons également constater que la description des propriétés des dispositifs mobiles est très simple sur ces approches. Ce qui limite les mécanismes d'adaptation aux caractéristiques de ces dispositifs. Dans la section 5.3, nous présentons d'autres modèles et représentations du contexte axés davantage sur une description extensive des propriétés matérielles et logicielles d'un dispositif mobile.

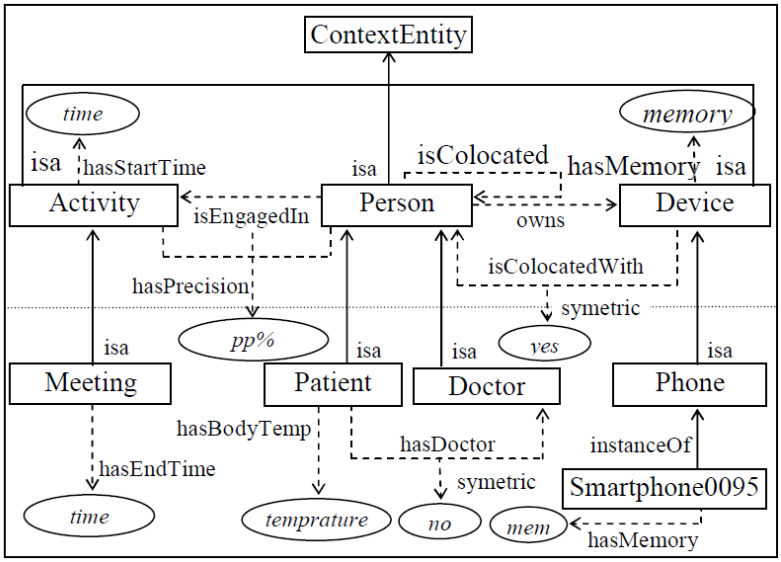


Figure 36 - Ontologie pour la représentation de contexte de l'architecture CoCA [Ejigu *et al.*, 2008].

5.2.3 Les architectures de gestion de contexte

Une application sensible au contexte inclut fréquemment les étapes suivantes : l'acquisition de contexte (tel que l'accès à un capteur GPS), l'enrichissement des informations

capturées (transformer les coordonnés GPS en un adresse), la détection de situations prédéfinies (l'utilisateur est chez lui), et l'exploitation des informations contextuelles (par exemple, pour l'annotation).

Les développeurs entremêlent fréquemment ces étapes au code métier afin de remplir le cahier de charges spécifique de l'application. Cette approche est utilisée quasiment par toutes les approches de gestion sensible au contexte de documents multimédias (Chapitre 3). Cependant, le choix limite la réutilisation de ces mécanismes et rend plus difficile l'extensibilité de l'application.

Afin d'éviter cet écueil, des architectures (canevas, intergiciels) d'applications sensibles au contexte ont émergé, telles que CoBra[Chen *et al.*, 2004], SOCAM [Gu *et al.*, 2005], CoCA [Ejigu *et al.*, 2008], PUMAS[Carrillo Ramos, 2007], CoDaMOS [Preuveneers *et al.*, 2004], COMMANTO [Strimpakou *et al.*, 2006] et le canevas proposé par [Henricksen *et al.*, 2004]. Ces architectures offrent des modules, plus génériques, d'acquisition de contexte qui assurent généralement les processus de capture, d'agrégation, d'interprétation et d'inférence de l'information contextuelle. La Figure 37 illustre le canevas d'acquisition de contexte proposé par Henricksen qui généralise les étapes d'une architecture sensible au contexte. L'acquisition des informations contextuelles est réalisée en boucle à partir des capteurs. Les informations initiales sont enrichies par des « interprétateurs » et stockées à l'aide d'un modèle de contexte pour, ensuite, être exploitées à des fins d'adaptation.

Du point de vue technologie, ces architectures de support d'applications sensibles au contexte peuvent être structurées de diverses manières. Parmi ces approches nous retenons :

- **Les approches par canevas (en anglais, *frameworks*)**. Ces solutions offrent un modèle d'organisation d'une application sensible au contexte divisé en modules qui sont chargés de mettre en œuvre une des activités de la gestion de contexte (telles que l'accès aux capteurs, l'agrégation d'information). Le Context Toolkit [Dey, 2001] et le canevas proposé par [Henricksen *et al*, 2004] sont des exemples de ces approches.

- **Les plates-formes multi-agents**. Ces approches partagent les activités de la gestion de contexte entre plusieurs agents qui communiquent entre eux en utilisant le standard proposé par la FIPA (*Foundation for Intelligent Physical Agents*). Par exemple, des agents vont être responsables de capturer le contexte, et d'autres de la décision d'adaptation. Les plates-formes PUMAS [Carrillo Ramos, 2007] et CoBra [Chen *et al.*, 2004] sont des exemples de ces approches. PUMAS a été proposé par un ancien membre de notre équipe de recherche. Cette plate-forme est composée de quatre Systèmes Multi-Agents (SMA), respectivement dédiés à la connexion aux Systèmes d'Information (SI), à la communication entre les utilisateurs et les SI, à la gestion de l'information, et à l'adaptation de celle-ci. L'adaptation est définie par un ensemble de règles d'inférences qui indiquent quand une méthode d'adaptation doit être déclenchée.

- **Les intergiciels de gestion de contexte**. Ces propositions fournissent une implémentation des principales fonctions d'une application sensible au contexte (capture, acquisition, raisonnement, méthodes adaptation, notification d'un nouveau contexte, etc.). Ces services sont intégrés au sein d'un intergiciel chargé de notifier et d'adapter les applications. CoDaMOS [Preuveneers *et al.*, 2004]., COMMANTO [Strimpakou *et al.*, 2006] et SOCAM [Gu *et al.*, 2005] sont des exemples de ces approches. Le système SOCAM propose un intergiciel installé sur un serveur qui reçoit des capteurs de l'environnement physique des informations sur le contexte d'un utilisateur mobile. L'intergiciel enrichit ces informations en

accédant aux registres des profils sur le serveur, et décide quand et comment les applications sur le dispositif mobile de l'utilisateur doivent être adaptées.

Bien que l'organisation de ces systèmes puisse inspirer la conception d'une architecture sensible au contexte pour la gestion de documents multimédias, ces canevas et intergiciels sont peu adaptés au développement d'applications sensibles au contexte pour les dispositifs mobiles en général. Ils s'avèrent forts consommateurs de ressources (mémoire et batterie) déjà limités sur les dispositifs mobiles [Preuveneers *et al.*, 2007]. En fait, le dispositif mobile dans ces approches n'est pas l'acteur principal de la capture et d'exécution des applications sensibles contexte. Le rôle du dispositif mobile est réduit à l'affichage des informations adaptées.

De plus, ces architectures s'appuient sur la méthodologie de conception du **tout-ou-rien.** Les canevas ou les intergiciels doivent être intègrés entièrement au code des applications. Ainsi, soit les concepteurs utilisent toutes les classes de canevas/intergiciel dans le code de l'application, soit il n'utilise pas l'architecture. Or, les plates-formes de programmation d'applications pour les dispositifs mobiles sont des environnements de programmation très hétérogènes pour lesquels le support des bibliothèques (classes, API, méthodes) diffère entre les dispositifs. Par exemple, sur la plate-forme J2ME MIDP 2.0, plusieurs API optionnelles (telles que l'API d'accès aux récepteurs GPS) ne sont disponibles que sur certains dispositifs mobiles. Si un intergiciel sensible au contexte contient du code qui accède au GPS à l'aide d'une API optionnelle, l'architecture entière ne pourra pas être déployée sur tous les dispositifs J2ME MIDP 2, même si l'application qui s'exécute sur l'intergiciel n'utilise pas le capteur de localisation.

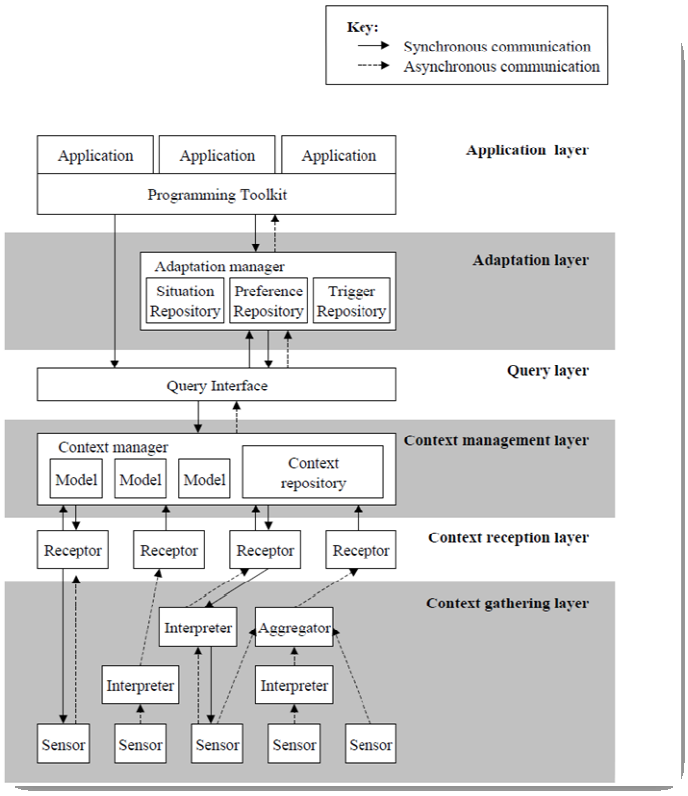


Figure 37 - Architecture générale d'un système sensible au contexte proposée par [Henricksen *et al.*, 2004].

Dans les prochaines sections, nous présentons des approches d'adaptation axées sur l'hétérogénéité des dispositifs mobiles. Ces approches peuvent être appliquées pour l'adaptation de l'architecture d'acquisition de contexte elle-même.

5.3 Profils de description de dispositifs mobiles

Selon le W3C Mobile Web Initiative[58], le succès d'exécution d'une application mobile exige une identification correcte des caractéristiques du dispositif sur lequel l'application est activée. La description des capacités d'un dispositif est ainsi fondamentale dans le guidage du processus d'adaptation du contenu, de la présentation et des services d'une application mobile [Kirsch *et al.*, 2005][Preuveneers *et al.*, 2004]. Par exemple, la taille d'affichage peut déterminer la taille d'une image à afficher, le support d'écran multitouche peut définir la présentation du menu, et l'absence d'un GPS peut empêcher l'accès à certaines fonctionnalités du système. Nous présentons dans cette section les principales technologies utilisées pour la représentation de profils de dispositifs mobiles, et des modèles pour la description du contexte d'utilisation de ces dispositifs. Nous nous intéressons particulièrement aux registres de profils de dispositifs, en vue d'une utilisation de ces registres dans un processus d'adaptation dynamique des fonctionnalités d'une application mobile.

5.3.1 Composite Capabilities/Preference Profiles (CC/PP)

CC/PP est une spécification du World Wide Web Consortium (W3C) [W3C, 2007] pour la description de caractéristiques d'un dispositif d'accès et de certaines préférences d'un utilisateur Web. L'objectif initial du CC/PP est d'offrir un support à l'adaptation de pages Web accessibles par divers navigateurs hétérogènes. Le navigateur Web envoie au détenteur d'une ressource (par exemple, une page Web) une description CC/PP de ses capacités et de préférences de l'utilisateur actuel. Le détenteur du contenu (par exemple, un serveur Web) peut ainsi déterminer la version d'une ressource qui doit être délivrée au client ou lancer un processus d'adaptation de la ressource existante.

La recommandation CC/PP définit une structure générale d'un profil de dispositif qui est divisée en trois branches : la plate-forme matérielle (*TerminalHardware*), la plate-forme logicielle (*TerminalSoftware*) et le navigateur Web (*TerminalBrowser*). Le standard CC/PP est lui-même basé sur un autre standard du W3C, le RDF. La Figure 38 présente un exemple de profil CC/PP fourni par la spécification. Les trois branches de CC/PP sont représentées comme des composants (*ccpp :component*) et leur capacités sont décrites comme des ressources RDF (*rdf :resource*), telles que la ressource *HardwarePlatform* en gras sur la figure. Une ressource possède un ensemble de propriétés (telles que la largeur d'écran - *prf :displaywidth* dans l'exemple) qui, pour chaque profil, contient une valeur (telle que 320 pour la largeur de l'écran du dispositif décrit dans la Figure 38).

Bien qu'une structure et un format de représentation de profils soient proposés par la recommandation, le standard CC/PP n'établit pas un unique vocabulaire pour la définition d'un profil de dispositif. Dans l'exemple de la Figure 38, toutes les ressources et leurs propriétés sont définies par un vocabulaire extérieur à la spécification (`xmlns:prf="http://example.com/Schema#)`. L'objectif est d'apporter de la flexibilité à la recommandation, et de permettre ainsi que chaque système puisse définir son propre vocabulaire.

[58] http://www.w3.org/Mobile/

Un exemple de vocabulaire est celui proposé par Lemlouma [Lemlouma, 2004] : l'UPS (*Universal Profiling Schema*). Ce vocabulaire établit des termes pour la description de propriétés matérielles (telle que la taille de mémoire) et de logiciels présents sur le dispositif mobile (tels que le système d'exploitation). UPS est utilisé pour la création d'un registre de profils de dispositifs qui est actionné lors d'une demande d'un document SMIL[59] par un terminal mobile. L'architecture proposée par cet auteur utilise la description de profils stockés sur le registre afin d'adapter le document demandé. Un autre exemple de vocabulaire est celui proposé par Indulska [Indulska *et al.*, 2003] qui propose d'étendre CC/PP pour la description du le contexte d'exécution d'une application mobile. Cette approche offre en plus la description de la localisation du dispositif et les caractéristiques dynamiques du réseau d'accès (la vitesse et le taux d'erreur).

L'inconvénient majeur de ces approches de vocabulaires CC/PP est le passage à l'échelle. Pour l'usage effectif d'un vocabulaire, un effort en amont de la description des profils des possibles dispositifs d'accès d'un système doit être réalisé. Un système doit capturer automatiquement les nouvelles informations du vocabulaire ou un administrateur du système doit décrire les profils des dispositifs mobiles comme sur le travail de Lemlouma. Ces approches sont envisageables dans le cas d'un système fermé, par exemple, celui d'une entreprise accédée par un nombre limité de modèles de dispositifs (fréquemment fourni par le propriétaire du système). Cependant, pour un système grand public, le défi posé par la création de profils est plus important, car le nombre de modèles croît constamment.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ccpp="http://www.w3.org/2002/11/08-ccpp-schema#"
         xmlns:prf="http://example.com/Schema#">
  <rdf:Description rdf:about="http://example.com/MyProfile">
    <ccpp:component>
      <rdf:Description rdf:about="http://example.com/TerminalHardware">
     <rdf:type rdf:resource="http://example.com/Schema#HardwarePlatform"/>
        <ccpp:defaults>
          <rdf:Description rdf:about="http://example.com/HWDefault">
     <rdf:type rdf:resource="http://example.com/Schema#HardwarePlatform"/>
            <prf:cpu>PPC</prf:cpu>
            <prf:displayWidth>320</prf:displayWidth>
            <prf:displayHeight>200</prf:displayHeight>
          </rdf:Description>
        </ccpp:defaults>
        <prf:displayHeight>640</prf:displayHeight>
        <prf:displayWidth>400</prf:displayWidth>
      </rdf:Description>
    </ccpp:component>
    <ccpp:component>
      <rdf:Description rdf:about="http://example.com/TerminalSoftware">
       <rdf:type rdf:resource="http://example.com/Schema#SoftwarePlatform" />
        <ccpp:defaults>
          <rdf:Description rdf:about="http://example.com/SWDefault">
            <rdf:type rdf:resource="http://example.com/Schema#SoftwarePlatform"/>
            <prf:name>EPOC</prf:name>
            <prf:vendor>Symbian</prf:vendor>
            <prf:version>2.0</prf:version>
          </rdf:Description>
        </ccpp:defaults>
      </rdf:Description>
    </ccpp:component>
    <ccpp:component>
  </rdf:Description>
</rdf:RDF>
```

Figure 38 - Exemple de Profil CC/PP (Source: W3C[60]).

[59] http://www.w3.org/AudioVideo/

[60] http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/#CCPPProfileStructOverview

5.3.2 User Agent Profile (UAProf)

UAProf est un vocabulaire CC/PP créé par le consortium Open Mobile Alliance (OMA) [OMA, 2001] en accord avec plusieurs fabricants de dispositifs mobiles (dont Nokia, Sony Ericsson, Siemens, Samsung). L'objectif d'OMA est d'offrir un vocabulaire commun de description des caractéristiques d'un modèle de dispositif mobile et, ainsi, de faciliter le processus d'adaptation de pages Web accédés par les utilisateurs nomades. Les fabricants de dispositifs, en plus d'utiliser UAProf, s'engagent à produire un profil pour chaque nouveau modèle de leurs terminaux mobiles. Contrairement à l'architecture proposée par la recommandation CC/PP, le profil UAProf n'est pas envoyé par le client mobile à chaque connexion. Le navigateur Web du dispositif envoie uniquement l'URI du profil à l'intérieur de l'entête de la connexion Wap ou Http. Ceci diminue la charge de communication entre le dispositif et le serveur Web. C'est le détenteur du contenu qui doit télécharger le profil afin de l'utiliser dans un processus d'adaptation.

Le vocabulaire proposé par OMA est destinée principalement à la description de l'écran du dispositif (notamment la propriété *prf : ScreenSize*), des propriétés du navigateur Web installé d'origine (le composant BrowserUA), et des capacités de reproduction de documents multimédias (par exemple, le support au format MPEG). Le profil UAProf contient également certaines propriétés pour la description des plates-formes mobiles, telles que la version du système d'exploitation et la version du profil de la machine virtuel Java. Cependant, son vocabulaire reste principalement tourné vers le support de l'adaptation d'applications Web.

L'adoption du vocabulaire UAProf par les constructeurs de dispositifs a apporté une solution concrète pour l'identification de ces terminaux lors d'accès Web. Toutefois, de nombreux inconvénients persistent dans l'utilisation à grande échelle de cette technologie :

- Les dispositifs mobiles ne possèdent pas tous des profils UAProf, principalement ceux dotés de Windows Mobile61;
- Les profils sont stockés par les fabricants, et, dans le cas d'une panne de leur serveur, l'identification d'un dispositif est compromise;
- Certaines imperfections sont détectables sur les profils UAProf et peuvent être fixées uniquement par les fabricants;
- Le vocabulaire UAProf est fermé et n'évolue pas pour combler les nouvelles caractéristiques des dispositifs (telles que la description de capteurs).

Le W3C Mobile Web Initiave (MWI) reconnait ces difficultés de la technologie UAProf et a proposé la création de registres plus fiables et flexibles : la recommandation Device Description Repository (DDR)62. Ces registres doivent contenir des profils de dispositifs de tous les fabricants confondus afin d'offrir un seul point d'accès à ces informations. Les profils peuvent être décrits en divers vocabulaires (UAProf, WURFL, etc.), néanmoins une interface commune d'interrogation est proposée par la recommandation. De plus, ces registres doivent être dupliqués afin de garantir une disponibilité maximale. Le groupe MWI doit assurer également la mise à jour et la qualité de ces bases de données. L'API d'accès aux registres DDR est désormais une spécification W3C depuis la fin de 2008. Cependant, aucun registre suivant cette spécification n'a été encore mis en place jusqu'à aujourd'hui…

[61] http://wurfl.sourceforge.net/uaprof.php

[62] http://www.w3.org/TR/DDR-Simple-API/

5.3.3 Wireless Universal Resource File - WURFL

WURFL est un projet Open Source pour la création et la maintenance d'un registre de profils de dispositifs mobiles. L'objectif du projet est d'aider le concepteur de pages WML dans la tâche consistant à rendre sa page adaptable à n'importe quel dispositif mobile. Le registre est structuré en XML et utilise un vocabulaire proposé et constamment mis à jour par la communauté WURFL. Ce vocabulaire englobe UAProf et des extensions pour la description d'autres propriétés du dispositif mobile, telles que la description du support à J2ME, Flash-Lite et WML. Le registre peut être téléchargé et exploité gratuitement dans une solution d'adaptation (commerciale ou gratuite). L'approche WURFL permet ainsi d'intégrer le registre de profils localement. Cette stratégie allège la charge de téléchargement de profils rencontrée sur les architectures purement UAProf. Des bibliothèques en JAVA et en PHP sont disponibles pour cette intégration. Un autre avantage de WURFL est de permettre la correction des profils, et l'ajout des nouvelles descriptions par tous membres de sa communauté.

L'identification d'un dispositif est mise en œuvre par l'API WURFL en utilisant plusieurs paramètres d'entrée tels que l'URI UAProf et la valeur de l'entête « User Agent ». Par conséquent, un terminal mobile qui ne dispose pas d'un profil UAProf peut être décrit sur WURFL et être identifié par la valeur de l'entête « User Agent ». Le vocabulaire WURFL a été conçu initialement pour l'adaptation de pages WML. Toutefois, son système de reconnaissance d'un dispositif, et la possibilité d'étendre son vocabulaire, ouvrent une voie pour la construction d'autres types de registre de profils. Un exemple est la création de registres pour le support d'adaptation d'applications mobiles développées à l'aide des plates-formes présentées dans le chapitre 4 [Parra *et al.*, 2008]. Le Tableau 4 illustre par exemple les propriétés de la plate-forme J2ME déjà rencontrées sur le vocabulaire WURFL.

Tableau 4 - Une partie du vocabulaire WURFL pour la description des propriétés de la machine virtuelle J2ME MIDP embarquée sur le dispositif mobile.

Propriété	Valeur possible	Description
j2me_mmapi_1_0	true/false	Support à MMAPI 1.0
j2me_wmapi_2_0	true/false	Support à WMAPI 2.0
j2me_btapi	true/false	Support à BlueTooth API
j2me_locapi	true/false	Support à Location API
j2me_nokia_ui	true/false	Support à Nokia UI
j2me_heap_size	Valeur entier	Limite de mémoire en temps d'exécution
j2me_screen_width	Valeur entier	Largeur de l'écran en pixels
j2me_photo_capture_enabled	true/false	Un Midlet a le droit de prendre des photos
j2me_capture_image_formats	String	Le type de format d'image disponible

5.3.4 Profils de description des plates-formes mobiles

Les propositions de description de modèles de dispositifs mobiles précédemment présentées sont principalement destinées à l'adaptation de pages Web (WML et XHTML). D'autres approches telles que Marjory[63] et les ontologies de description de dispositifs mobiles [Korpipää *et al.*; 2003][Wagelaar, 2005][Zhang *et al.*; 2008] ciblent l'adaptation d'applications mobiles qui s'exécutent localement sur les dispositifs mobiles.

[63] http://www.j2mepolish.org/cms/leftsection/introduction/marjory.html

Marjory est une base de données ouverte qui contient une description détaillée des caractéristiques matérielles et logicielles d'environ 900 modèles de terminaux mobiles disposant de J2ME MIDP. Le niveau de détail d'un profil Marjory permet d'avoir, par exemple, des informations sur la taille de l'écran, les bibliothèques J2ME MIDP supportées (telles que l'API d'accès aux récepteurs GPS) ou, encore, les propriétés d'accès à l'appareil photo du dispositif (telles que la résolution maximale permise). Marjory offre une application mobile J2ME MIDP qui, une fois installée sur un dispositif mobile, capture les informations sur le terminal et les envoie au registre de profils. Marjory offre également une interface Web de recherche manuelle de profils de dispositifs stockés dans son registre. Contrairement à WURFL, aucune API ou Web Service n'est prévue pour un accès au registre par une application tierce. Marjory est maintenue par l'entreprise J2MEPolish[64], dont la solution d'adaptation statique est présentée dans la prochaine section.

Les technologies de description de profils de dispositifs mobiles décrites jusqu'à ce point sont basées sur diverses représentations : modèle relationnel, XML et RDF. Cependant, ces technologies sont traitées par les propositions comme de simples représentations d'un ensemble d'attributs-valeurs (propriétés du dispositif et ses valeurs respectives). D'autres approches essayent d'apporter, au contraire, une forte représentation sémantique aux profils de dispositifs à l'aide des ontologies, principalement après l'arrivée des premiers travaux autour du Web sémantique [Bernes-Lee *et al.*, 2001]. Une ontologie de description de dispositifs mobiles est une représentation compréhensible par une machine qui contient la définition des concepts de base d'un dispositif (tels que les composants physiques et logiciels d'un terminal mobile) ainsi que des relations entre eux.

Hydra [Zhang *et al.*, 2008], CoDaMoS [Wagelaar, 2005] et l'ontologie proposée par Korpipää [Korpipää *et al.*, 2003] sont des exemples d'approches qui font l'éloge d'une représentation sémantique et formelle des capacités des dispositifs mobiles. Ces trois propositions sont assez similaires en termes de concepts et de relations représentés. Ils permettent également la représentation d'attributs statiques et dynamiques des dispositifs. La Figure 39 illustre une d'entre elles : l'ontologie CoDaMoS. Le schéma I de la figure présente les composants du dispositif (le concept *Platform*) : les capacités inhérentes au modèle du terminal (telles que le concept *Virtual Machine*), et les propriétés dynamiques qui changent lors d'une exécution (les concepts dérivés de *Resource*). Les relations entre les concepts de l'ontologie CoDaMoS sont aussi représentées telles que la relation *HasEnvironement* entre le dispositif et son environnement d'exécution.

L'ontologie de Korpipää, comme CoDaMoS, offre un vocabulaire pour décrire les propriétés basiques d'un modèle de dispositif. En revanche, l'ontologie de Korpipää permet en plus de représenter des états de configuration du dispositif, tels que « l'écran est allumé », « le téléphone est pris dans la main ».

Outre la création d'un vocabulaire commun et formel, l'intérêt d'utiliser des ontologies réside dans un possible usage de mécanismes de raisonnement (tels que la classification et les règles d'inférences). Ces mécanismes peuvent expliciter automatiquement des connaissances auparavant implicites dans les ontologies [Zhang *et al.*, 2008] [Strimpakou *et al.*, 2006]. Par exemple, le schéma II de l'ontologie CoDaMoS (Figure 39) montre une hiérarchie de machines virtuelles, et des relations entre ces machines et les bibliothèques de rendu graphique de Java (AWT et Swing). À partir de la découverte que la version de la machine virtuelle d'un dispositif mobile est Personal Java, le moteur d'inférence de CoDaMoS peut déduire que ce dispositif est capable d'afficher des composants AWT. Du même, il peut

[64] http://www.j2mepolish.org/

déduire qu'un dispositif ou réside la machine virtuelle J2ME MIDP 2 comporte l'exécution d'une application MIDP 1.

L'approche Hydra [Zhang *et al.*, 2008] utilise également des règles d'inférences pour dériver des informations porteuses d'un niveau sémantique plus élevé. Par exemple, les informations « niveau de batterie égale à 0.1 » et «communication WiFi activée » peuvent dériver de l'information « niveau de batterie vraiment très bas ». Cependant, « niveau de batterie égale à 0.1 » et «dispositif inactif » produit l'information « niveau de batterie bas ». Ces informations dérivées peuvent servir à mieux guider le processus d'adaptation d'une application.

Dans la suite de ce chapitre, nous concentrons notre attention sur les techniques d'adaptation d'applications mobiles qui exploitent les profils de dispositifs et les ontologies décrites antérieurement.

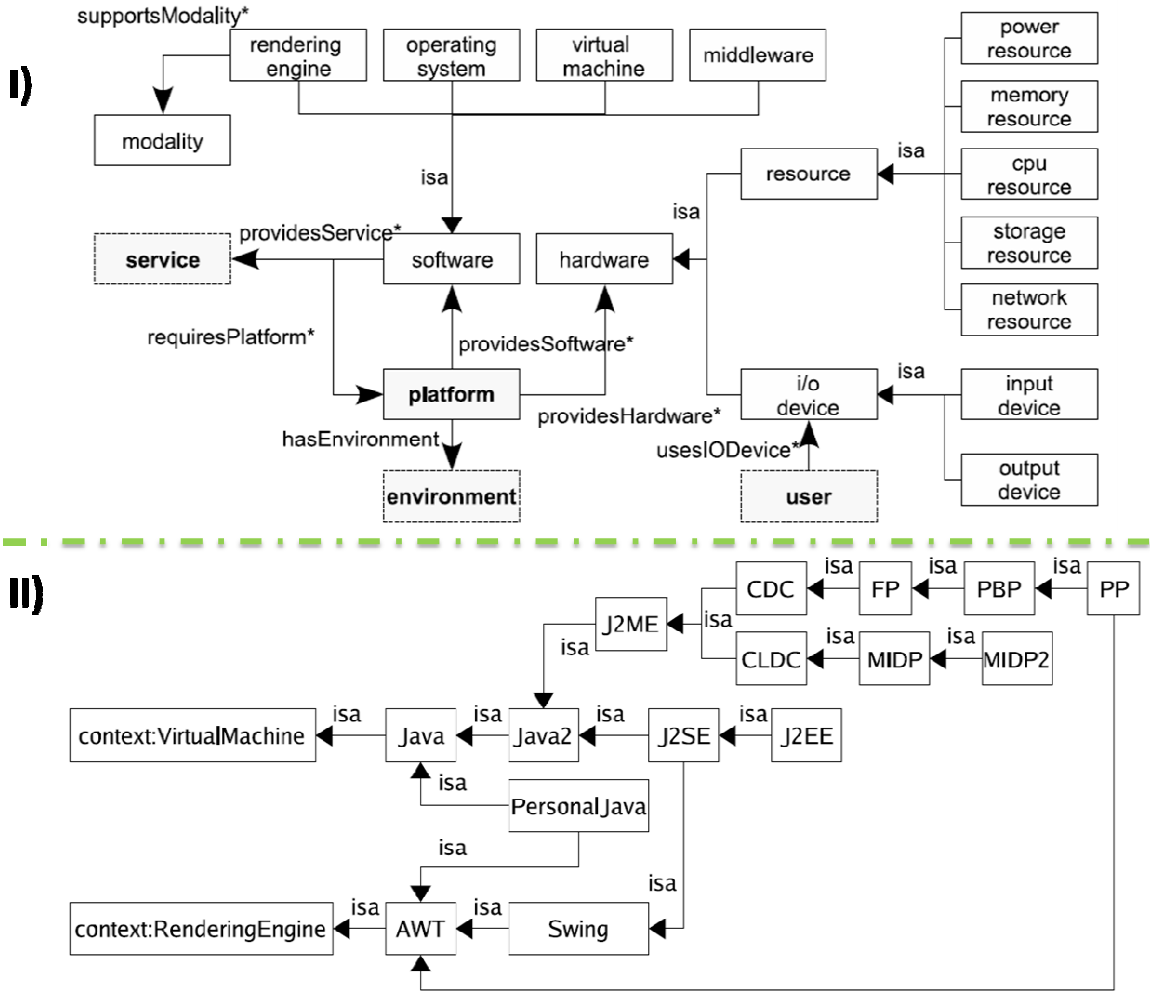


Figure 39 - Ontologie CoDaMoS pour la description de dispositifs mobiles et des profils/configurations de la plate-forme Java [Wagelaar, 2005].

5.4 Adaptation d'applications mobiles lors du développement

Nombreuses sont les approches d'adaptation d'applications mobiles qui ont émergées après l'avènement des systèmes d'exploitation mobiles ouverts. Nous nous intéressons

particulièrement dans cette section aux propositions d'adaptation des fonctionnalités (les services) d'une application qui s'appliquent avant la mise en œuvre de l'application. Ces propositions se caractérisent par la génération semi-automatique des versions d'une même application mobile, chacune adaptée à un modèle de dispositif choisi par ses développeurs.

D'abord, nous présentons les travaux qui exploitent des « directives » entremêlées au code de l'application (à l'exemple d'IFDEF du langage C). Ces approches se servent d'un pré-compilateur et d'une description du dispositif cible afin d'évaluer les directives. Elles génèrent ensuite une version de l'application compatible avec le modèle du dispositif choisi par le développeur.

Dans la seconde partie de la section, nous nous intéressons aux approches dirigées par modèles qui proposent des outils de génération de code et un ensemble de processus de transformation des modèles, afin de réduire l'effort de programmation de différentes versions d'une même application mobile.

5.4.1 Les approches basées sur des directives

Les approches J2ME Polish[65], Netbeans Mobility Pack[66], Celsius[67] proposent l'adaptation des applications mobiles aux dispositifs mobiles lors du développement, c'est-à-dire *avant* son exécution. Ces trois approches combinent un registre de profils de terminaux mobiles, un langage de directives et un préprocesseur afin de générer plusieurs versions du code d'une application en fonction d'un modèle de dispositif choisi par le concepteur.

J2ME Polish, par exemple, est une boîte à outils d'aide au développement d'applications J2ME MIDP et Doja I-Mode (un profil J2ME qui s'exécute sur la configuration CLDC). Le registre de profils Marjory présenté dans la section 5.3.4 est un des composants principaux de J2ME Polish. Cette boîte à outils payante offre un ensemble de bibliothèques qui améliore le visuel des applications Java Mobile. Une autre caractéristique de ces bibliothèques est la possibilité de décorer les interfaces graphiques à l'aide de feuilles de style CSS.

Le grand avantage de J2ME Polish est la génération semi-automatique de versions d'une même application proportionnée par cet outil. Lors du développement d'une application, les concepteurs peuvent spécifier des alternatives à l'intérieur du code d'une classe en utilisant un langage de directives similaire à l'IFDEF du langage C. Les concepteurs peuvent faire référence à toutes les propriétés de bas niveau d'un dispositif décrites dans un profil Marjory. La Figure 40 illustre un exemple d'utilisation de J2ME Polish. Le code de l'application de l'exemple contient une directive qui fait référence à la présence de la bibliothèque MMAPI (bibliothèque d'accès aux fonctions multimédias telle que le MMS).

Le développeur décrit qu'un morceau du code doit être intégré au code final de la classe seulement si la plate-forme du dispositif dispose de cette bibliothèque (la méthode *showSendPhotobyMMS()*). Avant la compilation de l'application, le concepteur informe le(s) modèle(s) cible(s) et les ressources utilisées (les images, les icônes, etc.) dans un document XML de configuration. Ensuite, le préprocesseur accède au registre Marjory afin d'exécuter les directives en fonction des propriétés du modèle du dispositif préalablement choisi par le concepteur. Le préprocesseur peut adapter également les ressources de l'application en fonction des propriétés du dispositif (par exemple, changer la taille ou le format d'une icône). Concernant le code final, le préprocesseur n'enlève pas le code qu'il ne doit pas exécuter. Le

[65] http://www.j2mepolish.org/

[66] http://www.netbeans.org/kb/trails/mobility.html

[67] http://www.mobile-distillery.com/celsius-2/overview-43.htm

préprocesseur transforme en réalité les lignes de code à ignorer en commentaires Java (//). La Figure 40 illustre les deux versions possibles du code de l'application.

Les outils Celsius et Netbeans Mobility Pack suivent quasiment la même approche d'adaptation que J2ME Polish. Elles se distinguent par le pouvoir d'expression de leurs langages de directives, par la qualité et la quantité de profils de DM de leurs registres, et, finalement, par le niveau d'automatisme de leurs outils d'aide à la génération des versions d'une application. Dans ces approches de directives, les concepteurs sont contraints d'intégrer le code d'adaptation dans le code métier de l'application. Ceci peut rendre plus difficile la réutilisation et la maintenance du code de l'application [Carton *et al.*, 2007]. Ces propositions apportent néanmoins une première solution à la fragmentation de bibliothèques J2ME MIDP et assistent la génération de versions distinctes.

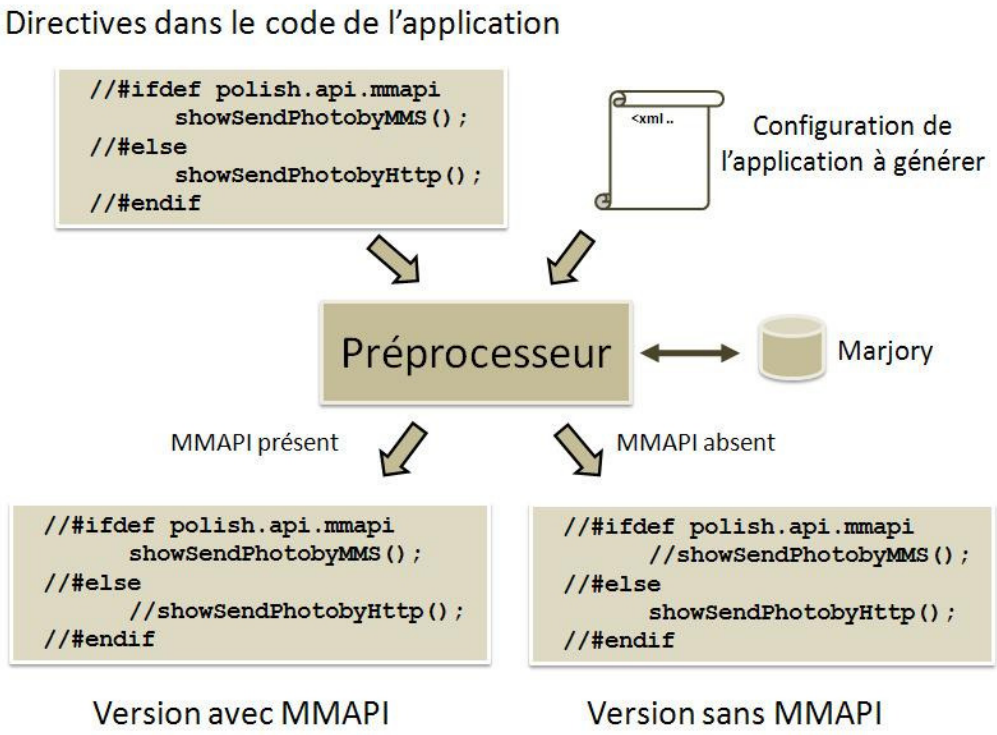


Figure 40 - Mécanisme de génération de versions de l'approche J2MEPolish.

5.4.2 Les approches dirigées par les modèles

Les approches basées sur des directives prennent en compte le processus d'adaptation dans les dernières phases du cycle de développement d'une application : *l'implémentation* et *le test*. D'autres approches prennent en considération l'adaptation dès les premières étapes du cycle de développement, par exemple, lors de phase *d'analyse* et *de modélisation*. C'est le cas des propositions présentées dans cette section qui utilisent les architectures dirigées par les modèles pour l'adaptation d'une application mobile.

L'architecture dirigée par les modèles (en anglais, MDA - *Model Driven Architecture*) est une démarche de développement d'applications, proposée par l'OMG (*Object Management Group*[68]). L'objectif de MDA est d'utiliser l'ingénierie dirigée par les modèles tout au long du cycle de développement d'une application. Au départ, un modèle métier indépendant d'une

[68] http://www.omg.org/

plate-forme spécifique (Platform Independent Model, PIM) est conçu. Ensuite, un processus de transformation génère un modèle dépendant d'une plate-forme en particulier (Platform Specific Model, PSM) qui peut être ajusté par les concepteurs. Des générateurs sont utilisés à la fin pour transformer le PSM dans le code final de l'application. Des techniques de modélisation et des techniques de transformation de modèles sont les principaux objets de recherche de ce domaine.

Les auteurs [Carton *et al.*, 2007] et [Wagelaar, 2005] proposent l'utilisation de MDA pour la génération d'applications mobiles adaptées à plusieurs plates-formes et dispositifs mobiles. Le principe de base de ces approches est de créer un modèle initial de l'application (PIM), indépendant d'un modèle de dispositif spécifique. Les concepts appartenant à ce modèle sont enrichies à l'aide des annotations et d'un langage par contraintes (tel que OCL). Une fois la plate-forme mobile cible choisie par les développeurs, des processus de transformation sont mis en place. Ces transformations seront guidées par les caractéristiques du dispositif (ou de la plate-forme mobile) cible et par les annotations décrites sur le modèles. Carton *et al.* [Carton *et al.*, 2007] offrent une extension d'UML, le Theme/UML afin de représenter ces informations qui guideront le processus de transformation.

Wagelaar [Wagelaar *et al.*, 2006] propose en plus l'utilisation d'un diagramme de variabilité (*features model*) et d'une base de connaissances (des ontologies et des règles) pour mieux choisir les possibles alternatives de transformation.

Des processus de transformation d'un modèle UML étendu vers les diverses versions de code doivent être codifiés a priori et peuvent être réutilisées dans la génération d'autres applications. Lorsque le concepteur choisit un dispositif pour générer l'application, il doit fournit une description de ce dispositif en utilisant l'ontologie CoDaMoS présentée dans la Figure 39. Le générateur de code peut ainsi déduire quelles sont les alternatives de transformations qui génèrent un code viable pour une exécution sur le dispositif cible. Par exemple, si le dispositif possède la machine virtuelle J2ME MIDP, le générateur de code produit une version LCDUI pour l'interface.

Malgré l'avancement de la communauté MDA, les processus de transformations de modèles ne sont pas encore assez matures, principalement dans la génération du code final. Cependant, la représentation de variabilités et le processus de raisonnement proposé par Wagelaar offrent une solution convenable de guidage d'un processus d'adaptation de fonctionnalités.

5.5 Adaptation dynamique et déploiement adaptatif d'applications mobiles

L'adaptation d'une application mobile ne se résume pas uniquement à la génération des versions lors de son développement. Des mécanismes d'adaptation peuvent être également mis en œuvre pour guider le déploiement d'une application mobile et pour changer le comportement de l'application lors de son exécution [Ayed *et al.,* 2008] [Zachariadis *et al.*, 2006] [Chan *et al.*, 2003] [Fjellheim, 2006].

Dans cette section, nous nous intéressons en particulier aux approches d'adaptation orientées composante. Afin de faciliter la compréhension de ces propositions, nous introduisons d'abord les fondements de la programmation par composants, et ses avantages pour l'adaptation d'une application mobile.

Nous présentons en détail ensuite les propositions de déploiement adaptatif qui prennent en compte les caractéristiques du dispositif mobile et du profil de l'utilisateur pour améliorer toutes les phases de déploiement d'une application (de l'installation à la mise à jour).

Contrairement aux approches de déploiement adaptatif qui ont été présentées précédemment, plusieurs travaux se consacrent essentiellement à la phase de reconfiguration

d'une application mobile [Zachariadis *et al.*, 2006][Chefrour, 2005][Chan *et al.*, 2003]. Une fois l'application installée, l'objectif de ces propositions est d'adapter l'application en fonction des changements des propriétés dynamiques du dispositif, telles que la mémoire vivre, la vitesse du flux du réseau, la batterie et la défaillance d'un capteur. Ces propositions sont présentées à la fin de cette section.

5.5.1 Programmation par composants et OSGi

Le modèle orienté objets (OO) est apparu comme la première méthodologie de développement appropriée pour la description de problèmes de domaines, et pour le développement plus organisé d'un système après la crise du logiciel des années 1970. L'encapsulation fournie par le modèle OO facilite certains changements dans un logiciel. Cependant, cette caractéristique, à elle seule, n'a pas été suffisante pour garantir le succès du modèle OO à s'adapter aux changements technologiques et pour garantir l'évolution facile d'un logiciel [Elfatatry, 2007]. En outre, les classes et les objets ont une granularité très fine, ce qui rend plus difficile la réutilisation du code.

La programmation par composants ou le développement orienté composants a été proposé avec l'objectif de garantir l'encapsulation avec une granularité plus grande. Ce nouveau paradigme de programmation fut inspiré du développement d'équipements électroniques construits à partir d'assemblage de composants électroniques de fonctions variables, néanmoins réutilisables en plusieurs configurations. La programmation par composants a pour caractéristique principale la séparation entre l'interface et l'implémentation des composants, qui sont l'unité de distribution et de déploiement des systèmes. Un composant logiciel[69] peut être défini comme une unité de composition qui contient des interfaces contractuellement spécifiées et des dépendances explicites envers un contexte d'exécution. Un composant peut être déployé indépendamment, être sujet à une tierce composition, et être utilisé, par divers systèmes (multi-use) [Szyperski, 2002] [Wang *et al.*, 2005]. Bien que l'objectif initial des composants fût de proposer l'encapsulation avec une granularité plus grande, les composants peuvent avoir des granularités très variées, allant d'un simple composant d'interface graphique à un système entier de réservation de billets d'avion.

La programmation par composants est une méthodologie importante de développement utilisée dans la conception de systèmes complexes (par exemple, des applications critiques, de grande taille et distribuées). L'utilisation de composants établit des frontières bien définies dans le code d'un système. Ces frontières simplifient la maintenance et l'extension d'un système. Une application développée à l'aide de ce paradigme de programmation devient un assemblage de composants préfabriqués. On voit apparaître ainsi deux types de programmeurs : les développeurs de composants (développement pour la réutilisation) et les assembleurs des composants (développement avec la réutilisation). Un composant peut être même fourni ou vendu par une autre entreprise.

Outre la réutilisation des « briques » préfabriquées, une caractéristique particulière du développement orienté composant est la *remplaçabilité*. L'interface d'un composant cache à ses clients (d'autres composants de l'application) les détails de sa mise en œuvre étant donné que toute communication avec le composant est faite par le biais de son interface. Un développeur peut ainsi substituer plus facilement un composant par un autre remplissant le même contrat (c'est-à-dire, possédant la même interface). La remplaçabilité des composants joue un rôle important dans l'évolution d'un système. Par exemple, un bug d'un composant

[69] La notion de composant est très variée et dépend du domaine d'application. Dans cette thèse, nous adoption la définition de Szyperski [Szyperski, 2002] qui était étendue par Wang *et al.* [Wang *et al.*, 2005]. La définition est exposée sur le texte.

peut être solutionné uniquement avec le remplacement du composant défectueux par un autre amélioré et respectant la même interface.

Les composants logiciels sont associés à leurs infrastructures (parfois, nommées containeurs ou *frameworks*). Différentes technologies de composants possèdent des infrastructures distinctes et acceptent ainsi différents formats ou modèles de composants. Une *infrastructure à composants* définit la structure de base au dessus de laquelle les composant s'exécutent ; un *modèle de composant* spécifie la description d'un composant valide et passible d'être branché à cette infrastructure, et un *mode de connexion* d'un composant identifie les possibilités d'assemblage et de communication entre les composants [Wang *et al.*, 2005].

Plusieurs infrastructures à composants existent aujourd'hui dont les plus remarquables sont CORBA, OpenCCM, Enterprise Java Beans, .NET et OSGi. Ces infrastructures sont utilisées en divers domaines tels que les applications Web, les systèmes répartis, et les applications en temps réel. Elles offrent parfois aux composants et aux applications un ensemble de services non-fonctionnels (tel que la sécurité, le log, la gestion de transactions, la persistance, etc.). Ces infrastructures garantissent également la gestion du cycle de vie des composants (création, destruction, activation, désactivation, etc.), et facilitent l'installation des composants de manière incrémentale et à distance.

Une autre capacité de certaines infrastructures à composants est le support à la reconfiguration dynamique qui correspond à la modification de la configuration d'un ou plusieurs composants lors de leurs exécutions sans exiger en contre partie l'arrêt ou le redémarrage du système entier [Rutheford *et al.*, 2002].

Bien que les infrastructures à composants n'aient pas été créées pour une exécution sur des plates-formes limitées en ressources, la programmation à composants est une des approches les plus utilisées pour l'adaptation du comportement d'une application mobile. Nous présentons dans la suite les avantages de l'utilisation de la programmation par composants pour l'adaptation d'une application mobile. Nous décrivons également OSGi, un des modèles de composants les plus diffusés aujourd'hui pour la création d'applications distribuées, et qui possède des implémentations pour certaines plates-formes mobiles.

5.5.1.1 Programmation par composants pour le développement d'applications mobiles

La représentation des applications mobiles sous forme de composants interopérables, outre la possibilité de réutilisation et la réduction de complexité qu'elle offre, facilite la mise en place d'un processus d'adaptation du comportement d'exécution de ces applications mobiles [Zachariadis *et al.*, 2006]. Quatre caractéristiques des modèles de composants sont retenues par les approches d'adaptation : le paramétrage, la remplaçabilité, la reconfiguration dynamique, et le contrôle du cycle de vie des composants. Ces quatre caractéristiques offrent la possibilité d'adapter la configuration d'une application avant son installation, de remplacer un composant lors de son exécution, de changer le comportement du système en fonction de ressources dynamiques du dispositif, et de délocaliser des composants afin d'améliorer la qualité de service fournie par l'application [Preuveneers *et al.*, 2007].

La programmation par composants habilite l'utilisation de l'adaptation, à la fois au niveau d'un composant en changeant ses paramètres d'exécution, et au niveau de l'assemblage en remplaçant les composants ou en modifiant leurs interconnexions [Chefrour, 2005]. Ces modifications peuvent être mises en œuvre avant le déploiement initial de l'application afin de changer les fonctionnalités disponibles, conformément, par exemple aux caractéristiques matérielles du dispositif mobile.

Une fois l'application mobile installée, la reconfiguration dynamique peut être entamée pour remplacer un composant ou changer ses paramètres d'exécution en fonction du contexte d'exécution de l'application [Zachariadis *et al.*, 2006] [Chan *et al.*, 2003]. La faiblesse du

niveau de batterie peut amener un système, par exemple, à changer le paramétrage de son composant d'accès au GPS afin de réduire la consommation d'énergie (par exemple, en diminuant de la fréquence de consultations du capteur).

Le contrôle du cycle de vie est un autre outil important dans le processus d'adaptation. Télécharger un nouveau composant, désactiver un composant en exécution, et activer un composant installé peut contribuer à adapter le comportement global d'une application mobile, et d'autres systèmes accédés par cette application (tels qu'un serveur de contenu) [Zachariadis *et al.*, 2006].

5.5.1.2 OSGi

L'OSGi Alliance[70] est un groupe crée en 1999, rassemblant des entreprises telles que Nokia, Philips, IBM et Motorola. Il propose et soutient la spécification ouverte d'une plate-forme de services reposant entièrement sur le langage Java. Le principe de base d'OSGi est d'offrir une plate-forme de services qui permet le contrôle complet du cycle de vie d'un service ou d'une application (un assemblage de services). OSGi vise, en particulier, les environnements avec des contraintes de ressources et possédant éventuellement une connexion réseau, comme les téléphones mobiles, les passerelles résidentielles et véhiculaires [Wang *et al.*, 2005]. Son objectif principal est d'offrir l'installation et la maintenance à distance de services déployés.

La plate-forme OSGi était destinée initialement aux passerelles (*gateway* en anglais) qui concentrent de grandes quantités de matériel (par exemple, un hub qui relie tous les capteurs repartis dans une salle de réunion « intelligente »), et qui nécessitent de contrôler ces matériels à distance (par exemple, activer un capteur de luminosité). Cependant, aujourd'hui OSGi est utilisé pour le développement et l'exécution d'applications dynamiques sur divers types de support (PDAs, boîte à connexion, un ordinateur, etc.). Plusieurs implémentations du *framework* de la spécification OSGi existent, parmi lesquels nous pouvons citer :

- **OSCAR**[71]. L'équipe Adele du laboratoire LIG a développé une implantation de code source libre d'OSGi, nommée OSCAR. Cette implémentation a été reprise et étendue dans le projet Apache Felix[72] qui possède une implémentation pour la plate-forme Android[73].
- **Concierge**[74]. L'implémentation la plus légère d'OSGi, Concierge occupe uniquement 80KB et est destinée aux dispositifs mobiles qui possèdent la machine virtuelle Java CDC (des PDAs et moins d'une dizaine de smartphones Nokia).
- **Equinox**[75]. L'environnement de développement multi-plate-forme Eclipse repose sur une implémentation de la spécification OSGi : Equinox. L'objectif est de rendre plus facile l'ajout et l'exclusion des *plugins* sur l'environnement Eclipse.

Bien que l'OSGi soit une plate-forme orientée services, conformément aux principes de SOA (*Service Oriented Architecture*)[76], elle peut être également vue comme une architecture

[70] http://www.osgi.org/

[71] http://forge.ow2.org/projects/oscar/

[72] http://felix.apache.org/site/index.html

[73] https://opensource.luminis.net/wiki/display/SITE/OSGi+Android

[74] http://concierge.sourceforge.net/

[75] http://www.eclipse.org/equinox/

[76] L'Architecture Orientée Services (AOS ou SOA – Service-Oriented Architecture en anglais) permet d'organiser des programmes/systèmes isolés dans un ensemble de services interconnectés, accessibles par une interface et des protocoles standard [Papazoglou, 2003]. Le service consiste en une fonction ou une fonctionnalité avec un ou plusieurs objectifs bien définis. Nous identifions un triplet sur l'architecture SOA: i) un annuaire de services, ii) les fournisseurs/producteurs de services, et iii) les clients/consommateurs des services. Le principe de base d'une architecture SOA est de permettre la

orientée composants dont le modèle de connexion des composants (les *bundles*) sont les services [Wang *et al.*, 2005]. Un composant sur OSGi est nommé *bundle* et est déployé sous la forme d'un fichier Jar. Un *bundle* englobe des ressources (des images, des fichiers textes), des classes Java, un fichier de description du composant (un manifeste). Un *bundle* peut être « branché » à un framework OSGi qui, pour sa part, offre une interface (un shell ou une interface graphique) de gestion du cycle de vie des *bundles*. Un *bundle* peut fournit des services, peut exécuter du code Java et peut utiliser également des services fournis par d'autres *bundles* ou par l'infrastructure elle-même. La Figure 41 présente le cycle de vie d'un *bundle*. L'installation, la désinstallation, l'activation et l'arrêt d'un *bundle* peuvent être commandé à partir de l'interface de gestion de l'implémentation de la spécification OSGi. Les événements d'activation et, d'arrêt sont transmis à un bundle par le biais de l'appel des méthodes *start* and *stop* d'une classe *activateur* (une classe qui hérite de BundleActivator) existant sur chaque *bundle*.

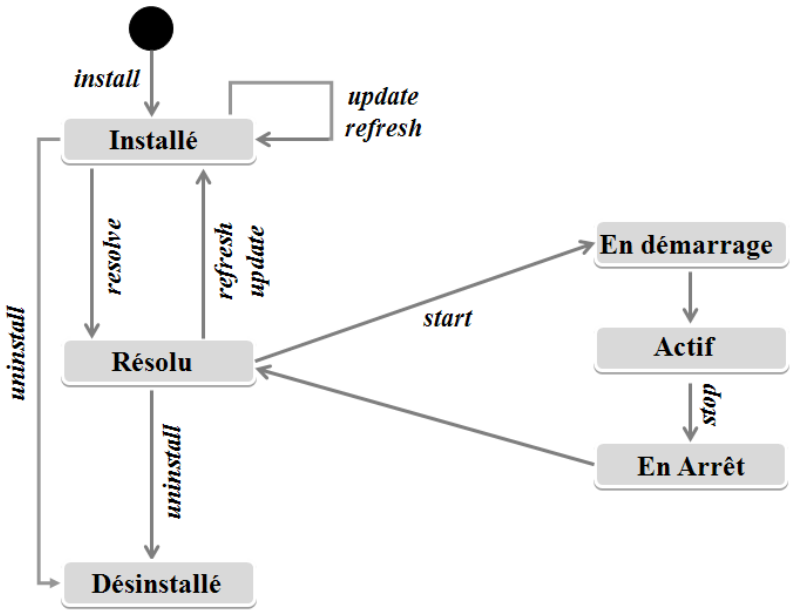


Figure 41- Cycle de vie des *bundles* sur OSGi.

Les *bundles* ne peuvent pas accéder directement aux méthodes des autres bundles. Le mécanisme de services garantit la communication. Les services sur OSGi sont, en réalité, une interface Java bien définie et partagée par les deux composants communicants. Cette interface Java doit être implémentée par une classe à l'intérieur du composant fournisseur du service. Le fournisseur du service doit, en plus, enregistrer cette interface auprès de l'annuaire de service de l'infrastructure.

Un client désirant ce service doit consulter l'infrastructure pour savoir s'il existe au moins un composant actif qui fournit un service possédant cette interface. Cette indirection dans le code facilite la reconfiguration dynamique des composants et oblige les développeurs à créer des codes conscients de la dynamicité des services sur OSGi. Un mécanisme de communication par événement est également fourni par l'infrastructure, un client peut s'enregistrer et être informé à chaque fois qu'un nouveau composant enregistre un service. De plus, un autre mécanisme d'événement suivant le patron de conception *White Board*[77] est

publication, la découverte, l'utilisation et la composition des services de manière dynamique. Bien que le terme SOA soit fortement associé à la technologie de Services Web, OSGi suit également ces principes [Kriens, 2007].

[77] http://www.theserverside.com/tt/articles/article.tss?l=WhiteboardForOSGi

implémenté par OSGi. Ce mécanisme offre l'échange d'événements entre des composants et garantit une meilleure lisibilité du code.

Nous présentons dans les prochaines sections les approches d'adaptation et de déploiement adaptatif, dont la plupart utilisent des mécanismes similaires ou reposent sur des infrastructures orientées composant.

5.5.2 Déploiement adaptatif d'applications mobiles

Le déploiement d'applications exige un engagement conséquent des utilisateurs sur la majorité des plates-formes mobiles comme nous avons décrit dans le chapitre précédent. Un déploiement idéal sur ces plates-formes assurerait [Cong *et al.*, 2008] [Ayed *et al.*, 2008] [Ajmani *et al.*, 2006]:

- *La prise en compte des ressources limitées des dispositifs mobiles.* Le déploiement doit éviter au maximum d'alourdir l'exécution des applications mobiles afin de prévenir un arrêt inespéré lors d'une des étapes du déploiement.
- *L'indépendance d'un modèle spécifique de dispositif et de plate-forme.* La généricité de l'approche du déploiement doit permettre de fonctionner sur plusieurs terminaux mobiles.
- *L'adaptation de ces activités.* Le déploiement doit prendre en compte les propriétés du dispositif dans la réalisation de ces étapes.
- *L'absence d'intervention de l'utilisateur.* De l'installation à la désinstallation, le déploiement doit être le plus automatisé possible.
- *La possibilité de retourner en arrière.* Dans le cas d'une mise à jour erronée ou non acceptable par l'utilisateur, le gestionnaire de déploiement doit récupérer la version ou l'étape précédente de l'application.

Plusieurs approches ont émergé afin de combler, dans une certaine mesure, ces pré-requis d'un déploiement idéal d'une application mobile [Cong *et al.,* 2008][Ayed *et al.*, 2008] [Donsez, 2006] [Fjellheim, 2006]. L'hétérogénéité des dispositifs mobiles et l'intervention récurrente de l'utilisateur sont les inconvénients du déploiement les plus étudiés par ces propositions. Des mécanismes d'adaptation aux caractéristiques des dispositifs mobiles, et des techniques d'automatisation des étapes de déploiement sont les stratégies les plus exploitées. Ces propositions sont décrites plus en détail dans les sections suivantes.

5.5.2.1 OTA-PSD (Over The Air Provider-initiated Software Deployment)

L'approche OTA-PSD [Cong *et al.,* 2008] repose sur une extension du modèle de déploiement OTA(Over-The-Air) de la plate-forme J2ME MIDP afin de rendre le déploiement des MIDlets plus automatisé. Les auteurs proposent la création d'un portail d'applications mobiles pour faciliter l'installation et la mise à jour des MIDlets. Ce portail possède un registre des applications disponibles, un registre des profils des modèles de dispositifs, et un registre de préférences des utilisateurs du portail. Ceux-ci doivent renseigner manuellement sur les caractéristiques de leurs dispositifs, et peuvent également s'inscrire à un service de notification d'applications. Les utilisateurs du portail indiquent les catégories d'applications qui les intéressent, et les capacités de leurs dispositifs à partir d'une application mobile préinstallée sur leurs terminaux (le *Software Manager*). Ces informations sont prises en compte par le service de notification, la contribution principale de la proposition OTA-PSD, qui est illustrée sur la Figure 42.

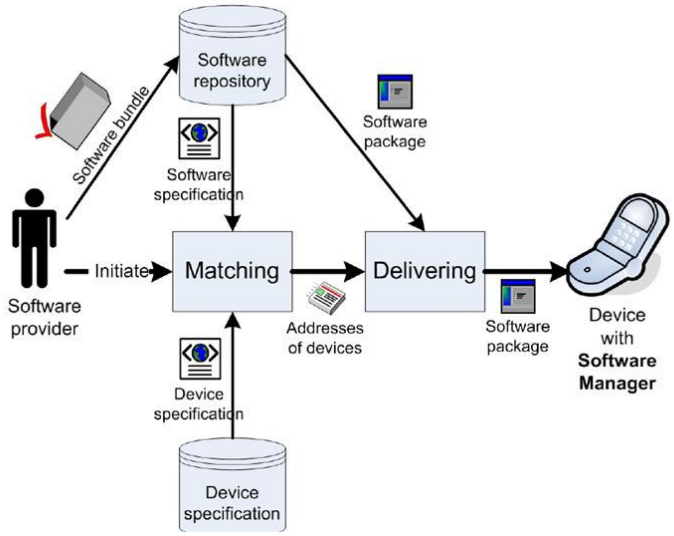


Figure 42 - Processus de déploiement adaptatif OTA-PSD (Source :[Cong *et al.*, 2008]).

Lorsqu'un fournisseur rend disponible une application sur le portail (Figure 42), le serveur du portail analyse quels sont les utilisateurs potentiellement intéressés par la catégorie de cette application. Ces utilisateurs doivent en plus disposer d'un dispositif en mesure de l'activer.

Le processus de filtrage des MIDlets repose sur une spécification des métadonnées des applications. Ces métadonnées, représentées sous forme de document XML, contiennent la description de la catégorie d'une application mobile et les pré-requis à son exécution (par exemple, exigence de la plate-forme J2ME MIDP version 2.0). Le serveur du portail confronte ainsi les métadonnées des applications aux profils des dispositifs mobiles des utilisateurs. Dans le cas d'un succès, un message de notification est ensuite envoyé au logiciel de déploiement (le *Software Manager*) et les utilisateurs peuvent, s'ils le souhaitent, démarrer l'installation de l'application.

La mise à jour d'une application mobile sur le portail déclenche un processus similaire de notification aux utilisateurs. Tous les usagers du portail qui possèdent une ancienne version de l'application sont avertis automatiquement dès que le logiciel Software Manager est activé. Ce logiciel permet ensuite d'acquérir et de télécharger la nouvelle version du MIDlet.

Malgré l'obligation d'une description manuelle des dispositifs mobiles de la part des usagers, l'approche OTA-PSD diminue la probabilité d'un déploiement d'une application qui ne pourrait pas s'exécuter correctement sur le dispositif de l'utilisateur.

5.5.2.2 Approches orientées composants pour le déploiement adaptatif

Contrairement à OTA-PSD, qui adapte le déploiement d'applications mobiles monolithiques, d'autres propositions reposent sur le déploiement adaptatif d'applications orientées composants [Ayed *et al.,* 2008] [Donsez, 2006] [Fjellheim, 2006]. Ces approches supposent qu'un service (c'est-à-dire la fonctionnalité implémentée par le composant et fournie par le biais de son interface) peut posséder différentes implémentations (la même interface implémentée par plusieurs composants), et que, chaque implémentation est plus adaptée à un contexte d'utilisation (par exemple, une implémentation pour les PDAs dotés de la plate-forme J2ME CDC, et une autre pour les smartphones J2ME MIDP). Dans ces approches, une application n'est plus décrite comme un assemblage d'implémentations de composants. Elle est, en revanche, une méta-application définie comme une composition de services (ou des composants génériques). Lors d'une demande d'installation, un gestionnaire de déploiement décide quels sont les composants réels (les implémentations) qui fourniront les services lors de l'exécution de l'application. Le gestionnaire de déploiement est capable de

générer ainsi plusieurs configurations différentes de l'application mobile, chacune adaptée à un contexte d'exécution particulier. La Figure 43 illustre un schéma généralisé du déploiement dans ces propositions adaptatives.

La première étape avant l'installation d'une application est la caractérisation de son futur contexte d'utilisation qui, dans ces approches, est réduit à l'identification de la catégorie du dispositif (PDA, téléphone, TV, etc.), de sa plate-forme d'exécution (J2ME MIDP, Java CDC, J2SE, etc.), et des préférences de l'utilisateur (par exemple, le type d'affichage désiré). La découverte du futur contexte d'utilisation n'est pas nécessairement automatique. Le modèle du dispositif, par exemple, peut être informé par le responsable du déploiement ou par l'utilisateur qui télécharge l'application.

Des métadonnées sont ajoutées aux fichiers de configuration des applications (la description de la méta-application ou son plan de déploiement), afin de spécifier les conditions favorables d'exécution de chaque implémentation des composants (tel que la plate-forme pour laquelle le composant est développé). Ces informations, conjointement à la caractérisation du futur contexte d'utilisation sont exploitées par le gestionnaire du déploiement dans la génération de l'assemblage final de l'application. Ensuite, le processus de *packaging* est mis en œuvre et l'installation de l'application est complétée. L'assemblage choisi reste immuable tout au long de l'exécution de l'application.

Nous présentons ci-dessous les particularités de ces trois approches.

OTA-3DMA [Fjellheim, 2006]

Fjellheim propose une extension du modèle OTA, nommée OTA-3DMA qui offre l'adaptation du déploiement d'une application mobile en deux niveaux : *i)* l'adaptation des services fournis par l'application, et *ii)* l'adaptation du protocole de déploiement lui-même. OTA-3DMA exploite des informations sur la plate-forme mobile du dispositif cible et les préférences des utilisateurs dans le processus d'adaptation. OTA-3DMA possède deux protocoles de déploiement : un pour J2ME MIDP et un pour la plate-forme J2ME CDC (Personal Java). Le gestionnaire de déploiement sélectionne le protocole en accord avec la plate-forme du dispositif, renseignée manuellement par l'utilisateur. Les méta-applications sont décrites comme une liste de services de l'infrastructure AOS (Active Object Spaces) également proposée par cet auteur. Chaque service possède un triplet de métadonnées {*activité de l'utilisateur*, *plate-forme de fonctionnement*, *taille du code*}.

Lors d'une demande d'application à partir d'un téléphone mobile, le protocole de déploiement J2ME MIDP est activé. Il élimine de la liste de la méta-application tous les services qui ne possèdent pas d'implémentation MIDP, et qui ne sont pas désirés par l'utilisateur. Le gestionnaire de déploiement supprime également les services dont l'implémentation présente une taille supérieure à celle admise par le dispositif. Ensuite, la liste filtrée est exploitée lors du processus de génération du MIDletSuite. Des composants de connexion peuvent être ajoutés au JAR final si une implémentation d'un service exige la communication avec le serveur de l'infrastructure AOS. Un URI de l'application finale est envoyé à l'utilisateur qui peut enfin télécharger et installer l'application.

Les versions CDC de services de l'infrastructure AOS communiquent avec le serveur en utilisant RMI (*Remote Method Invocation*). Le déploiement de ces versions est ainsi différent du déploiement d'une application MIDP. La partie client du code d'un service est envoyée à un middleware sur le dispositif et n'exige pas la génération d'un Jar ou l'addition de composants de communication avec l'infrastructure. Cependant, le processus de filtrage reste similaire à celui destiné aux dispositifs MIDP.

Bien qu'OTA-3DMA offre un déploiement adaptatif d'une application orientée composant, cette adaptation est limitée à un mécanisme d'élimination des services et à une sélection d'implémentations entre deux plates-formes. Ce mécanisme peut ne pas être suffisant pour

empêcher l'échec de l'installation d'une application si, par exemple, un de ses composants utilise une bibliothèque indisponible sur le dispositif de l'utilisateur.

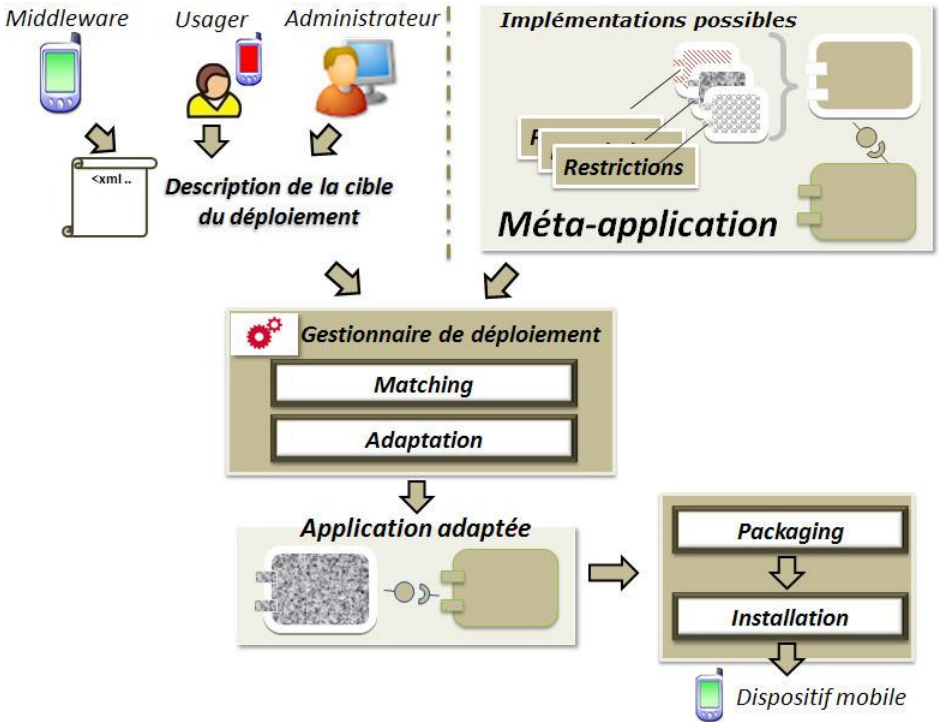


Figure 43 - Schéma généralisé des approches de déploiement adaptatif orientées composants [Ayed *et al.*, 2008] [Donsez, 2006] [Fjellheim, 2006].

[Donsez, 2006]

Nos maisons sont de plus en plus peuplées d'équipements (TV, climatisation, centrale d'alarme, etc.) susceptibles d'être contrôler à distance par un ordinateur ou une centrale numérique. Donsez propose une architecture pour la découverte dynamique et pour le contrôle de ces équipements en couplant OSGi et UPnP (Universal Plug and Play). Les applications sous cette architecture sont des points de contrôle graphiques qui permettent de consulter l'état des équipements d'une maison et de les modifier. Ces applications peuvent être déployées sur une hétérogénéité de dispositifs dotés d'un écran (PDA, télephone mobile, télé avec Java embarqué, etc.). Lorsque la présence d'un nouvel équipement est détectée par UPnP, l'architecture proposée doit déployer un point de contrôle sur le dispositif de l'utilisateur afin d'établir l'interaction entre l'usager et le nouvel équipement.

Les applications sont, en réalité, un assemblage de composants nommés *controlets*. Chaque *controlet* est adapté à un canevas graphique (LCDUI pour MIDP, eSWT pour les PDA, JavaTV pour les téléviseurs). Une *controlet* peut aussi être constituée de sous-composants. Par exemple, une pour chaque fonctionnalité de l'équipement. Un ordre et une priorité d'affichage peuvent être également spécifiés entre les sous-composants. La priorité guide le gestionnaire de déploiement dans la sélection des *controlets* les plus importantes lorsque l'écran du dispositif ne permet que l'affichage d'une partie des *controlets,* ou bien encore lorsque l'utilisateur souhaite une interface simplifiée. Les métadonnées des *controlets* sont regroupées dans un document XML étendant les informations de déploiement. Pour tout nouvel équipement découvert dans le réseau UPnP, l'architecture démarre une opération de courtage pour déterminer la *controlet* la plus adéquate pour l'équipement découvert. Parmi la liste des composants disponibles, l'algorithme de courtage élimine d'abord les composants qui ne satisfont pas les contraintes du dispositif d'affichage de l'utilisateur, tels que le *toolkit* graphique et le système d'exploitation si le composant embarque des bibliothèques natives.

L'algorithme cherche ensuite le composant le plus spécialisé pour le contrôle de l'équipement découvert. À la fin du processus de courtage, un *bundle* OSGi est généré et déployé sur le dispositif de l'utilisateur. La Figure 44 illustre les deux applications générées lors du déploiement pour contrôler un même équipement.

Figure 44 - Deux applications différentes qui contrôlent le même dispositif UPnP (Source : [Donsez, 2006]).

L'architecture proposée par Donsez est destinée à un domaine (la domotique) assez distant du domaine d'applications multimédias et multi-capteurs. Cependant, le courtage dynamique d'un assemblage de composants peut être un mécanisme précieux pour alléger le problème d'hétérogénéité des applications mobiles multimédias.

CADeComp- [Ayed *et al.*, 2008]

CADeComp offre un modèle enrichi de description de plans de déploiements d'applications orientées composants. Le fonctionnement des composants est basé sur un modèle de données permettant la définition de la variabilité des paramètres de déploiement en fonction des informations du futur contexte d'exécution de l'application. Par exemple, le concepteur peut spécifier qu'un composant doit être installé seulement si le contexte de l'utilisateur est « Pocket PC ».

Contrairement aux approches précédentes, le plan de déploiement (similaire à une méta-application) proposé par Ayed permet de spécifier la variabilité d'une application à plusieurs niveaux. Le concepteur peut décrire des changements de la structure de l'application (les dépendances, les composants et les connexions à activer, la localisation des composants) en fonction des caractéristiques du dispositif cible. Il peut également définir des variations des paramètres fonctionnels et non-fonctionnels d'un composant en accord, par exemple, avec la langue de l'utilisateur. Les méta-informations de variabilité sont décrites à l'aide de contrats d'adaptation reliés à un modèle par objets (UML), et représentés sous forme d'un document XML. La Figure 45 illustre un fragment d'un plan de déploiement qui décrit les restrictions d'exécution du composant *firstaidworkerimplem.*

```
<componentpackagedescription name="viewcomponentpackage"   id="partnerpack">
  <componentimplementationdescription name="firstaidworkerimplem" id="rescimpl">
     <monolithiccomponent>
        <placementconstraint type=strong   relevantcontextref="PocketPCOS">
        <placementconstraint type= strong  relevantcontextref="firstaidworkerProfile">
      </monolithiccomponent>
  </componentimplementationdescription>
</componentpackagedescription>
```

Figure 45- Un fragment du plan de déploiement proposé dans l'architecture CAdeComp.

Les dispositifs mobiles des utilisateurs doivent contenir un intergiciel implémenté en CORBA. Lors du déploiement d'une application, l'intergiciel capture les informations sur

l'environnement d'exécution du terminal mobile (le modèle de dispositif, l'utilisateur) et envoie ces informations au gestionnaire de déploiement. Celui-ci utilise les informations sur le contexte d'exécution pour l'évaluation des contrats du plan de déploiement, et, ensuite, génère l'application qui doit être déployée. La configuration générée ne change pas lors de l'exécution de l'application.

CORBA est une infrastructure à composant inadaptée aux environnements limités des dispositifs mobiles en raison de sa forte consommation des ressources (mémoire, batterie, etc.) [Preuveneers *et al.*, 2007]. Ceci a probablement empêché ces auteurs de tester CADeComp sur un dispositif mobile (un test sur un ordinateur de bureau a été effectué). Néanmoins, le plan de déploiement proposé par ces auteurs offre une représentation suffisamment riche pour décrire les variabilités d'un assemblage de composants d'une véritable application mobile.

5.5.3 Adaptation dynamique d'une application mobile

Les propositions de déploiement adaptatif présentées précédemment adaptent les applications mobiles quelques instants avant leur transfert vers le dispositif mobile. Ce processus d'installation adaptative est activé soit à partir d'une demande des utilisateurs, soit par le gestionnaire de déploiement lui-même (comme dans la proposition OTA-PSD). Toutefois, dans ces propositions, l'application n'est pas adaptée pendant la durée de son exécution. Une fois la configuration de l'application choisie, elle reste inchangée jusqu'à sa mise à jour ou sa désinstallation.

D'autres propositions s'intéressent à l'adaptation dynamique d'une application mobile après son activation. Or, l'environnement d'exécution des applications mobiles est en constante évolution. Les ressources utilisées par une application (mémoire vive, batterie, réseau, etc.) varient énormément et peuvent compromettre le correct fonctionnement de l'application. Une bande passante moins performante ou la surcharge de la mémoire vivre peuvent ralentir l'exécution d'une application. Le déplacement de l'utilisateur affecte également l'ensemble de services fournis par une application mobile. Par exemple, un service de géolocalisation par GPS (tel que l'indication de la position du dispositif mobile sur une carte) peut ne plus fonctionner correctement dès que l'utilisateur pénètre à l'intérieur d'un bâtiment. Cette dynamicité de l'environnement d'exécution oblige généralement les concepteurs d'applications mobiles à prévoir dans le code de l'application des mécanismes d'adaptation capables de faire réagir l'application à la dégradation ou à l'amélioration de son environnement d'exécution. Cependant, l'intégration directe de méthodes d'adaptation au code source d'une application limite la réutilisation de ces mécanismes et rend plus difficile l'extensibilité de l'application elle-même [Chefrour, 2005][Baudaulf *et al.*, 2007][Kirsch, 2006].

Des infrastructures d'adaptation dynamique (des *middlewares*, des *frameworks*, etc.) sont ainsi apparues afin d'alléger la tâche de développement des mécanismes d'adaptation. L'idée de base de ces propositions est de séparer au maximum le processus d'adaptation du code fonctionnel d'une application mobile [Zachariadis *et al.*, 2006][Baudaulf *et al.*, 2007]. En général, les applications délèguent à l'infrastructure la surveillance de son environnement d'exécution et le contrôle de l'adaptation.

Parmi ces propositions, nous retenons celles qui adaptent des applications mobiles et orientées composants [Chan *et al.*,2003][Chefrour, 2005][Kim *et al.*, 2006][Zheng *et al.*, 2007]. Ces approches se servent des mécanismes de reconfiguration dynamique des infrastructures à composants et accomplissent le processus d'adaptation d'une application mobile sans la nécessité d'arrêter son exécution.

Nous présentons brièvement dans cette section les multiples facettes de la reconfiguration dynamique, et, par la suite, son utilisation pour l'adaptation dynamique d'une application mobile.

5.5.3.1 Reconfiguration dynamique

Dans une infrastructure à composants, la reconfiguration dynamique est principalement mise en œuvre en utilisant [Rutheford *et al.*, 2002][Ketfi, 2004]:

- *Le changement des paramètres de configuration des composants*. Certains modèles de composant acceptent le passage de paramètres de configuration à un composant par un fichier de configuration, ainsi que par l'appel d'une méthode particulière du modèle. L'administrateur du système, ou, un autre composant, peut modifier les paramètres d'un composant lors de son exécution afin de changer son comportement dynamique.
- *Le remplacement d'un composant*. La reconfiguration dynamique dans ce cas consiste à modifier l'implémentation d'un composant en exécution, sans modifier l'interface des services qu'il offre. Généralement, ce changement est réalisée grâce à la substitution du composant par un autre déjà présent et possédant le même contrat. Le système global, ou l'administrateur du système, peut également télécharger une nouvelle version du composant et, ensuite, procéder à la reconfiguration. Dans ce cas, la reconfiguration dynamique est une mise à jour partielle du système.
- *La modification de l'assemblage*. Une modification des interconnexions entre les composants et les changements de leurs interfaces sont également d'autres types de reconfigurations dynamiques possibles. Cependant, ces modifications sont plus rares et plus complexes à réaliser sans la réinitialisation du système global.
- *La modification de la localisation (migration)*. Une application orientée composant peut s'exécuter de manière distribuée sur plusieurs nœuds (tels qu'un dispositif mobile et un serveur). La modification de la localisation correspond à la migration des instances de composants d'un nœud d'exécution vers un autre site pour la répartition des charges par exemple. Ce type de reconfiguration n'affecte pas l'architecture logique (l'assemblage) de l'application. Néanmoins, la communication, entre les composants ayant migré et les autres instances, peut être modifiée selon la nouvelle localisation des composants.

Les objectifs de la reconfiguration dynamique sont très variés [Ketfi, 2004]. Elle peut être *correctionnelle* en modifiant un composant ou une configuration qui a causé un disfonctionnement du système. La reconfiguration dynamique peut également aider *l'évolution* du système en offrant un mécanisme d'ajout de nouvelles fonctionnalités sans l'arrêt de l'application. De plus, la reconfiguration peut être *perfective* afin d'améliorer les performances d'un système.

La reconfiguration dynamique soulève principalement trois défis : i) la décision de démarrer la reconfiguration, ii) la mise en œuvre de la reconfiguration qui peut comprendre un support de transfert d'état, et ii) la maintenance de la cohérence du système après la reconfiguration [Polakovic *et al.*, 2005][Ketfi, 2004]. Par exemple, changer uniquement une implémentation d'un composant sans modifier son interface ne garantit pas l'absence d'une modification de la « sémantique » du fonctionnement d'un composant. Par conséquent, la reconfiguration d'un composant pourra éventuellement exiger en plus le remplacement de ses clients afin de garantir la cohérence du système global.

5.5.3.2 Approches de reconfiguration dynamique d'une application mobile

Les plates-formes mobiles n'offrent quasiment pas de mécanismes pour le support à l'adaptation dynamique. Elles se limitent à fournir aux applications des avertissements de modifications de quelques configurations du dispositif (telles que l'inclinaison de l'écran sur iPhone OS) et des bibliothèques pour inspecter l'état des variables dynamiques du dispositif hébergeur de l'application. Les concepteurs sont obligés de coder eux-mêmes les mécanismes d'adaptation.

Des approches d'aide à la conception d'applications mobiles adaptatives ont ainsi vu le jour progressivement [Chan *et al.*, 2003][Chefrour, 2005][Zachariadis *et al.*, 2006][Kim *et al.*, 2006][Zheng *et al.*, 2007] [Hens *et al.*, 2007][Louberry *et al.*, 2008]. Nous étudions dans cette section ces propositions qui profitent des techniques de reconfiguration dynamique pour compléter le processus d'adaptation dynamique. Ces approches orientées composants possèdent des intergiciels (en anglais, *middleware*) qui fournissent des méthodes d'adaptation et de surveillance des variables dynamiques du dispositif afin de rendre plus facile le développement d'applications mobiles adaptatives.

SATIN [Zachariadis *et al.*, 2006]

SATIN est une version allégée du modèle à composants CORBA et est destinée à la plate-forme J2ME CDC (Personal Java). SATIN ne contient pas lui-même de techniques d'adaptation prédéfinies. L'infrastructure délègue au concepteur de l'application le codage du processus d'adaptation. Les concepteurs bénéficient en compensation des méthodes de découverte des composants, un système d'avertissement (échange de messages entre les composants), et la possibilité de reconfigurer dynamiquement une application en téléchargeant et en instanciant des nouveaux composants. Ces trois méthodes peuvent être combinées afin de créer une application mobile adaptative.

MobiPADS [Chan *et al.*, 2003]

MobiPADS est un intergiciel J2SE réflexif[78] pour l'exécution d'applications mobiles organisées selon l'architecture client-serveur (dont le client est hébergé sur le dispositif mobile). Contrairement à SATIN, l'intergiciel MobiPADS possède des mécanismes pour adapter dynamiquement une application. Les applications s'exécutant sur MobiPADS sont un enchaînement de services (une séquence de services dont les données sortant du premier service sont les entrées du service suivant, etc.). Un service, nommé *mobilet*, possède deux composants, un sur le dispositif mobile, et un autre sur le serveur. L'intergiciel surveille des propriétés dynamiques du dispositif : la mémoire vivre, le réseau, la capacité de stockage, et le niveau de la batterie. Il offre un mécanisme de notification d'événements qui font référence à ces variables surveillées. Le concepteur d'une application établit dans un document XML, le profil de l'application, un ensemble des événements et les actions devant être entamées par l'intergiciel. Ce profil d'application définit également l'enchaînement initial des *mobilets* et plusieurs séquences alternatives.

Un événement est décrit par un langage de règles conditionnelles (de type Si-Alors) qui contient les mots réservés « AND », « OR », « > », « < », « = ». Par exemple, l'événement « réseau-trop-lent » peut être défini en XML comme « Network_Delay < 100 » où *Network_Delay* est une variable surveillée par l'intergiciel. Lorsqu'un événement est validé, deux actions sont possibles : une notification à l'application mobile et une reconfiguration d'enchaînement des services. La notification est un simple appel d'une méthode de l'application qui peut ensuite décider de s'adapter. La reconfiguration de l'enchaînement est plus complexe. L'intergiciel cherche sur le document XML l'alternative à la séquence de service actuelle qui a été associée à l'événement détecté. Une fois établie la séquence,

[78] La caractéristique de réflexion dans un système fait référence à la capacité de surveiller ses conditions d'exécution et modifier lui-même son comportement [Smith, 1982].

l'intergiciel démarre la reconfiguration. Afin de compléter ce processus, MobiPADS peut changer l'ordre de la séquence de services, supprimer un service ou en ajouter un nouveau. Des messages sont envoyés aux services avant qu'une suppression soit faite, le *mobilet* peut ainsi finir son exécution en sécurité.

ACEEL [Chefrour, 2005]

ACEEL est un modèle de composant pour la création d'applications mobiles auto-adaptatives. Différemment de MobiPADS, ACEEL ne permet pas de permuter entre deux assemblages de composants. Le paramétrage et le changement d'une implémentation d'un composant sont les uniques manières de reconfigurer dynamiquement une application. Le modèle ACEEL utilise la réflexion en séparant un composant en deux niveaux : le niveau de base et le méta-niveau. Le niveau de base contient les différents comportements fonctionnels d'un composant en fonction du contexte d'exécution de l'application (voir Figure 46). Ces comportements sont modélisés en utilisant le patron de conception *Strategy* [Gamma *et al.,* 1995]. Une classe (*Context*) fournit aux clients d'un composant une interface et cache à ses clients l'accès à l'implémentation de sa démarche fonctionnelle. Le méta-niveau du composant contrôle le processus d'adaptation et est intégré à l'infrastructure à composants. Le concepteur décrit des politiques d'adaptation à l'aide d'un script XML (voir exemple Figure 46 – I). Ce script est interprété par l'infrastructure qui surveille les propriétés dynamiques de l'environnement d'exécution (réseau, mémoire et batterie). Lorsqu'une politique d'adaptation est validée, l'infrastructure consulte l'état du composant (variable *envState*) et appelle la méthode *adaptBehaviour* afin que le composant change son comportement. Le concepteur du composant est responsable du code qui effectue la reconfiguration et doit garantir également la cohérence de son changement. ACEEL est implémenté en Python qui s'exécute sur les dispositifs Windows Mobile. L'infrastructure a été conçue principalement pour l'adaptation au flux de données multimédias (vidéos, images). Néanmoins, son modèle à composants peut être repris pour adapter d'autres types de comportements d'une application mobile.

I)

```
<res name=Bandwidth>
<exceeds value=100kbs
action=change_param(compression_rate=1)/
>
<falls value=100kbs
action=change_param(compression_rate=7)/
>
</res>
<res name=ErrorRate>
<exceeds value=10^-6
action=change_to_behavior(M-JPEGCodec)/>
<falls value=10^-6
action=change_to_behavior(MPEGCodec)/>
</res>
```

II)

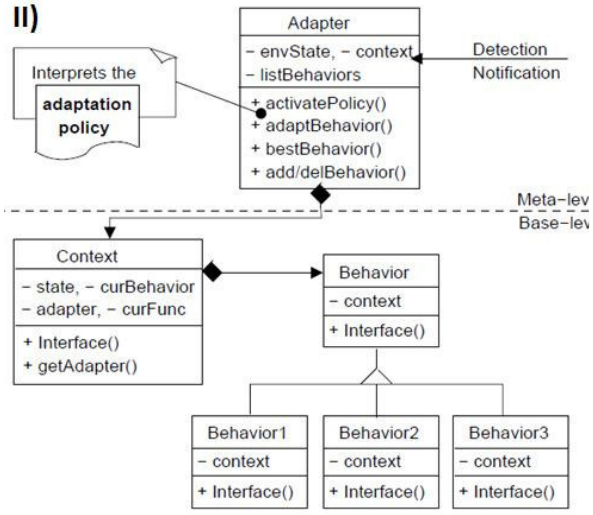


Figure 46 - Proposition ACEEL: I) script des politiques d'adaptation, II) Diagramme de classes avec la description des alternatives de comportement d'un composant.

Le modèle d'adaptation proposé par ACEEL encapsule à l'intérieur d'un composant ses différentes manières d'exécuter. D'autres approches [Kim *et al.*, 2006][Zheng *et al.*, 2007] [Hens *et al.*, 2007][Louberry *et al.*, 2008] préconisent l'emploi d'un composant pour chaque comportement et l'explicitation de ses restrictions d'exécution et de ses qualités de service . À l'instar de propositions d'installation adaptative (Section 5.5.2.2), ces approches supposent qu'un service (la fonctionnalité fournie par le composant) peut être implémenté de diverses manières tout en conservant son interface de connexion et que chaque implémentation du

service est plus adaptée à un contexte d'exécution. Généralement, ces approches considèrent aussi la définition de la part du concepteur de multiples configurations d'une application (la liste de services, les interconnexions, etc.), chacune plus adéquate à un contexte d'exécution particulier.

La Figure 47 illustre le fonctionnement d'une de ces infrastructures : la plate-forme de reconfiguration proposée par Louberry *et al.* [Louberry *et al.*, 2008]. Le schéma de cette proposition peut être également vu comme une généralisation de la reconfiguration adaptative de toutes les autres approches. Les intergiciels de ces propositions se chargent de surveiller l'environnement d'exécution et de tester en boucle si le contexte actuel de l'environnement n'exige pas une reconfiguration.

Par exemple, Zheng *et al.* [Zheng *et al.*, 2007] proposent une extension du middleware StarCCM[79] (une implémentation C++ de CORBA) pour l'adaptation dynamique. Ces auteurs établissent un langage pour la description d'un plan de déploiement enrichi. Ce plan spécifie l'assemblage de l'application, les différentes implémentations des composants, la localisation de leurs sites de déploiement et les valeurs des propriétés de configuration des composants. Un concepteur peut associer à chaque paramètre du plan de déploiement un ensemble de règles conditionnelles qui définissent les diverses valeurs du paramètre en fonction des variables dynamiques de l'environnement d'exécution de l'application. L'intergiciel analyse ces conditions indéfiniment et change les paramètres lorsqu'une condition est validée. Une règle conditionnelle peut exiger la modification de l'implémentation d'un composant. Pour la mise en œuvre, l'intergiciel examine en plus si les restrictions d'exécution de l'implémentation sont satisfaites (par exemple, mémoire vive > 300Kb). De plus, les composants sont groupés en deux catégories : les obligatoires et les optionnels. Si la restriction d'exécution d'un composant optionnel n'est pas satisfaite, le composant est stoppé tandis que pour un composant obligatoire, toute l'application est arrêtée.

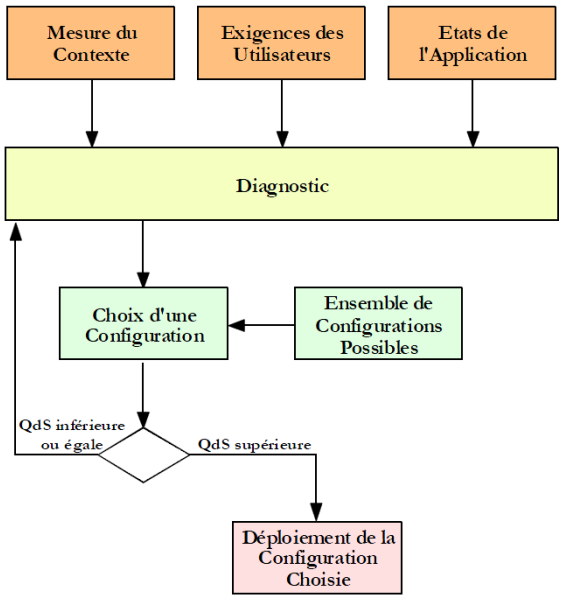


Figure 47- Schéma de reconfiguration dynamique proposée par Louberry *et al.* [Louberry *et al.*, 2008].

Les auteurs [Hens *et al.*, 2007] et [Louberry *et al.*, 2008] se sont plutôt intéressés à la reconfiguration d'une application pour répartir sa charge d'exécution. L'idée de base est de choisir entre deux configurations : i) configuration d'exécution locale, dans laquelle la plupart du code de l'application est activé sur un dispositif mobile limité en ressources (capteur, téléphone), ii) la migration du code fonctionnel vers un dispositif doté de plus de capacités.

[79] http://starccm.sourceforge.net/

Hens *et al.* [Hens *et al.*, 2007] propose, de plus, de simuler chacune des répartitions possibles en estimant les capacités de calcul, et de mémoire des nœuds aussi que le débit du réseau. L'intergiciel calcule ensuite l'impact de chaque configuration possible sur la latence de l'application globale et choisit celle dont l'impact est le plus faible. Un inconvénient de cette approche est qu'elle repose sur un problème a priori NP-complet et les estimations des propriétés dynamiques ne sont pas toujours fiables.

Les propositions de reconfiguration dynamique présentées jusqu'à ce point utilisent des techniques simples pour la description des choix possibles de configurations. D'autres propositions [Kim *et al.*, 2006][Wei *et al.*, 2008] s'appuient sur des ontologies pour représenter la variabilité de composants et pour guider la décision d'activer la reconfiguration et de comment la réaliser. Par exemple, Wei *et al.* [Wei *et al.*, 2008] offrent un vocabulaire OWL pour représenter les métadonnées des composants d'une application et ses restrictions d'exécution. Au moment de l'exécution, un intergiciel met en place des mécanismes de raisonnement afin de décider quels sont les composants qui peuvent être activés.

5.6 Synthèse

Dans ce chapitre nous avons présenté un panorama des approches sensibles au contexte qui définissent, représentent et capturent le contexte d'utilisation. Particulièrement, nous avons fixé notre attention sur les propositions d'adaptation d'applications mobiles dont les solutions peuvent être étendues afin d'atténuer les problèmes liés à l'hétérogénéité de dispositifs mobiles.

Une section a été consacrée à la présentation des notions de *contexte d'utilisation,* et à la description des approches de représentation de l'information contextuelle. On peut regretter que la caractérisation du contexte pour un seul usage lors d'un processus d'adaptation. Le manque de généricité des modèles présentés ne permet pas leurs extensions en vue d'une utilisation pour l'annotation contextuelle de documents multimédias.

Une autre critique que nous adressons aux propositions de représentation de contexte est qu'elles sont dédiées soit i) à une description exclusive et extensive des dispositifs mobiles (tel que UaProf), soit ii) à une caractérisation plus large englobant d'autres éléments contextuels, mais avec une simplification de la description des dispositifs d'accès.

À l'instar des solutions de représentation des métadonnées, les ontologies de description de contexte sont les plus prometteuses, car les systèmes sensibles au contexte peuvent exploiter les mécanismes d'inférence (classification, règles, etc.) pour augmenter le niveau sémantique de l'information à propos du *contexte d'utilisation.* Cependant, les ontologies, notamment celles destinées à la description de dispositifs mobiles, supportent mal le passage à l'échelle, puisqu'elles exigent la création en amont des profils des possibles dispositifs susceptibles d'accéder au système. De plus, la capture en boucle d'informations sur l'état d'exécution du dispositif (mémoire, batterie, etc.) peut contribuer à l'épuisement des ressources du dispositif. Malgré ces inconvénients, une combinaison du pouvoir d'expression et de raisonnement de ces ontologies avec un registre de profils tels que WURFL et Marjory, peut apporter une solution de haut niveau sémantique à l'identification des capacités d'un dispositif mobile (ses propriétés statiques) et à la caractérisation de son état actuel (ses propriétés dynamiques) lors de l'utilisation d'une application.

Les solutions concernant les architectures (cadriciels et canevas) de capture et de gestion de contexte peuvent être généralisées en vue d'une double utilisation : i) l'adaptation d'applications mobiles, et ii) le support à l'exécution des services de gestion de multimédia personnel. On peut regretter que, toutefois, l'inadaptation de la plupart de ces architectures à une exécution sur les téléphones mobiles en raison des ressources limitées (mémoire et la batterie), de l'hétérogénéité des bibliothèques supportés sur ces dispositifs et de l'approche de

tout-ou-rien des architectures. En conséquence, il y a un réel besoin d'outils pour alléger ce processus de conception et de développement d'applications mobiles et sensibles au contexte.

Enfin, nous avons présenté des approches d'adaptation d'applications mobiles et de déploiement adaptatif qui peuvent être étendues vers une proposition d'adaptation d'applications mobiles et sensibles au contexte. Le Tableau 5 offre une vue synthétique des propositions de déploiement adaptatif qui ont inspiré notre proposition de thèse. Ces approches exigent l'installation préalable d'un intergiciel sur les dispositifs mobiles des utilisateurs, et cet intergiciel n'est pas lui-même adaptable. Or, le développement et le déploiement de cet intergiciel sont limités du fait de la hétérogénéité des dispositifs. De plus, les systèmes de mise en correspondance entre les restrictions des applications et le contexte d'exécution capturé par l'intergiciel sont relativement simples sur ces approches et, à l'exception de la proposition de Controlets [Donsez, 2006], les mécanismes de détection de conflits ne sont pas pris en compte par ces approches.

Tableau 5 - Vue synthétique des propositions de déploiement adaptatif.

Critères	OTA-PSD	CADComp	Controlets	OTA-3DMA
Identification automatique des DM	NON	NON	Identification de la plate-forme	NON
Plate-forme	J2ME MIDP	Java CDC	Java CDC	Java CDC/MIDP
Type d'application	Monolithique	Orientée composants	Orientée services	Orientée composants
Exigence d'installation d'un intergiciel	OUI (Un canevas au dessus de MIDP)	OUI (OpenCCM)	OUI (OSGi)	OUI (le canevas 3DMA)
Métadonnées de l'application	Décrites en XML	Décrites en XML	Informations sur le type d'équipement de contrôle en XML	
Restrictions	Décrites en XML	Décrites en XML en reliées à un modèle de contexte par objet	Décrites en XML	
Adaptation après l'installation	NON	NON	OUI	NON
Mise à jour	Par le biais de notifications	____	____	____
Mise en correspondance entre les restrictions et le contexte d'exécution	Programmer en Java	Programmer en Java	Programmer en Java	Programmer en Java
Détection de conflits de règles d'adaptation	____	____	Priorité	____
Technologie de Composants	____	Corba	OSGi	FarGo

PROPOSITION

PROBLEMATIQUE ET APPROCHE

Les mécanismes de gestion de documents multimédias personnels reposent fondamentalement sur l'association de métadonnées à ces documents. Les mécanismes d'organisation et d'indexation des outils de gestion sont les aspects les plus dépendants de ces informations qui sont, la plupart du temps, assurées par les utilisateurs eux-mêmes.

Dans l'état de l'art, nous avons exposé les avantages de concevoir des solutions de gestion de multimédias personnels autour de l'utilisation de dispositifs mobiles et de la découverte du *contexte de création* de ces documents. En fait, la génération automatique de ces métadonnées contextuelles peut réduire les inconvénients de l'annotation manuelle, car une grande partie des métadonnées souhaitées, ou, effectivement, ajoutées par les utilisateurs font référence au contexte de création [Naaman *et al.*, 2004]. Nous souhaitons à l'aide de nos propositions d'être en mesure de réduire ce besoin d'une annotation manuelle exhaustive pour la gestion de documents multimédias. Notre objectif est de transformer cette activité répétitive en une activité de validation d'annotations grâce à une grande quantité de métadonnées qui seront produites automatiquement.

Nous avons présenté certaines approches d'annotation et d'organisation de documents multimédias possédant des objectifs similaires. Cependant, chaque approche confine ces métadonnées à un ensemble réduit d'informations contextuelles, et l'extension de ces métadonnées est difficile en raison de l'absence d'un modèle de représentation de ces informations qui soit indépendant de l'application. De plus, l'apport sémantique des informations contextuelles n'est quasiment pas exploité par ces approches lors des phases d'indexation, de recherche et de partage de contenu. Dans la majorité de ces approches, les métadonnées sont utilisées uniquement pour l'organisation de documents multimédias.

En ce qui concerne les approches de partage sensible au contexte de contenu, les informations contextuelles sont très peu formalisées et le contexte de création est fréquemment réduit à des informations de niveau symbolique (telles que les coordonnées géographiques).

Il existe un réel besoin d'un modèle de représentation des métadonnées qui intègre les informations les plus utilisées par ces approches et qui, simultanément, augmente l'expressivité des annotations et élève leur niveau sémantique. L'usage d'un vocabulaire formel et enrichi des métadonnées, notamment celles liées au contexte de création, peut

améliorer les différents aspects de la gestion de documents multimédias. En effet cette représentation peut être exploitée, par exemple, pour la réduction des problèmes d'ambigüité et d'imprécision des requêtes des utilisateurs. L'apport sémantique de ces métadonnées ouvre de nouvelles perspectives pour l'organisation, la visualisation et le partage automatique de documents multimédias.

Un autre inconvénient des approches sensible au contexte de gestion de documents multimédias est la difficulté de faire interopérer ces annotations lorsque l'on transfert un document d'un outil à un autre. Bien que les approches utilisent comme paramètre d'entrée des métadonnées EXIF, elles stockent les métadonnées générées par un processus d'enrichissement à travers une représentation interne, car les vocabulaires existants n'offrent pas une description élargie du contexte de création des documents.

Nous tenons à souligner également que les propositions de gestion sensible au contexte de documents multimédias entremêlent fréquemment les étapes d'acquisition et d'exploitation de contexte au code métier des outils de gestion de documents. À l'instar des premiers systèmes d'information sensibles au contexte, ce choix de conception limite la réutilisation des mécanismes d'acquisition et d'exploitation de contexte, et rend plus difficile leur extensibilité.

La conception et le développement de canevas et d'intergiciels qui assurent ces étapes de la gestion de l'information contextuelle peuvent apporter plus de flexibilité. Nous entendons ici appliquer les mêmes principes de modélisation et de structuration de ces étapes qui sont fournis par les architectures de capture et d'enrichissement du contexte présentées dans l'état de l'art. Bien que conçues exclusivement pour l'adaptation de Systèmes d'Information, la façon dont ces architectures organisent la capture, représentent et infèrent des informations contextuelles, constitue une base de départ pour le développement d'une architecture destinée à la mise en œuvre de services sensibles au contexte de gestion de multimédias.

Un défi supplémentaire doit être relevé lors de la conception de cette architecture : l'hétérogénéité des dispositifs mobiles. La capture du contexte de création, la production de documents multimédias, l'exécution de services mobiles d'annotation et de partage sont les fondements de la gestion sensible au contexte de documents multimédias. Cependant, ces mécanismes sont les plus impactés par l'hétérogénéité des dispositifs mobiles, car ils doivent accéder à des bibliothèques et à des ressources qui existent sous diverses formes sur les plates-formes de développement d'applications mobiles. Ce point est fréquemment négligé par les architectures sensibles au contexte qui sont conçues pour une mise en œuvre sur des plates-formes très gourmandes en ressources, et dont les bibliothèques ne sont pas disponibles sur la majorité de dispositifs mobiles. Ceci empêche l'utilisation de ces architectures sur un grand nombre de modèles de dispositifs mobiles. Également, l'approche « tout-ou-rien » de ces architectures, oblige au déploiement de *toute* la plate-forme d'acquisition de contexte, compris lorsque l'application n'utilise qu'une partie des informations contextuelles. Ainsi, une architecture, destinée à la mise en œuvre de services sensibles au contexte de gestion de multimédias, doit permettre un déploiement graduel et adapté de leurs mécanismes de capture et d'enrichissement de contexte afin d'assurer sa portabilité et son exécution de manière optimisée en termes de consommation de ressources (mémoire, batterie, …).

Dans l'état de l'art, nous avons étudié des approches d'adaptation d'applications mobiles qui tentent de franchir l'obstacle de l'hétérogénéité des dispositifs mobiles et, en conséquence, peuvent être adoptées pour l'adaptation des services de gestion de documents multimédias et de l'architecture d'acquisition de contexte elle-même. Les approches de déploiement adaptatif d'applications orientées composants nous semblent devoir être les privilégier. La programmation orientée composant facilite l'introduction de mécanismes d'adaptation (tels que le paramétrage, le remplacement de composants) et peut être utilisée pour développer les applications mobiles de gestion de documents multimédias et l'architecture d'acquisition de contexte. L'adaptation lors du déploiement, spécialement dans

la phase d'installation, permet d'installer sur les dispositifs mobiles uniquement les composants qui fonctionnement sur ces dispositifs et, par conséquent, peut être utilisée pour éviter de déployer un intergiciel complet d'acquisition de contexte ou des modules incompatibles avec le dispositif de l'utilisateur. On peut reprocher aux propositions de déploiement adaptatif présentées dans l'état de l'art l'incapacité de reconnaitre automatiquement les caractéristiques des dispositifs mobiles et l'obligation d'installer un intergiciel qui n'est pas lui-même adaptable aux plates-formes de dispositifs mobiles.

En synthèse, les informations contextuelles peuvent améliorer la gestion de documents multimédias, néanmoins, la découverte à l'aide de dispositifs mobiles de ces informations (et la production de multimédias elle-même) doit prendre en compte la hétérogénéité et les restrictions inhérentes à ces dispositifs.

Dans cette thèse, nous souhaitons proposer des solutions de représentation, capture et d'enrichissement sémantique de métadonnées décrivant le contexte de création d'un document multimédia. Nous optons pour une gestion sensible au contexte de documents multimédias personnels qui repose sur l'usage d'ontologies pour la représentation des métadonnées capturées et générées par nos plates-formes. Nous exploitons ensuite ces métadonnées enrichies dans l'organisation, l'annotation, la recherche et le partage de documents multimédias personnels, en vue d'une évolution de ces mécanismes.

Nous cherchons également dans cette thèse à concevoir des solutions plus flexibles, concernant l'acquisition de contexte et la production de multimédias, réalisées sur des dispositifs mobiles. Nous souhaitons fournir des mécanismes qui facilitent le développement et le déploiement d'applications mobiles qui exploitent les informations contextuelles pour la gestion de multimédia. Ces solutions doivent être conçues de manière à garantir leur portabilité sur un grand nombre de modèles de dispositifs mobiles. Nous optons pour l'usage de la programmation orientée composant et d'un mécanisme de déploiement adaptatif afin de garantir la portabilité des applications multimédias, et de la plate-forme d'acquisition de contexte elle-même, sur une grande diversité de modèles de dispositifs mobiles, même si la portabilité sous-tend une réduction du nombre de services de l'application, y compris de la plate-forme d'acquisition de contexte, ou la dégradation de leurs fonctionnalités.

L'ensemble de nos solutions est intégré au sein de la plate-forme CoMMediA (de l'anglais *COntext-aware Mobile multiMEDIa Architecture*) qui sera présenté dans les prochains chapitres.

6 MODELES POUR LA REPRESENTATION DES METADONNEES DE DOCUMENTS MULTIMEDIAS PERSONNELS

Dans ce chapitre, nous présentons notre modèle de représentation de contexte et ses extensions pour la représentation de l'annotation de documents multimédias, et pour l'adaptation d'applications mobiles. Les différents stades d'évolution de ces modèles ont été rapportés dans [Viana *et al.*, 2007a] [Viana *et al.*, 2007b][Viana *et al.*, 2008a][Viana *et al.,*2008b][Viana *et al.,* 2009b]. Nous utilisons le langage OWL, standard Web de représentation d'ontologies, pour exprimer nos modèles. La sémantique offerte par la version OWL-DL assure un bon compromis entre le pouvoir d'expression du langage et son efficacité pour le raisonnement.

Nous proposons d'abord un modèle pour exprimer une notion élargie de contexte : le modèle *ContextTop*. Dans l'état de l'art, nous avons pu constater que les diverses notions de contexte et leurs modèles de représentation sont plutôt destinés à l'adaptation des Systèmes d'Information. Cependant, nous voulons être en mesure de caractériser à la fois le *contexte de création* d'un document multimédia personnel, et le *contexte d'utilisation* d'une application en vue d'une adaptation de ses services, car ces deux notions de contexte partagent des concepts communs et peuvent hériter d'une notion de contexte plus générique.

Nous étendons *ContextTop* pour la représentation d'un large ensemble de propriétés, décrivant le contexte de création des documents multimédias, et quelques aspects relatifs au contenu lui-même du document. Ce modèle, nommé *ContextMultimedia*, est un support pour stocker et organiser les métadonnées des documents multimédias personnels. Les instances de ce modèle peuvent être complétées : i) automatiquement, par le processus de capture et d'enrichissement de contexte ou ii) manuellement, à l'aide d'un outil d'annotation.

Nous présentons à la fin de ce chapitre une plate-forme pour l'acquisition et la dérivation de métadonnées relatives aux documents multimédias personnels. Cette plate-forme exploite le modèle *ContextMultimedia* pour la représentation de ces métadonnées, et l'utilise également pour l'intégration des services d'enrichissement de contexte. Nous détaillons la composition de services Web que nous utilisons pour ce processus d'enrichissement de

contexte. De plus, nous exposons une approche de raisonnement à base de règles SWRL, pour dériver des informations concernant le contexte social de création d'un document multimédia.

6.1 L'ontologie Context Top

6.1.1 Notion élargie de contexte

Les définitions de la notion de contexte dans l'informatique sensible au contexte, spécialement celles de Dey [Dey, 2001], Chaari *et al.* [Chaari *et al.*, 2005] et Kirsch [Kirsch, 2006], confinent le contexte à un ensemble d'éléments qui caractérisent l'interaction entre l'utilisateur et l'application, et qui peuvent affecter cette interaction. D'après ces définitions, seules les informations qui peuvent améliorer l'interaction utilisateur-application doivent appartenir au contexte. Ainsi, le concepteur d'un système sensible au contexte ne doit ni modéliser, ni acquérir des éléments qui ne sont pas utiles pour adapter cette interaction. Cependant, certaines informations concernant la situation de l'utilisateur, sans liaison directe avec l'interaction, peuvent être décisives pour décrire d'autres activités d'un système. Par exemple, la découverte de noms de personnes proches de l'utilisateur n'a pas d'influence, en principe, sur l'interaction entre l'usager et le logiciel de prise de photos de son téléphone mobile. Néanmoins, cette information peut jouer un rôle important dans l'annotation de la photo produite par ce logiciel, car cette information est utile pour indexer le document et facilite une recherche ultérieure.

Nous visons une notion de contexte qui soit en mesure de caractériser à la fois le *contexte d'utilisation* d'une application en vue d'une adaptation de ses services et le *contexte de création* d'un document multimédia personnel en vue d'un usage pour l'annotation. Ces deux notions de contexte partagent des concepts communs et peuvent hériter d'une notion de contexte plus générique. Cette nouvelle notion de contexte doit être pragmatique, afin d'aider la conception de systèmes sensibles au contexte, spécialement dans le guidage du processus d'acquisition d'informations contextuelles. Cette notion doit également souligner les aspects dynamique et évolutif du contexte et orienter la création d'un modèle pour sa représentation.

En adaptant la définition de Dey, nous proposons la définition suivante : « fait partie du *contexte toute information qui peut décrire la situation des entités (et leurs relations) impliquées dans une action qui est jugée importante par le système. Ces entités sont tous les concepts abstraits et les objets physiques présents dans la zone d'observation du système à un instant* t_n *d'observation*».

Le contexte englobe des aspects au-delà des paramètres classiques d'entrée d'un système sensible au contexte et que le système souhaite capturer. L'intérêt de tout système sensible au contexte est de caractériser la situation d'une entité ou d'une action réalisée par cette entité afin de réagir en conséquence (par exemple, en changeant de comportement, en produisant des métadonnées). Une entité pour nous, tout comme pour Dey, est à la fois un objet physique (l'utilisateur, son dispositif mobile, les personnes aux environs de l'utilisateur, etc.) et un concept moins tangible (la machine virtuelle du téléphone mobile, l'humeur de l'utilisateur, le système lui-même, etc.).

Le contexte inclurait tout élément capable de contribuer à la bonne identification et à la compréhension d'une action réalisée par une de ces entités (telle que l'action d'accéder à un service réalisée par un agent logiciel externe, la prise d'une photo, ou le clic sur l'interface graphique exécuté par un utilisateur). En d'autres termes, le contexte dans lequel une action se produit donne du sens à cette action et peut aider le système à mieux répondre à cette action. Le contexte peut être spécialisé selon l'action qu'il décrit : le *contexte d'utilisation* de

l'application, le *contexte de création* du document multimédia, le *contexte d'exécution* d'une application, etc.

Nous tenons à souligner qu'en général les systèmes sensibles au contexte ne capturent forcément qu'une partie du contexte, car les systèmes d'acquisition ne sont pas capables de déterminer, avec plus ou moins de précision, toutes les entités et toutes les caractéristiques dont le système a besoin [Kirsch *et al.*, 2005].

Nous avons ainsi introduit dans notre définition l'idée de zone d'observation et de zone d'intérêt. La **zone d'observation** définit l'espace physique couvert par les capteurs du système et ces sources d'information. Cette zone varie selon la portée du système (le dispositif, une salle de réunion, toute une ville, etc.). La **zone d'intérêt** comprend les entités que le système juge importantes pour définir la situation d'un autre élément que lui-même (tel que l'utilisateur, le dispositif) ou une action produite par cet élément.

La Figure 48 illustre notre définition. Les éléments appartenant au contexte sont tous ceux qui se trouvent à l'intérieur de l'intersection de la zone d'intérêt d'un système (les entités et les relations qu'il **voudrait** caractériser) et de sa zone d'observation (les entités et les relations qu'il **peut** observer).

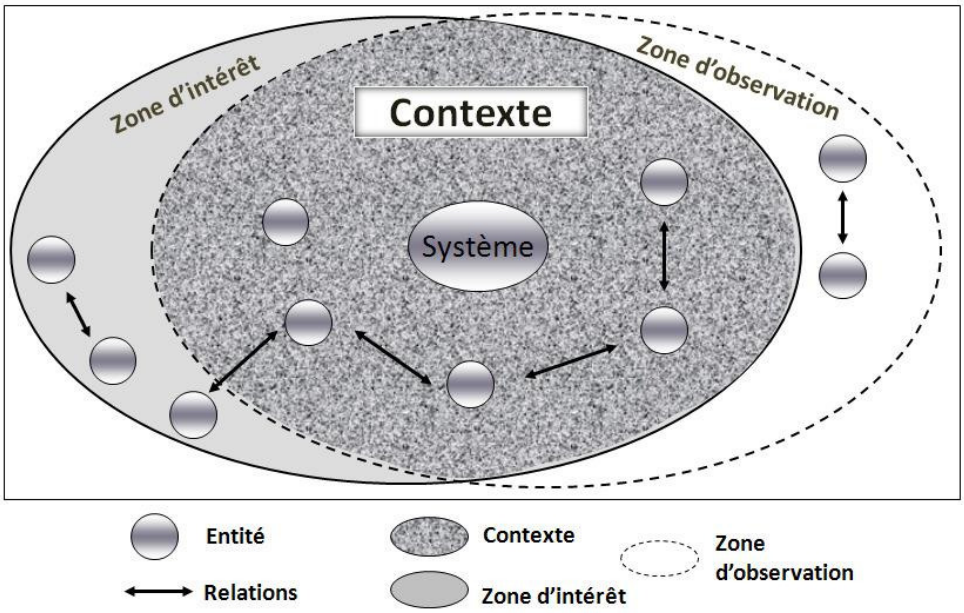


Figure 48 - Contexte défini comme étant l'intersection de la zone d'intéret du système et de sa zone d'observation.

Chaque système faisant usage de la notion de contexte doit déterminer les éléments qui composent la zone d'intérêt à partir de ses objectifs. En conséquence, la zone d'intérêt d'un système peut varier d'un ensemble simple et réduit d'informations (le modèle du dispositif d'accès au système) à une composition d'éléments plus riche sémantiquement et plus complexe à capturer/inférer (telle que la relation d'amitié qui lie l'utilisateur du système et une personne à proximité). Nous tenons à souligner que la description de l'état du système peut également participer de sa zone d'intérêt.

Le contexte acquis par ces systèmes sera composé par le sous-ensemble des éléments de la zone d'intérêt qu'il peut effectivement caractériser.

Nous avons également ajouté à la définition la notion d'instant d'observation (t_n) car le contexte reflète l'état temporel des entités observées qui peuvent changer et évoluer au fil du temps. De plus, l'ensemble des entités qui composent le contexte peut lui-aussi être modifié. En fait, le changement contextuel est perçu de trois manières :

- **L'altération des propriétés des entités observées**. Les états des entités surveillées par le système (et le système lui-même) changent tout au long de son exécution. Prenons l'exemple de la localisation de l'utilisateur qui peut changer au cours du temps d'observation. Ceci exige une acquisition de contexte continue afin de répondre à des éventuels changements.
- **La modification de la zone d'observation**. La zone d'observation peut s'élargir par l'incorporation de nouveaux capteurs et de nouvelles sources d'informations. De même que, la défaillance d'un capteur ou la déconnexion d'une source de données réduit l'ensemble des entités passibles d'être observées par le système. Par exemple, la défaillance du capteur GPS amène l'impossibilité d'acquérir la localisation de l'utilisateur.
- **Le changement de la zone d'intérêt**. L'ensemble d'entités que le système a besoin de caractériser peut changer en fonction de la phase du cycle de vie du système et, également, selon ses fonctionnalités en cours d'exécution. Par exemple, lors de l'installation d'un système mobile d'annotation multimédia, le système a plutôt besoin de connaître les caractéristiques du dispositif mobile afin d'adapter ses fonctionnalités. Cependant, lors de l'exécution de son service d'annotation, la localisation du dispositif mobile est l'information la plus importante à connaître.

Un aperçu du processus de caractérisation du contexte tel que nous le considérons, et qui repose sur notre définition, est illustrée dans la Figure 49.

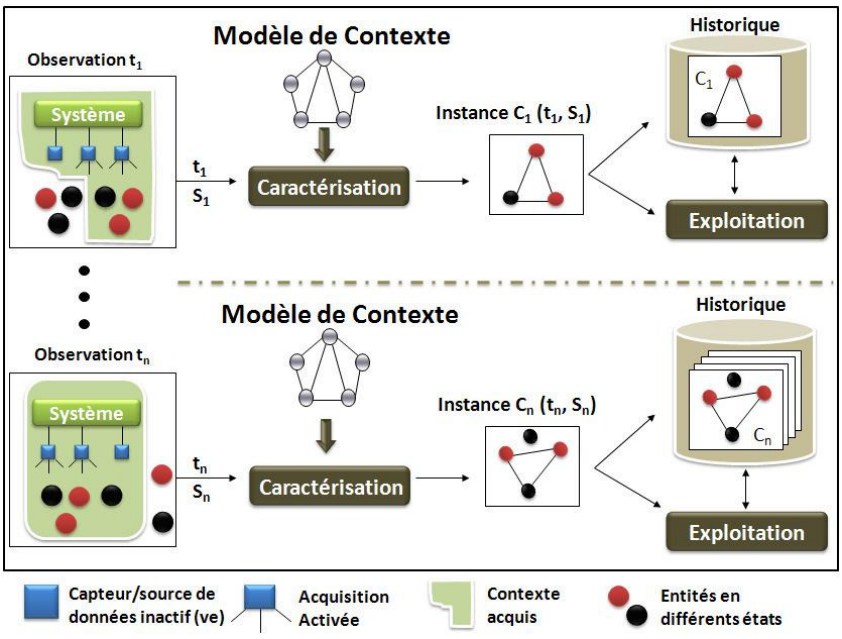


Figure 49 - Processus de caractérisation du contexte.

L'acquisition de contexte est généralement un processus continu qui est mis en œuvre avec une fréquence d'observation variant selon le système. Chaque observation englobe les états des entités à un instant d'observation donné. Par exemple, dans la Figure 49, le contexte est composé de quatre entités (le système et trois autres entités) lors de l'instant t_1.

Le système décrit les propriétés des entités observées à l'aide d'un modèle de contexte. Ce modèle doit être sémantiquement riche et extensible pour permettre une représentation en machine des propriétés acquises par l'observation. Au départ, le résultat de ce processus de caractérisation est une instance du modèle qui consolide en machine l'état des entités perçues à l'intérieur de la zone d'observation S_1 lors de l'instant t_1. Ces informations contextuelles

peuvent ensuite être stockées afin de garder une trace de leur évolution, ou être directement exploitées par le système (par exemple, pour adapter son comportement).

Dans la Figure 49, le contexte acquis lors de l'instant t_n illustre le changement de contexte tel que nous l'avons présenté auparavant. Au delà de la modification des valeurs des propriétés des entités (illustrée par un code de couleurs), l'ensemble lui-même des entités observées a changé (le contexte C_n est composé du système et de quatre autres entités).

Notre notion de contexte met ainsi en évidence les aspects spatiaux et temporels des informations contextuelles qui doivent être pris en compte lors des processus d'acquisition et d'exploitation de ces informations. Nous choisissons également de bien différencier l'ensemble d'entités susceptibles d'être observées du regroupement d'éléments qui composent effectivement le contexte. En fait, l'évolution de la zone d'intérêt d'un système peut guider l'adaptation de l'acquisition du contexte elle-même. Un système sensible au contexte peut désactiver/activer les capteurs selon les besoins de ces services en exécution, en réduisant ainsi la consommation des ressources du dispositif hôte du système. Dans la suite de ce chapitre, nous détaillons le modèle de contexte proposé qui suit notre définition.

6.1.2 ContextTop

Nous proposons un modèle de contexte générique qui supporte le processus de caractérisation illustré par la Figure 49. L'objectif du modèle est de stocker les principales propriétés des entités appartenant au contexte d'une action dont la description est utile au système. La généricité, visée par le modèle, nous conduit à une description de contexte qui n'introduit pas de concepts prédéfinis pour les entités. Nous offrons plutôt des catégories (ou dimensions) qui guident le concepteur dans cette tâche d'énumération des composantes du contexte d'une action.

La Figure 50 présente une illustration graphique de notre modèle : l'ontologie *Context Top*. Une action (concept *Action*) possède un contexte (*Context*) qui est composé de plusieurs éléments contextuels (*Context Element*). Le contexte peut également décrire la situation de ses sous-éléments (propriété *describeTheSituationOf)*. La propriété *hasContext* est la propriété inverse de *describeTheSituationOf.*

Nous avons choisi de segmenter le contexte en cinq dimensions : sociale, spatiale, temporelle, spatio-temporelle et informationnelle. Cette segmentation est représentée par la spécialisation du concept *Context Element*. La **dimension sociale** fait référence aux aspects liés aux personnes présentes à l'intérieur de la zone d'observation (par exemple, l'utilisateur du système, son profil, son réseau social). La **dimension spatiale** exprime des informations concernant le lieu où l'action s'est déroulée (par exemple, les coordonnées géographiques, l'adresse postale, etc., les objets fixes à proximité). La **dimension temporelle** inclut des informations liées à l'instant d'observation (par exemple, la date, le jour de la semaine, etc.). La **dimension spatio-temporelle** est composée d'entités qui nécessitent des informations spatiales et temporelles pour être acquises (par exemple, la météo, les objets mobiles à proximité). La **dimension informationnelle** englobe les entités matérielles et logicielles intégrées au système (le dispositif mobile, le système lui-même, les capteurs, etc.). Le choix des éléments qui composent ces dimensions revient au concepteur du système sensible au contexte.

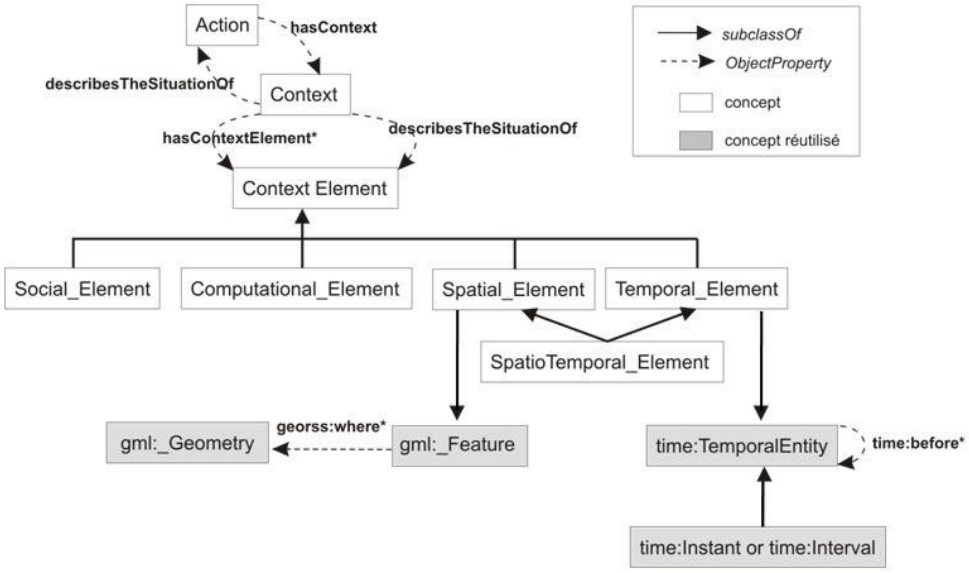


Figure 50 - Illustration graphique de notre ontologie Context Top.

Au lieu de proposer de nouvelles propriétés pour la représentation des éléments spatiaux et temporels, nous avons opté pour la réutilisation et l'extension d'ontologies existantes. En fait, un des avantages de l'utilisation des ontologies provient de l'intégration et de l'interconnexion de plusieurs vocabulaires [Shadbolt *et al.*, 2006]. Lors d'une utilisation de notre modèle pour la description de métadonnées ou pour l'échange d'informations contextuelles, un autre système pourra « comprendre » une partie de concepts décrits grâce à l'approche de réutilisation. Il pourra découvrir de nouveaux concepts et relations, même s'il ne comprend que la sémantique des ontologies réutilisées.

Nous avons utilisé une représentation OWL-DL du vocabulaire GeoRSS[80] pour la représentation des entités spatiales. Initialement, GeoRSS était un vocabulaire destiné à inclure les coordonnées géographiques dans des flux RSS (RSS est un format XML utilisé pour décrire les flux de contenu). Aujourd'hui, GeoRSS est utilisé également pour la dissémination des métadonnées géographiques sur plusieurs types de documents Web afin de promouvoir l'interactivité entre des applications hétérogènes. La version de GeoRSS GML fournit une série de concepts et relations pour la description de entités géographiques (de l'anglais, *feature*) comprenant les systèmes de référence de coordonnées, la géométrie, la topologie et les unités de mesure. GeoRSS GML fournit une description plus simple que le langage de balises géographique GML (*Geographic Mark-up Language*) proposé par l'OGC (*Open Geospatial Consortium*[81]). Cependant, son expressivité est suffisante pour le type d'usage que nous envisageons pour la dimension spatiale de notre modèle de contexte.

Nous profitons de l'héritage multiple offert par le langage OWL pour la représentation de concepts de la dimension spatiale. La Figure 50 montre la relation entre l'élément spatial (*Spatial Element*) et les concepts de GeoRSS (les concepts *gml:Geometry* et *gml:Feature*). Un concepteur peut ainsi lier de manière standard une géométrie à un élément spatial.

Une approche similaire a été choisie pour la représentation d'informations temporelles concernant le contexte d'une action. Nous avons utilisé des concepts de l'ontologie OWL-Time[82]. Cette ontologie, proposée par le W3C, permet la description de propriétés temporelles

[80] http://fr.wikipedia.org/wiki/GeoRSS

[81] http://www.opengeospatial.org/

[82] http://www.w3.org/TR/2006/WD-owl-time-20060927/

des services Web et de métadonnées temporelles associées à des documents Web. OWL-Time fournit un vocabulaire pour exprimer des relations d'ordre entre des instants et des intervalles, ainsi que des informations sur leurs durées, et des propriétés concernant la date et l'heure. Le concept *Temporal Element* de notre ontologie hérite les propriétés et les relations du concept *Temporal Entity* de l'ontologie OWL-Time.

La Figure 51 illustre un exemple d'extension de l'ontologie *Context Top* pour la caractérisation du contexte selon la définition de Dey. Le contexte décrit l'interaction entre l'utilisateur (*Windson*) et une application (*ContextToolkit*). Les concepts noircis sont ceux proposés par le concepteur pour une appartenance à la zone d'intérêt du système : *User, Place, Human Action, System* et *Mobile Device*. Un exemple d'observation du contexte, C_1, est également présenté dans la Figure 51. Nous pouvons interpréter cette instanciation du modèle comme étant: Le contexte C_1, observé lors de l'instant t_1 (27/09/2009 10:44:46), décrit l'interaction entre l'utilisateur *Windson* et le système (*ContextToolkit*). Le contexte est composé de l'utilisateur *Windson,* localisé sur un lieu (*Place*) nommé *Home.* Cet endroit est décrit par un polygone (la géométrie *Home Geometry*). *Windson* utilise le dispositif *Nokia N95* pour interagir avec le système..

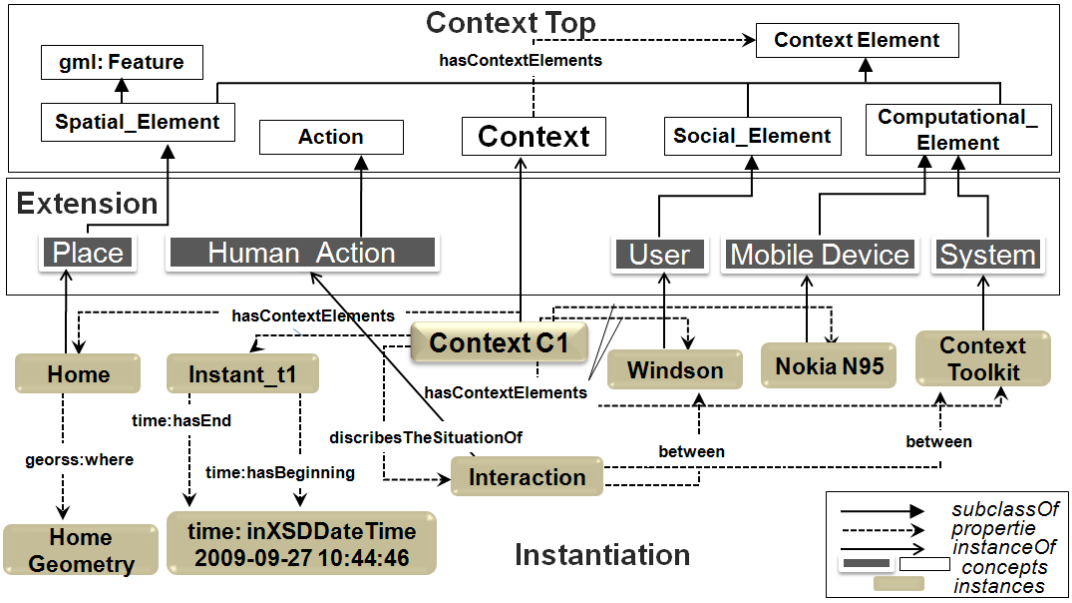


Figure 51 - Extension et instanciation de l'ontologie Context Top.

6.2 L'ontologie ContextMultimedia

Dans cette section, nous présentons une extension de l'ontologie Context Top pour la représentation d'annotations d'un document multimédia personnel. Nous visons le stockage d'un ensemble de métadonnées liées, principalement, au contexte de création de ces documents. Dans l'état de l'art, nous avons recensé quelques approches destinées à la représentation d'annotations de documents multimédias (telles que MPEG7, EXIF, XMP). Ces approches proposent des vocabulaires destinés à la description du contenu d'un document, et leurs attributs contextuels sont très limités. Nous voulons avec *ContextMultimedia* couvrir la majorité des informations contextuelles exploitées par les approches d'annotation sensibles au contexte. Nous souhaitons ainsi offrir une représentation suffisamment riche qui facilite l'utilisation, le partage et l'accroissement des métadonnées associées aux documents multimédias personnels.

Nous discutons en premier lieu des éléments les plus pertinents pour la structuration de ces métadonnées. Notre proposition repose sur des études de comportement des utilisateurs lors

de l'annotation manuelle et de la recherche de photos personnelles. Ces études réalisées par les auteurs Naaman *et al.* [Naaman *et al.*, 2004] et Matellanes *et al.* [Matellanes *et al.,* 2005] sont présentées dans le chapitre 3. Nous les appliquons pour l'annotation de documents multimédias en général (photos, vidéos, audio et texte) car les aspects contextuels sont pratiquement les mêmes sur ces documents. Nous présentons dans la suite les concepts et les relations de l'ontologie *ContextMultimedia*. Nous optons par le même principe de réutilisation d'autres ontologies appliqué lors de la conception de l'ontologie Context Top. L'intégration des ontologies réutilisées (Context Top, NEXIF, MPEG7 OWL, FOAF, Dublin Core) et un exemple d'annotation d'une photo à l'aide de *ContextMultimedia,* sont également présentés dans cette section.

6.2.1 Quels concepts sont les plus pertinents pour l'annotation ?

Les objectifs principaux du processus d'annotation d'un document multimédia personnel sont : i) servir de point de départ pour l'indexation de ces documents et, par conséquent, faciliter les processus futurs de recherche et de navigation au sein d'une collection de documents; et ii) aider au souvenir du contenu d'un document et de son contexte de création. Un modèle d'annotation doit offrir ainsi des concepts et des relations permettant ces deux usages.

Concernant le contenu des documents multimédias personnels, les vocabulaires proposés par les technologies Dublin Core, MPEG 7 et VDO couvrent la majorité des usages de ce type d'annotation au sein des outils de gestion de multimédias personnels. Au delà des aspects de niveau signal, l'annotation du contenu concerne principalement l'identification d'objets et des personnes sur les images et sur les vidéos. Ces métadonnées peuvent être représentées par une combinaison des technologies susmentionnées. Cette combinaison doit également inclure l'association de mots-clés et de descriptions textuelles à l'annotation.

Concernant le contexte de création d'un document multimédia personnel, les concepts les plus exploités pour l'organisation et pour l'annotation des multimédias personnels sont : i) la notion d'événement, et ii) des aspects relatifs aux cinq dimensions de contexte de l'ontologie *Context Top*. Cependant, comme nous avons pu le constater dans l'état de l'art, il n'existe pas, à notre connaissance, de modèle pour représenter ces concepts.

La **notion d'événement** concerne généralement un ensemble de documents créés pendant la réalisation d'un fait marquant (un voyage, une fête, un diner, etc.). La notion d'événement constitue un lien entre les documents d'une collection. Par conséquent, elle peut être exploitée pour la répartition des documents en regroupements qui représentent chacun un événement distinct. La notion d'événement possède également une forte association à un endroit et à une durée. Sa représentation exige ainsi la prise en compte de son caractère spatio-temporel.

Les autres aspects contextuels font référence à la situation de l'utilisateur lors de la création d'un unique document. Ils répondent des questions du genre : « Où » et « Quand » ai-je pris cette photo ? « Qui » était avec moi ce jour là ? La Figure 52 illustre la répartition de ces aspects selon les cinq dimensions de contexte que nous proposons. Nous décrivons à présent nos choix de description pour chacune de ces dimensions.

Les éléments appartenant au **contexte spatial** de création d'un document sont les informations les plus utiles pour le souvenir et pour la recherche efficace d'un document multimédia personnel [Naaman et *al.*, 2004]. Cependant, la notion de lieu ou d'endroit de création recouvre plusieurs significations [Christensen, 2001].

Le sens commun définit un endroit comme étant un espace géographique avec un contour défini, et qui appartient à une organisation hiérarchique (par exemple, le lieu Champs Élysées appartient à la hiérarchie Champs Élysées, Paris, France, Europe). Ce sens commun est dérivé de l'organisation administrative et politique de l'espace à laquelle nous sommes familiarisés

grâce à la notion d'adresse géographique. La notion de lieu peut également porter une sémantique limitée à un groupe de quelques personnes ou à une communauté. Cette notion est basée sur des expériences partagées par ces personnes (par exemple, le lieu Champs Élysées peut être décrit comme étant « notre avenue préférée »). Souvent des notions de relations spatiales qualitatives apparaissent entre divers concepts géographiques. Par exemple, on peut trouver des relations de distance relatives à des points d'intérêt (« près de l'Arc de Triomphe »), des relations d'orientation (« au nord-ouest de Paris ») et des relations topologiques (« à l'intérieur de la place de l'Etoile »).

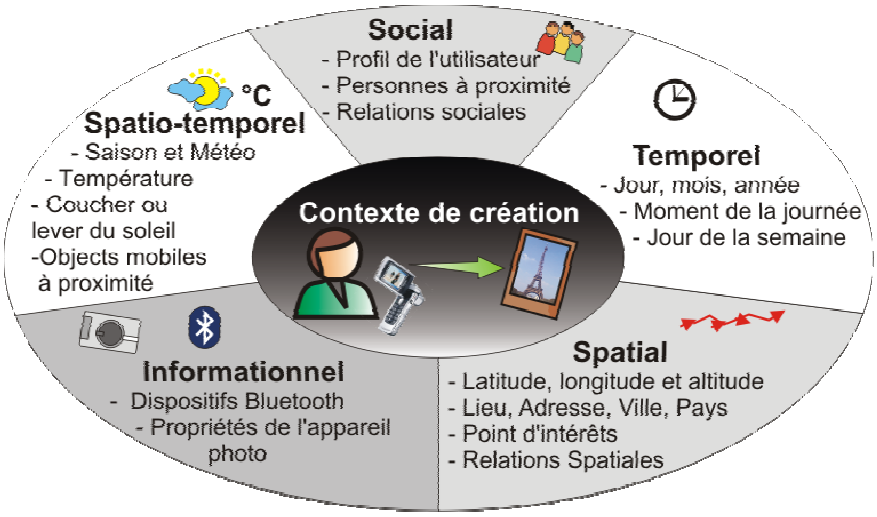


Figure 52 - Aspects contextuels couvert par l'annotation de multimédias personnels.

Tous ces divers sens de la notion de lieu peuvent être exploités pour l'annotation de documents multimédias personnels car ces informations sont fréquemment utilisées par les utilisateurs lors de l'annotation manuelle de leurs documents. Dans la dimension spatiale de notre modèle d'annotation, nous offrons plusieurs formes pour l'expression des informations spatiales associées à un document multimédia : des coordonnées géographiques (latitude, longitude, altitude, système de coordonnées), les relations spatiales dérivées (par exemple, « près de », « au nord de ») et des définitions personnelles de lieu (« ma maison », « mon restaurant préféré »). Il est important de noter que la notion de lieu de création peut faire référence à la fois à la position de la scène enregistrée et à la localisation de l'utilisateur lors de la création du document, car ces deux endroits peuvent être distants. Par conséquent, nous ajoutons à notre modèle la possibilité d'exprimer ces deux localisations.

Le **contexte temporel** de création d'un document est un autre aspect important à prendre en compte dans l'organisation et lors de la recherche des documents multimédias personnels. Nous avons vu dans l'état l'art qu'une date précise n'est pas l'attribut temporel le plus utilisé lorsqu'une personne tente de trouver un document d'une manière chronologique. Lorsqu'un document n'a pas été créé à une date facilement identifiable (par exemple, un jour d'anniversaire, une date commémorative), les personnes emploient plutôt le mois, le jour de la semaine, le moment de la journée (le matin, l'après-midi, la nuit, etc.), et/ou l'année pour formuler leur requête. Ainsi, les éléments temporels de notre modèle permettent l'association d'un instant (date et heure) et d'un intervalle de création aux documents aussi bien que les diverses interprétations et attributs temporels énumérés ci-dessus (Figure 52).

Au-delà des aspects spatiaux et temporels, les divers travaux de gestion de multimédia sensible au contexte (décrits dans le Chapitre 3) ont montré l'intérêt de découvrir d'autres informations contextuelles concernant la création d'un document : la saison, la période de la journée par rapport au coucher/au lever du soleil et les conditions météorologiques (la température, etc.). Par exemple, identifier si une image a été prise le soir (après le coucher du

soleil) ou pendant la journée (après le lever du soleil et avant le coucher du soleil) a été mentionné par les utilisateurs comme étant aussi important que l'information du mois lors duquel une photo a été prise [Naaman *et al.*, 2004]. La **dimension spatio-temporelle** de notre modèle comporte la description de ces informations contextuelles. En fait, ces métadonnées sont fréquemment calculées à partir d'informations spatiales et temporelles. Pour découvrir la saison, par exemple, le système a besoin au moins de la date et de l'identification de l'hémisphère où l'utilisateur se trouvait lors de la création de son document multimédia. Nous ajoutons également, à la dimension spatio-temporelle, la description d'objets mobiles (c'est-à-dire, des objets de position non fixe) qui ont été détectés à proximité de l'utilisateur lors de création d'un document. Cette information peut s'avérer utile lors du processus d'enrichissement du contexte de ce document.

Une autre information significative pour se rappeler d'un document multimédia personnel est la caractérisation des personnes présentes à proximité lors de la création de ce contenu. Ces personnes peuvent avoir participé à la scène enregistrée (une voix dans un message audio ou dans un vidéo) ou être même le sujet principal du multimédia (le focus d'une photo). Le **contexte social** de notre modèle offre des concepts pour la description de ces personnes (leur noms, leur profil, leur réseau de connaissances, etc.). Cette information peut être utilisée pour l'annotation et pour le partage d'un document multimédia. Au delà de la simple identification des personnes se trouvant à proximité, notre modèle permet d'exprimer les relations sociales existantes entre ces personnes et le créateur du document multimédia. Nous considérons le propriétaire du dispositif d'enregistrement du multimédia comme étant le créateur du document multimédia. Le contexte social décrit également des métadonnées liées au créateur du document (son nom, son profil Web, etc.).

L'identification et la configuration de l'appareil utilisé pour la création du document sont les métadonnées plus présentes pour les approches EXIF et XMP. Malgré l'intérêt mineur porté par la plupart des utilisateurs à l'usage de ces métadonnées pour l'annotation de leurs documents multimédias, nous tenons à les inclure dans la **dimension informationnelle** de notre modèle, car ces métadonnées offrent un aperçu statistique des appareils et des configurations les plus expérimentées par les utilisateurs. Afin de représenter ces informations, nous incluons dans notre modèle les principaux attributs d'EXIF. Le contexte informationnel de notre modèle est également composé de la description des dispositifs informationnels détectés dans la zone d'observation (par exemple, RFID, dispositifs Bluetooth) pendant la création de documents. Ces informations peuvent être utiles pour l'enrichissement du contexte, spécialement pour la découverte du contexte social d'un document multimédia.

La Figure 53 illustre un exemple des métadonnées que nous envisageons de stocker à l'aide de notre modèle d'annotation.

La Figure 53 présente un ensemble de métadonnées relatives à une photo prise à l'intérieur du Stade de France dans la banlieue de Paris, en France. Nous voulons représenter, par exemple, qu'une personne (Karol) a était détectée à proximité lors de la prise de cette photo et qu'elle appartient au réseau social du créateur de ce document (Windson).

Nous présentons, dans la suite, les concepts et les relations de notre modèle d'annotation et des mécanismes pour capturer et dériver la plupart des métadonnées illustrées dans la Figure 53.

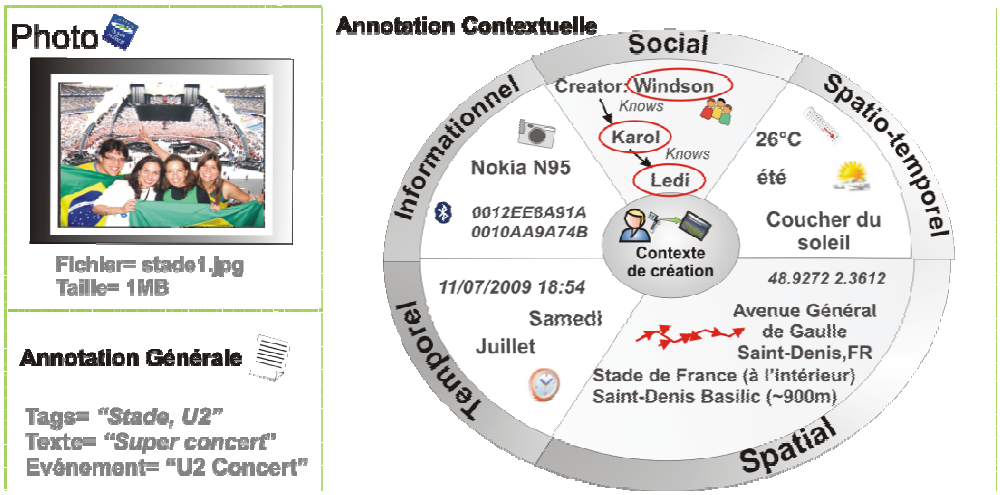


Figure 53 - Exemple d'annotation d'une photo.

6.2.2 Ontologie ContextMultimedia

Nous proposons un modèle qui supporte la description du contenu et du contexte de création d'un document multimédia personnel. Une vue générique du modèle *ContextMultimedia* est présentée dans la Figure 54. *ContextMultimedia* représente les aspects jugés pertinents pour l'annotation de documents multimédias personnels, tel que décrit dans la section précédente.

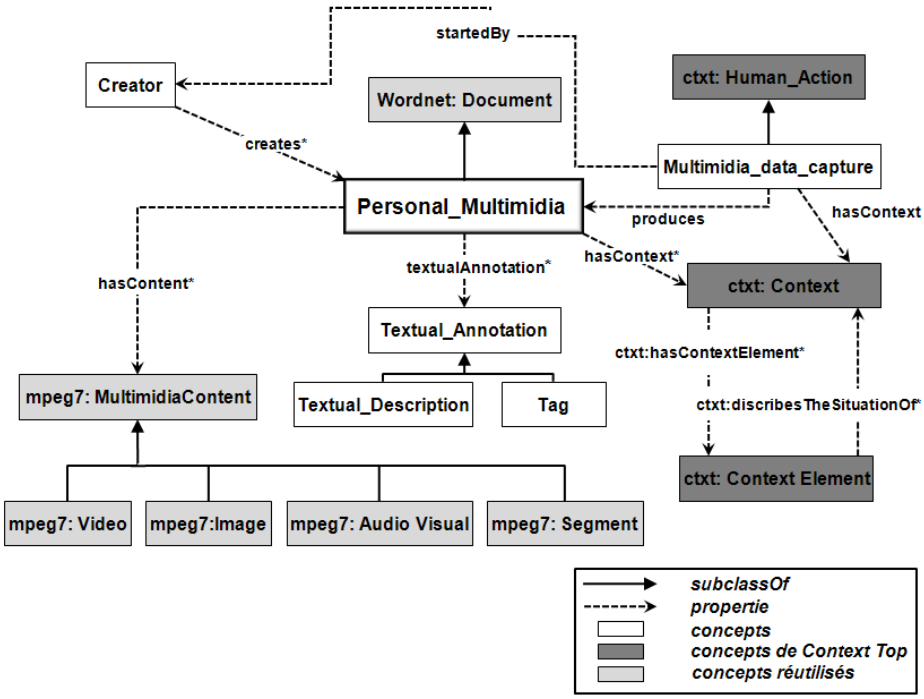


Figure 54 - Squelette des concepts de l'ontologie ContextMultimedia.

Le concept central de notre modèle est le document multimédia personnel (*Personal_Multimedia*). Nous le représentons comme étant un sous-concept de *wordnet:Document* décrit par la base de données lexique Wordnet, qui permet ainsi de lier notre ontologie à un thesaurus d'acceptation plus large. *Personal_Multimedia* possède une liste d'attributs provenant du vocabulaire Dublin Core pour une description simplifiée du document. Le Tableau 1 présente ces propriétés et la sémantique qui nous leur associons.

Nous introduisons également dans le modèle trois manières d'associer des métadonnées plus complexes à un document multimédia personnel : i) métadonnées spécifiques au contenu (*mpeg7:MultimediaContent*), ii) métadonnées concernant le contexte de création du document (*ctxt:Context*), et iii) des annotations textuelles (*Textual_Annotation*).

L'annotation du contenu exploite une représentation en OWL du langage MPEG7[83]. Celle-ci offre un vocabulaire pour la description des propriétés selon le type de document multimédia (vidéo, image, etc.), et de ses attributs de niveau signal (par exemple, le concept *mpeg7:EdgeHistogram* décrit la texture d'une image ou d'un segment de vidéo). La représentation ontologique de MPEG7 permet la description de plusieurs segments d'un même document multimédia (par exemple, le concept *mpeg7:StillRegion* exprime une région d'une image unique ou d'un fichier vidéo). La segmentation peut être utilisée pour décrire les régions d'une image occupée par des visages de personnes. L'objectif de l'introduction de cette ontologie dans notre modèle est d'offrir aux concepteurs de systèmes de gestion de multimédias une manière standard pour représenter le contenu d'un document personnel.

Nom de la propriété	Description
dc:creator	Le nom du propriétaire du dispositif de création du document multimédia
dc:identifier	Le nom du fichier du document multimédia personnel
dc:type	Le type de multimédia : {*image, video, text, sound*}
dc:format	L'encodage du document :{*mpeg, jpeg, mp3, etc.*}
dc:subject	Des mots-clés associés au document, séparés par des virgules
dc:description	La description textuelle du document
dc:date	La date de création

Tableau 6 – Propriétés de Dublin Core associées au concept Personal_Multimedia.

Un concepteur peut déjà exploiter les propriétés *dc:subject* et *dc:description* de *Personal_Multimedia* pour l'association de mots-clés et d'une description textuelle au document multimédia. Nous ajoutons les sous-concepts de *Textual_Annotation* : le concept *Tag* permet l'ajout de mots-clés au document, et le concept *Textual_Description* offre l'association d'une description en langage naturel. La différence entre ces concepts et les attributs hérités du vocabulaire de Dublin Core est la possibilité d'indiquer la provenance de la description (manuelle, automatique) et si une validation de ces métadonnées par l'utilisateur a eu lieu (propriété *isValidate*).

La description de métadonnées contextuelles du document est introduite grâce à une relation (*produces*) entre l'action de production d'un contenu (*Multimedia_Data_Capture*) et le document (*Personal_Multimedia*). En fait, *Multimedia_Data_Capture* est un sous-concept de la classe *ctxt :Action* appartenant à notre ontologie *ContextTop* qui permette ainsi l'association d'un contexte à cette action. Le contexte décrivant l'action de création d'un document est associé également au concept *Personal_Multimedia* par le biais de la relation *hasContext*. L'intérêt principal de notre modèle est de décrire le contexte de création d'un document. Cependant, nous ne limitons pas les métadonnées contextuelles d'un document multimédia à la description de l'action de création, car un concepteur peut souhaiter associer une liste de contextes à un document (la relation *hasContext* possède ainsi une multiplicité 1..*). Par exemple, pour une vidéo, le modèle peut décrire deux groupes de métadonnées concernant le contexte de l'utilisateur : un groupe décrivant le contexte au début de l'enregistrement et un autre concernant le contexte lorsque l'enregistrement touche à sa fin.

Nous décrivons à présent les concepts appartenant à chaque dimension contextuelle.

La dimension spatiale du contexte est représentée par l'ontologie ContextMultimedia en utilisant les concepts illustrés dans la Figure 55. Afin de modéliser les divers sens que la

[83] http://metadata.net/mpeg7/

notion de lieu peut prendre pour l'annotation d'un document multimédia, nous introduisons deux sous-concepts de *Spatial_Element* : *Place* et *GeoreferencedObject*. Le concept *Place* définit le lieu où le document multimédia a été créé. Nous pouvons ajouter une description et une géométrie à ce lieu grâce aux propriétés héritées du concept *gml:Feature* de l'ontologie GeoRSS. Une description peut indiquer le nom attribué par l'utilisateur à ce lieu (par exemple, « chez moi »). La géométrie associée au concept *Place* varie d'un simple *gml:Point* doté de coordonnées géographiques (latitude, longitude et altitude) à un polygone plus complexe qui décrit une région. Une adresse géographique (*Address*) peut être également liée au concept *Place*. L'adresse possède des attributs décrivant le numéro, la rue, la ville, le département, la région, le pays et le continent de du lieu où le document multimédia a été créé. Les propriétés de l'adresse suivent le découpage administratif proposé par Geonames[84].

Le concept *GeoReferencedObject* stocke des informations sur des points d'intérêt statiques que le système d'annotation juge pertinents de mettre en relation avec le document multimédia personnel. Dans l'exemple d'annotation d'une photo illustré par la Figure 53, le Stade de France et la Basilique de Saint Denis sont les deux points d'intérêt choisis par le système pour prend part au contexte spatial de la photo d'exemple.

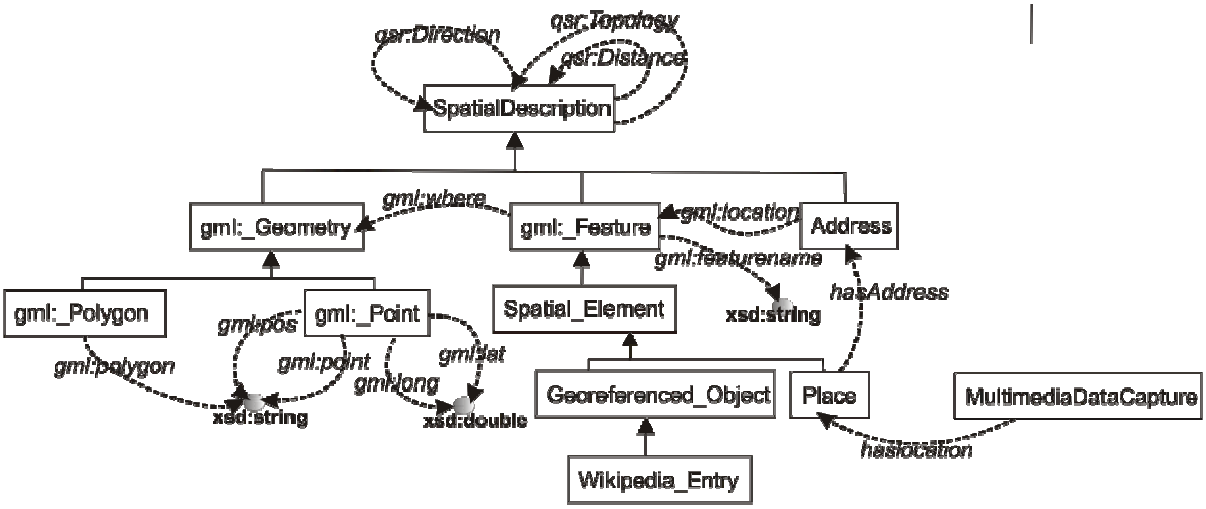


Figure 55 - Éléments de la dimension spatiale de ContextMultimedia.

Afin de représenter les **relations spatiales qualitatives** entre les points d'intérêt et le lieu de création d'un document, nous utilisons l'ontologie *Qualitative Spatial Relations* proposée par Miron Alina [Miron *et al.*, 2007], un membre de notre équipe de recherche. Cette ontologie, représentée dans la Figure 55 par le concept *Spatial_Description*, modélise trois types génériques de relations spatiales : des relations de distance (*qsr:distance*), des relations de direction (*qsr:direction*) et des relations topologiques (*qsr:topology*). Les **relations topologiques** sont de deux types : à l'intérieur (*insideOf*) et à l'extérieur (*outsideOf*). Ces métadonnées définissent la position relative du lieu de création du document multimédia par rapport un point d'intérêt dont la géométrie est décrite par un polygone. Les **relations de direction** change selon la représentation géométrique disponible pour décrire les points d'intérêt et le lieu de création du document multimédia. Si ces objets spatiaux sont décrits par des points, huit relations de direction sont possibles : *NorthFrom, SouthFrom, EastFrom, WestFrom, NorthEastFrom, NorthWestFrom, SouthEastFrom, SouthWestFrom.* Si au moins un de ces objets possèdent un polygone décrivant sa géométrie et la relation topologique

[84] http://www.geonames.org/

insideOf est valide, huit autre relations sont disponibles pour l'annotation : *NorthOf, SouthOf, EastOf, WestOf, NorthEastOf, NorthWestOf, SouthEastOf, SouthWestOf, CenterOf*.

La Figure 56a illustre un exemple d'utilisation de ces relations spatiales. Le lieu de création des documents multimédias (point *mdc*) est représenté par des coordonnées géographiques et un point d'intérêt, le Stade de France, est représenté par un polygone. Avec ces informations, un outil d'annotation peut inférer que les documents multimédias ont été créés à l'intérieur du Stade de France (relation topologique *insideOf (mdc, Stade de France)* et plus spécifiquement dans la partie Est du stade (relation de direction *EastOf (mdc, Stade de France)*).

Les **relations de distance** sont plus complexes de représenter en raison des attributs concernant les méthodes de calcul utilisées (distances euclidiennes, distances en transport public, etc.) et l'échelle. Pour simplifier cette représentation, beaucoup plus complexe sur l'ontologie de départ, nous considérons que les points d'intérêt et le lieu de création des documents sont décrits dans la même échelle et le même système de coordonnées. Nous adoptons également la distance euclidienne pour le calcul da distance, car les données exigées par les autres méthodes de calcul sont plus difficiles à acquérir. Au delà de la valeur numérique de la distance, deux qualitatifs peuvent être ajoutés comme des métadonnées: proche (*close*) et très proche (*veryClose*). La Figure 56b illustre un exemple de relations de distance entre le lieu de prise d'une collection des photos et le Stade de France. Pour cet exemple, nous définissons que *très proche* correspond à une distance inférieure à 100 mètres et *proche* à une distance inférieure à 1 km. Ainsi, la relation *close(mdc, Stade de France)* doit exister dans l'annotation des photos de l'exemple. Nous montrons les méthodes existantes pour le calcul de ces relations spatiales qualitatives la section 6.3.

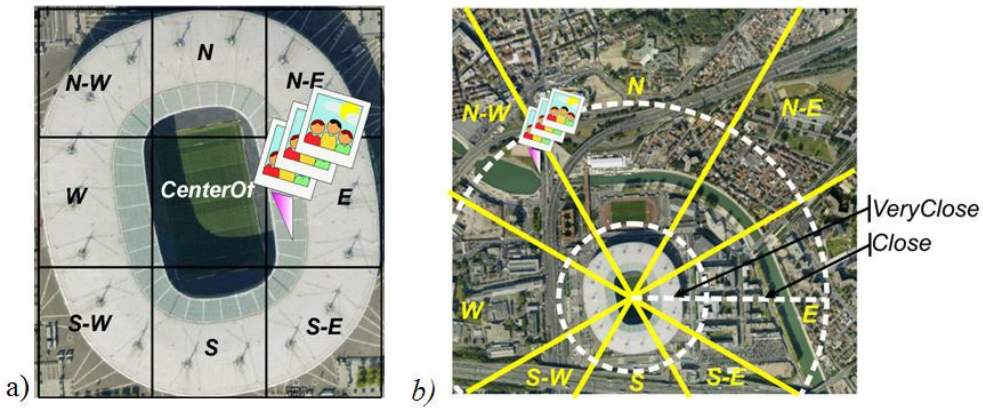


Figure 56 - Exemples d'utilisation des relations spatiales qualitatives.

Pour la représentation des **informations temporelles** relatives au contexte de création d'un document, nous n'avons pas eu besoin d'ajouter de nouveaux concepts à l'ontologie *ContextMultimedia*. En fait, le concept *Temporal_Element* de l'ontologie *Context Top* possède déjà des propriétés et des relations suffisamment riches pour représenter tous les attributs temporels nécessaires (par exemple, jour de la semaine, date, année, mois, etc.). Le concept *Temporal_Element* hérite cette capacité d'expression de l'ontologie OWL-Time.

Les concepts et les relations appartenant au **contexte social** d'un document multimédia personnel sont présentés dans Figure 57. Nous introduisons le concept *Person* pour modéliser les personnes présentes lors de la création d'un document multimédia. Nous réutilisons l'ontologie FOAF pour représenter les profils de ces personnes. Cette ontologie offre un vocabulaire pour décrire, par exemple, le nom de l'utilisateur (*foaf:surname*), l'email (*foaf:mbox*) et l'url des profils de connaissances de l'utilisateur (*foaf:knows*). Le concept Person diffère de *foaf:Person* grâce à deux nouvelles propriétés : *hasBluetoothDevice* et *isPresentIn*. La première relation permet d'associer un dispositif Bluetooth aux profils des

personnes. Cette information pourra être utilisée pour déduire la présence d'une personne lors de la création d'un document, puisque elle peut être comparée aux métadonnées stockées par le contexte informationnel. La propriété *isPresentIn* permet de lier le profil d'une personne à un segment du contenu multimédia. Par exemple, pour indiquer qu'une personne apparait lors d'une scène d'une vidéo. Nous définissons un sous-concept de *Person*, le concept *Creator*, pour stocker des informations relatives au producteur/propriétaire du document. Nous proposons également des sous-propriétés de *foaf:Knows* pour qualifier les relations sociales entre les individus (par exemple, que une personne A est l'épouse (*isWifeOf*) du créateur du document). Nous utilisons une adaptation du vocabulaire proposé par OntoAlbum (Chai *et al.*, 2008). La découverte du contexte social d'un document multimédia est détaillée dans la section 6.3.

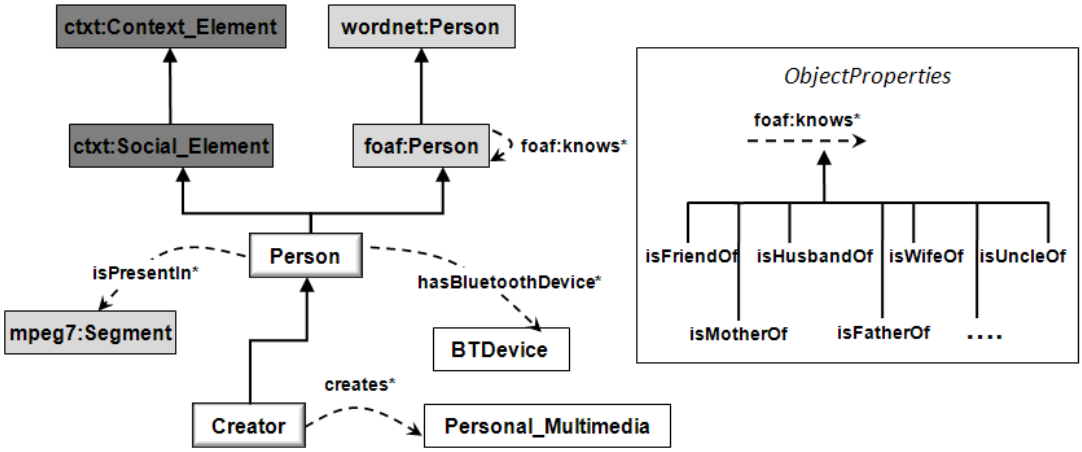


Figure 57 - Concepts et relations de la dimension sociale du contexte de création.

Nous modélisons **des métadonnées spatio-temporelles** en utilisant le concept *Physical_Environnement* qui représente les conditions météorologiques de l'environnement physique où l'action de création du document s'est déroulée. De plus, ce concept stocke d'autres attributs spatio-temporels (la saison, la température, etc.). La Figure 58 présente le concept *Physical_Environnement* et quelques unes de ces propriétés de données (*DataProperties*, en anglais). Nous pouvons décrire, par exemple, la saison lors de laquelle le document à été créé (propriété *season*), et la période de la journée par rapport au coucher/au lever du soleil (propriété *lightstatus*). Nous rappelons que le concept *Physical_Environnement* hérite de *ctxt:Spatial_Element*, nous pouvons ainsi lui associer le lieu auquel ces métadonnées spatio-temporelles font référence.

La Figure 58 illustre également les concepts de la **dimension informationnelle** du contexte stockés par l'ontologie ContextMultimedia. La modélisation des aspects informationnels fait appel aux concepts et aux relations de l'ontologie NEXIF[85] pour la description des attributs du dispositif utilisé (*Digital_Camera*) par l'utilisateur pour la création du document multimédia. Cette ontologie contient le même vocabulaire fourni par EXIF. L'intégration de NEXIF est assurée par la propriété *hasShotConfiguration* qui lie le concept *Digital_Camera* au concept *Shot_Configuration*. Celui-ci hérite de tous les attributs offerts par EXIF pour la description d'une photo, par exemple : l'intensité du flash employée pour la prise de la photo (*nexif :flashEnergy*), et la distance focale utilisée (*nexif :focalLenght*). Le concept *Shot_Configuration* doit être utilisé uniquement lors de la description des métadonnées d'une image.

[85] http://www.semanticdesktop.org/ontologies/nexif/

Au-delà des métadonnées EXIF, l**e contexte informationnel** comporte la description des dispositifs détectés à l'intérieur de la zone d'observation du système. Ainsi, une application mobile d'annotation peut exprimer, à l'aide de notre ontologie, des informations concernant les dispositifs Bluetooth à proximité du dispositif mobile hôte de l'application. Nous avons pu constater dans l'état de l'art que cette information est exploitée par certaines approches (telles que ContextWatcher [Koolwaaij *et al.*, 2006]) pour l'identification d'objets informatiques (des imprimantes, des bornes d'accès, etc.) et de personnes (les propriétaires des dispositifs Bluetooth détectés).

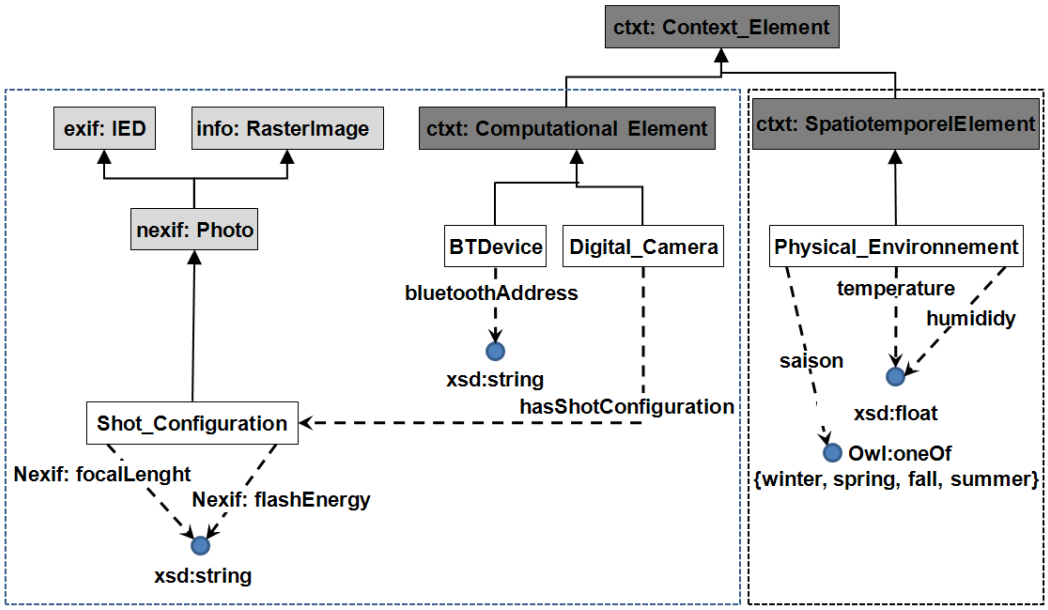


Figure 58 - Concepts des dimensions spatio-temporelle et informationnelle de ContextMultimedia.

La Figure 59 illustre un exemple de détection de dispositifs et l'annotation correspondante. Chaque adresse identifiée d'un dispositif Bluetooth est décrite en utilisant la classe *BTDevice*. Dans l'exemple, trois adresses de dispositifs voisins ont été acquises par le système d'annotation (0076AG8AA56B, 0012EE8AA91A, 0012EE8AB91A). Ces adresses sont uniques et peuvent identifier ainsi le dispositif (un fonctionnement semblable à l'adresse MAC-*Media Access Control* d'une carte réseau). À ce stade, les noms des propriétaires des dispositifs et leurs relations sociales avec le créateur du document sont toujours inconnus. Dans la section 6.3, nous illustrons des mécanismes qui infèrent ces informations à partir des adresses Bluetooth et d'une description du réseau social du créateur du document multimédia.

Il est important d'indiquer que certains concepts appartenant au contexte de création d'un document multimédia ne peuvent pas être facilement catégorisés dans une seule dimension de notre modèle en raison du caractère spatio-temporel de quasiment toutes ces métadonnées contextuelles. Par exemple, une personne peut être considérée comme étant à la fois un objet mobile et un élément du contexte social. Le concepteur d'un système de gestion de multimédia qui utilise notre modèle peut contourner cet inconvénient à l'aide de l'héritage multiple supporté par le langage OWL. Ainsi, un concept pourra hériter en même temps de divers sous-concepts de (*Context_Element*), et appartenir à diverses dimensions selon les besoins de modélisation.

Tous les concepts de *ContextMultimedia* présentés jusqu'ici point sont destinés à la description de métadonnées d'un seul document multimédia personnel. Nous introduisons à présent des métadonnées relatives à un ensemble de documents multimédias. L'objectif est de permettre la représentation de collections de documents (par exemple, des albums de photographies), et de la **notion d'événement** qui peut être associée à ces collections. La

Figure 60 illustre les concepts et les relations utilisés pour cette représentation. Une collection (concept *Collection*) possède (propriété *documents*) une liste de documents multimédias (*Personal_Multimedia*). Le concept *Collection* regroupe les mêmes propriétés du vocabulaire Dublin Core utilisées par le concept Personal_Multimedia, à l'exception des attributs décrivant le format et le type de document qui n'apportent pas de sens à une collection. Des annotations textuelles (*Textual_Annotation*) sont également liées à une collection par le biais de la propriété *textual_annotation*. Un outil d'annotation peut utiliser, par exemple, les sous-concepts de *Textual_Annotation* pour associer à une collection les mots-clés les plus utilisés pour l'annotation des documents de la collection.

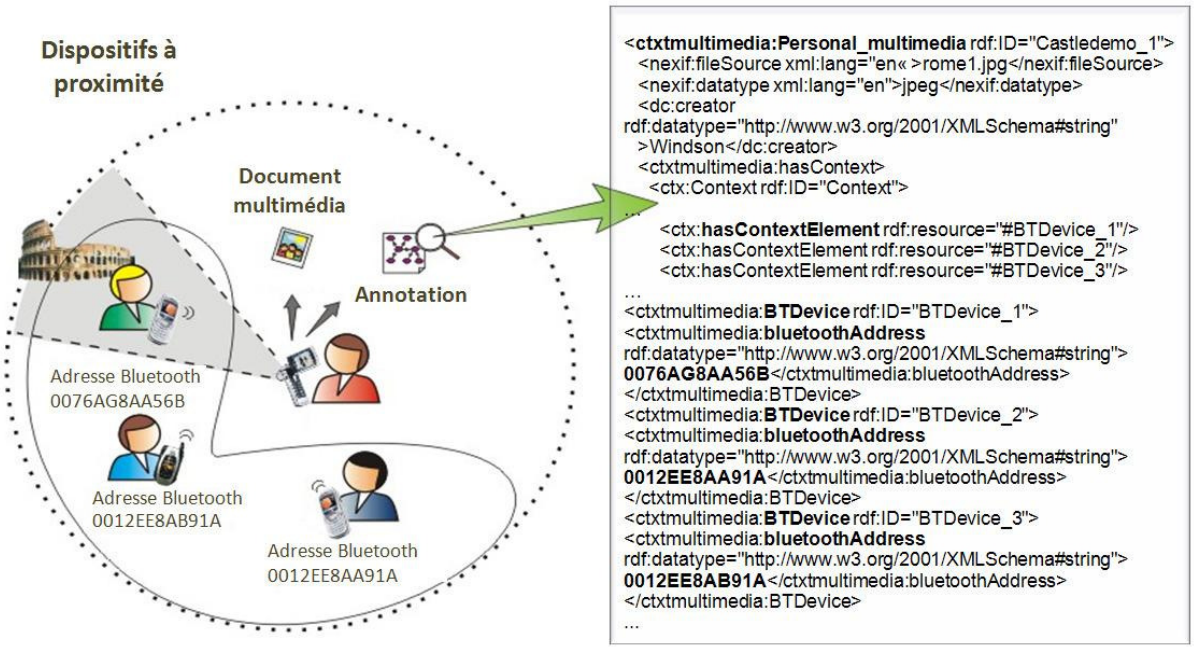


Figure 59 - Exemple d'annotation relative au contexte informationnel.

La **notion d'événement** est introduite dans notre modèle grâce au concept *EventCollection* qui est un sous-concept de Collection. Suite aux considérations de l'aspect spatio-temporel inhérent à un événement, nous intégrons au concept *Event_Collection* des capacités de description de propriétés temporelles (héritées de *time :Interval*) et spatiales (héritées de *gml:Feature*). Ce choix de modélisation confère à un outil d'annotation la possibilité d'associer à une collection un intervalle qui signale la durée d'un événement, et une région géoréférencée pour indiquer le lieu où cet événement s'est déroulé. Une autre représentation spatiale possible dans notre modèle est la représentation du trajet parcouru par l'utilisateur pendant la durée de l'événement. Cette représentation est assurée par les concepts *TrackPoint* et *TrackList*.

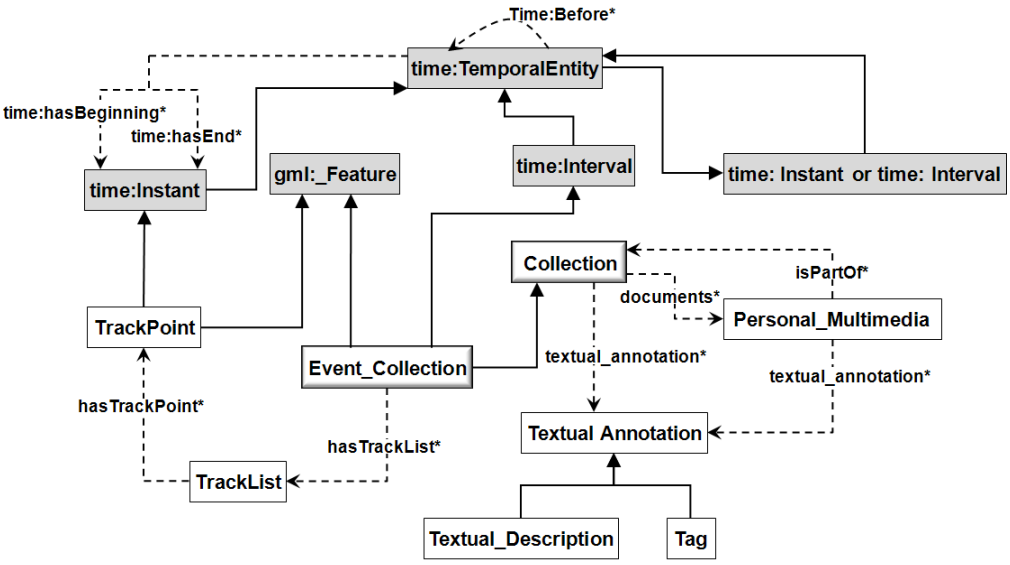


Figure 60 - Représentation d'une collection et de la notion d'événement dans ContextMultimedia.

La Figure 61 illustre la caractérisation d'un événement telle que nous envisageons de la représenter à l'aide de notre modèle. Il s'agit d'une promenade touristique à Rome en Italie. Au départ de la promenade, l'utilisateur signale le début d'événement (instant t_0). Pendant toute sa durée, le dispositif mobile garde une trace du déplacement de l'utilisateur en le représentant à l'aide du concept *TrackList*.

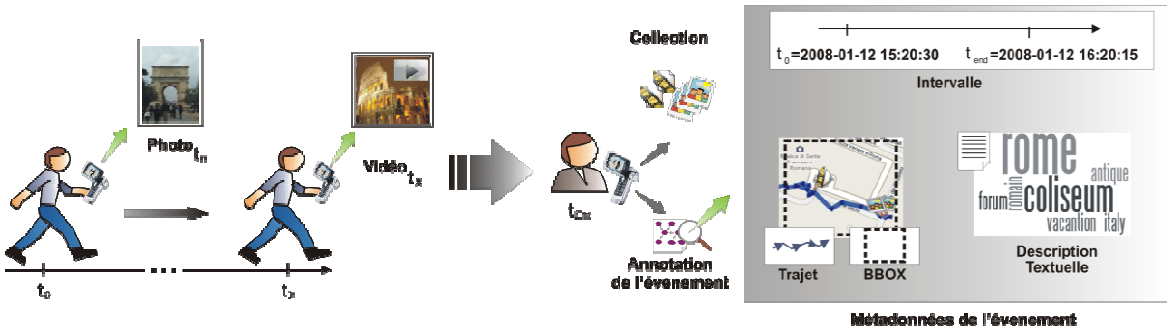


Figure 61 - Exemple de collection associée à un événement.

À la fin de l'événement (instant t_{end}), les documents multimédias créés sont regroupés et associés à l'annotation de la collection. Ces métadonnées incluent l'intervalle de l'événement (t_0 à t_{end}), les mots-clés les plus utilisés par l'utilisateur pour annoter ses documents, et deux informations spatiales : le trajet et la BBOX. La BBOX indique la région géographique où s'est déroulé l'événement. Elle peut être calculée comme étant la plus-petite région englobant le trajet. Toutes ces métadonnées peuvent être représentées par les concepts illustrés dans la Figure 60.

Dans l'état de l'art, nous avons vu des approches (tel que PhotoGeo) qui tente de générer des collections représentant des événements en utilisant des méthodes de partitionnement de données (*data clustering*, en anglais). Parfois, ces regroupements générés peuvent représenter de faux événements qui doivent être corrigés par les utilisateurs. Afin de proposer un modèle d'annotation générique qui puisse également être utilisé par ce type d'approche, nous intégrons au concept *Event_Collection* des propriétés pour la description du mécanisme exploité lors de la construction de la collection spatio-temporelle (par exemple : manuel, automatique) et son état actuel de validation (propriété *isValidated*).

Dans le Tableau 7, nous présentons la représentation en OWL de quelques concepts et relations de l'ontologie ContextMultimedia. Le fichier complet de l'ontologie est disponible sur le site du projet PhotoMap[86]. Nous tenons à préciser qu'un outil d'annotation n'est pas obligé d'utiliser tous les concepts de notre modèle, car l'ontologie ContextMultimedia n'impose la présence d'aucun concept pour la définition de métadonnées d'un document multimédia personnel. Ainsi, le concepteur de l'outil d'annotation peut choisir les concepts de notre modèle selon les besoins de l'application et sa capacité à acquérir et à calculer les métadonnées.

Tableau 7 – Description en OWL de certains concepts de l'ontologie ContextMultimedia.

	Syntaxe en logique de description	Représentation en OWL
(1)	$\sqsubseteq \top$	*owl:Thing*
(2)	gml:Feature $\sqsubseteq \top$ gml:Geometry $\sqsubseteq \top$	*gml:Feature* ***rdfs:subClassOf*** *owl:Thing* *gml:Geometry* ***rdfs:subClassOf*** *owl:Thing*
(3)	ctxt:Spatial_Element $\sqsubseteq$ gml:Feature	*ctxt:Spatial_Element* ***rdfs:subClassOf*** *gml:Feature*
(4)	geoRSS:where $\exists$geoRSS:where. $\sqsubseteq$ gml:Feature $\top \sqsubseteq \forall$geoRSS:where.gml:Geometry	*geoRSS:where* est une ***rdfs: ObjectProperty*** *rdfs:domain = gml:Feature* *rdfs:range = gml:Geometry*
(5)	isWifeOf $\sqsubseteq$ foaf :Knows isHusbandOf $\sqsubseteq$ foaf :Knows (isWifeOf)$^{-}$ $\equiv$husbandOf	*isWifeOF* ***rdfs:subPropertyOf*** *foaf :Knows* *isHusbandOf* ***rdfs:subPropertyOf*** *foaf :Knows* *isWifeOF* ***owl:inverseOf*** *husbandOf*
(6)	temperature_value $\exists$temperature_value $\sqsubseteq$ Physical_Environement $\top \sqsubseteq$xsd:float	*temperature_value* ***owl:FunctionalProperty*** *rdfs:domain = Physical_Environment* *rdfs:range =xsd:float*
(7)	Georeferenced_Object $\sqsubseteq$ ctxt:Spatial_Element Wikipedia_Entry $\sqsubseteq$ Georeferenced_Object	*Georeferenced_Object* ***rdfs:subClassOf*** *ctxt:Spatial_Element* *Wikipedia_Entry* ***rdfs:subClassOf*** *Georeferenced_Object*
(8)	BTDevice $\sqsubseteq$ ctxt:Computational_Element	*BTDevice* ***rdfs:subClassOf*** *ctxt :Computational_Element*
(9)	Place $\sqsubseteq$ ctxt:Spatial_Element	*Place* ***rdfs: subClassOf*** *ctxt:Spatial_Element*

6.2.3 Exemple d'annotation

Nous présentons dans cette section un exemple d'utilisation de notre ontologie pour l'annotation d'un document multimédia personnel. Nous montrons comment les concepts et les relations de *ContextMultimedia* représentent les métadonnées de la photo de l'exemple de la Figure 53 (section 6.2.1). Ces métadonnées peuvent être acquises automatiquement ou saisies manuellement par le créateur de la photo. Un système de gestion de multimédia peut ainsi exploiter notre ontologie pour le stockage de ces métadonnées.

Pour annoter un document multimédia personnel, une instance de l'ontologie *ContextMultimedia* doit être créée. Une représentation visuelle de l'annotation de notre document exemple est illustrée dans la Figure 62. Les noms des instances suivent un canevas : **le préfixe « _ » suivi du nom du concept instancié**. Nous pouvons textuellement interpréter l'annotation comme étant : « le document multimédia personnel est une image (*dc:type= image*) appartenant à une collection (*_Event_Collection*) qui décrit un événement (*dc:title= « U2 Concert »).* Deux mots-clés (*_Tag_1* et *_Tag_2*) sont associés à cette image. Le premier mot-clé (*tag= « U2 »*) a été ajouté par l'utilisateur, et le deuxième (*tag= « Stade »*) par le système. Le mot-clé généré automatiquement n'a pas été encore validé par un humain (*isValidated= false*). Concernant le contexte de création de ce document

[86] http://pyro.imag.fr/PhotoMap/

(*_ctxt :Context*), deux dispositifs Bluetooth (*_BT_Device_1* et *_BT_Device_2*) ont été détectés à proximité du lieu de prise de la photo. D'ailleurs, cette photo a été prise un Samedi (*time :dayofweek= «saturday»*) par un appareil photo numérique de modèle Nokia_N95 (*_Digital _Camera*). Au moment de la création de l'image, la température était de 26°C et le soleil se couchait (*lightstatus= « sunset »*)".

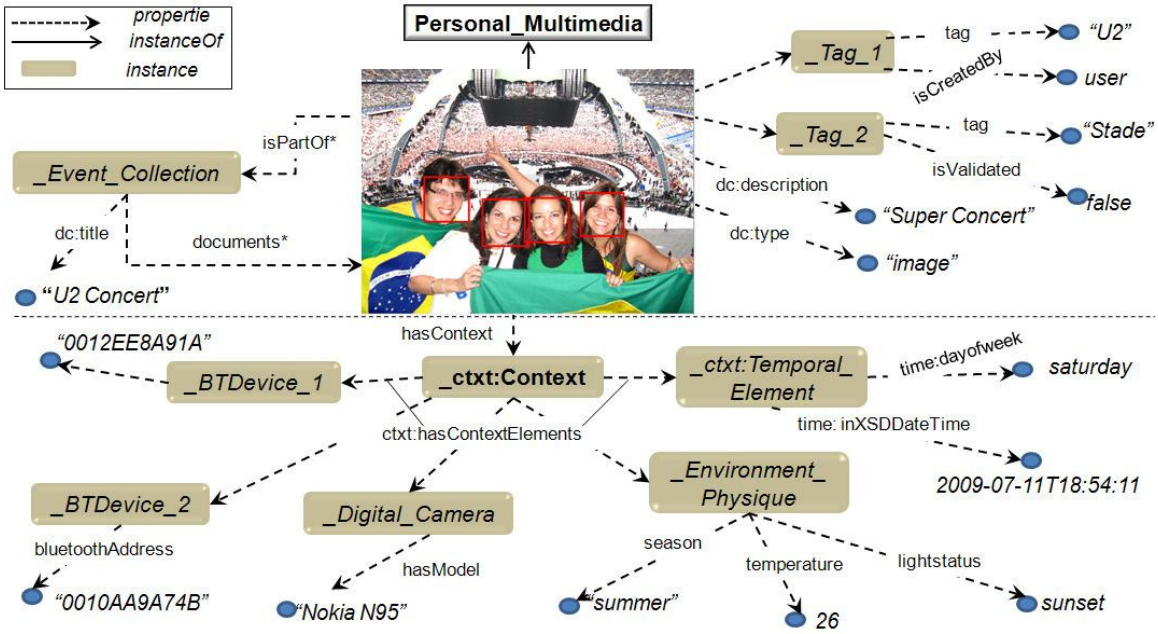


Figure 62 – Représentation visuelle de l'annotation d'une photo à l'aide de *ContextMultimedia*.

La Figure 63 illustre l'annotation du contenu et des dimensions spatiale et sociale du contexte de création de la photo. Une région de l'image (*_StillRegion*) est mise en valeur sur la photo pour identifier le visage d'une des personnes qui apparaissent sur l'image. Nous pouvons traduire l'annotation représentée par ContextMultimedia en langage naturel : « L'image possède une annotation sur son contenu (*_mpeg7:Image*) qui consiste en un découpage spatial (*mpeg7:spatial_decomposition*). En fait, ce découpage (*_mpeg7:StillRegion*) est un polygone décrit par les coordonnées {200 210 200 280 … 200 210}. Le contexte social de l'image est composé de deux personnes : Karol (*_Person*) et Windson (*_Creator*). Windson est le créateur/propriétaire de l'image. Karol fait partie du réseau social (*foaf :knows*) de Windson, en réalité, elle est son épouse (*wifeOf*). Son visage est également présent (*isPresentIn*) dans le segment *_StillRegion*. La photo a été prise dans un endroit (*_Place*) dont les coordonnées géographiques sont «48.9272 2.3612 ». Cet endroit se situe à l'intérieur d'un objet géoréférencé (*_GeoreferenceObject*) nommé Stade de France. L'adresse (*_Address*) du lieu de création de la photo est l'avenue Général de Gaulle, Saint Denis, France.

Le Tableau 8 présente la sérialisation en RDF d'une partie de l'annotation de la photo exemplifiée. Nous pouvons noter que la personne (*_Person*), dont le visage est affiché dans le segment *_StillRegion*, possède un des dispositifs Bluetooth détectés à proximité de le lieu de prise de la photo (Figure 62).

Ces métadonnées sérialisées peuvent être inclues dans le fichier de la photo, par exemple, en les incorporant à l'entête EXIF disponible dans le format d'image JPEG (paramètre description). Ce choix de stockage garanti une plus grande **mobilité à l'annotation**. Un système d'annotation peut également stocker les métadonnées dans un fichier séparé du document multimédia afin d'éviter l'ouverture et la décompression des fichiers à chaque lecture des métadonnées.

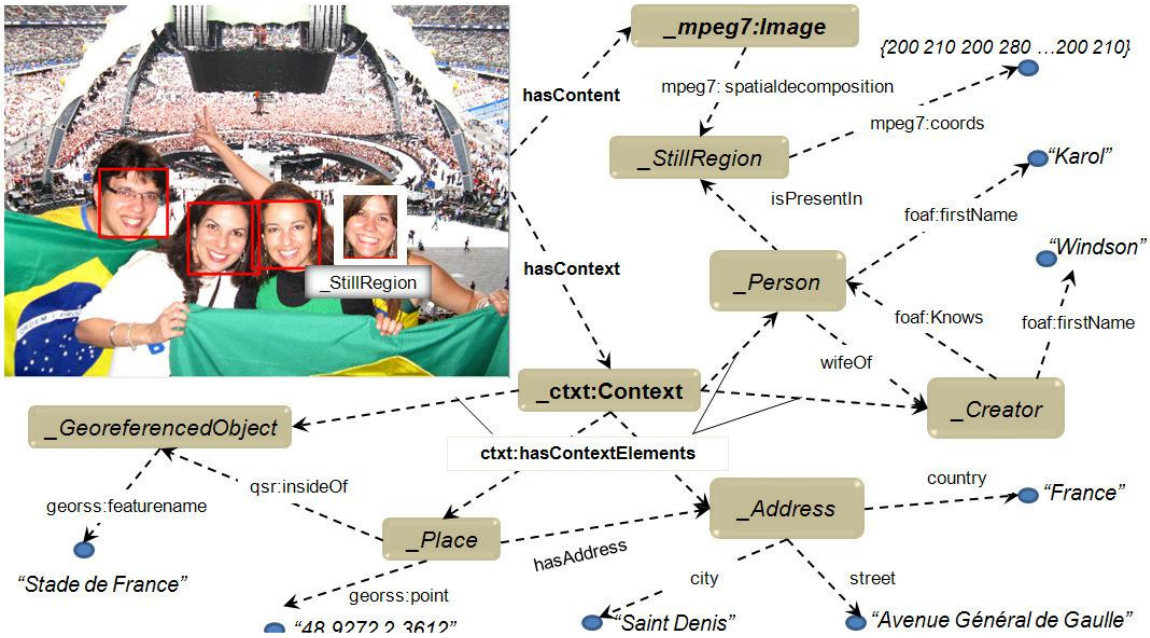


Figure 63 – L'annotation sur le contenu et sur les contextes spatial et social à l'aide de *ContextMultimedia*.

Tableau 8 - Sérialisation en RDF/XML de l'annotation de la photo exemple.

```
<?xml version="1.0"?>
<rdf:RDF  <owl:Ontology rdf:about="stade1.jpg">  </owl:Ontology>
  …
      <ctxtMultimedia:Personal_Multimedia
rdf:ID="_Personal_Multimedia">
        <dc:identifier
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >stade1.jpg</dc:identifier>
        <dc:creator
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Windson</dc:creator>
        <ctxtMultimedia:isPartOf rdf:resource="#_Event_Collection"/>
        <ctxtMultimedia:hasContent>
          <mpeg7:Image rdf:about=" #_mpeg7:Image">
            <mpeg7:spatial_decomposition rdf:resource=" #_StillRegion"/>
          </mpeg7:Image>
        </ctxtMultimedia:hasContent>
        <ctxtMultimedia:hasContext>
          <ctxt:Context rdf:ID="_Context">
            <ctxt:hasContextElement>
    <ctxt:hasContextElement rdf:resource=" #_Place"/>
            <ctxt:hasContextElement rdf:resource="#_Person"/>
            <ctxt:hasContextElement>
...
  <ctxtMultimedia:Place rdf:ID="_Place">
    <ctxtMultimedia:insideOf>
      <ctxtMultimedia:GeoreferencedObject
rdf:ID="_GeoreferencedObject">
        <georss:featurename
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Stade de France</georss:featurename>
```

```
      </ctxtMultimedia:GeoreferencedObject>
    </ctxtMultimedia:insideOf>
    <georss:point
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >48.9272 2.3612</georss:point>
  </ctxtMultimedia:Place>
  <ctxtMultimedia:Creator rdf:ID="_Creator">
    <ctxtMultimedia:husbandOf>
      <ctxtMultimedia:Person rdf:ID="_Person">
        <ctxtMultimedia:ispresentedIn>
          <mpeg7:StillRegion rdf:ID="_StillRegion">
            <mpeg7:Spatial_Mask>
              <mpeg7:Polygon rdf:ID="Polygon">
                  <mpeg7:Coords rdf:ID="mpeg7_Coords_32">
                    <coords
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    >200 210 200 280 ...200 210</coords>
                  </mpeg7_Coords>
              </mpeg7:Polygon>
            </mpeg7_Spatial_Mask>
          </mpeg7:StillRegion>
        </ctxtMultimedia:ispresentedIn>
        <ctxtMultimedia:wifeOf rdf:resource="#_Creator"/>
        <ctxtMultimedia:bluetoothAddress
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >0010AA9474B</ctxtMultimedia:bluetoothAddress>
        <foaf:firstName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Karol</foaf:firstName>
      </ctxtMultimedia:Person>
    </ctxtMultimedia:husbandOf>
    <foaf:firstName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Windson</foaf:firstName>
  </ctxtMultimedia:Creator>
....
```

6.3 ContextAnnotator : une plate-forme d'acquisition et d'enrichissement des métadonnées associées aux documents multimédias personnels

Le vocabulaire fourni par l'ontologie ContextMultimedia permet la représentation formelle de quasiment toutes les annotations présentes dans les approches de gestion de documents multimédias décrites dans l'état de l'art (Chapitre 3). Dans cette section, nous proposons une plate-forme d'acquisition et d'inférence de ces annotations dénommée *ContextAnnotator*.

Nous souhaitons produire, à l'aide de notre plate-forme, une grande quantité de métadonnées relatives aux documents multimédias personnels créés par des dispositifs mobiles. Lors de la création de ces documents, un ensemble initial d'informations liées au contexte de création doit être capturé par l'application mobile de production de contenu (par exemple, le logiciel de prise de photo). Ces informations initiales (telles que la date et l'heure, et les coordonnées géographiques du dispositif mobile) doivent être ensuite représentées sur forme d'instances de l'ontologie *ContextMultimedia*. L'objectif de la plate-forme *ContextAnnotator* est d'augmenter le nombre et d'améliorer le niveau sémantique des métadonnées du document multimédia, à partir de ces informations initialement capturées par le dispositif mobile lors de la création du document.

Les capacités de calcul et d'accès à des sources de données exigées par ces mécanismes d'enrichissement de métadonnées empêchent néanmoins la mise en œuvre de *ContextAnnotator* directement sur les plates-formes mobiles d'aujourd'hui. Cette restriction advient des caractéristiques matérielles et logicielles encore limitées des plates-formes mobiles, comme nous avons pu le constater dans l'état de l'art. Ainsi, *ContextAnnotator* est conçu pour une exécution sur des ordinateurs de bureau ou sur des serveurs Web dont l'accès internet est supposé illimité et stable.

Nous envisageons une plate-forme d'enrichissement de métadonnées flexible et extensible, afin que les mécanismes de génération de métadonnées puissent être étendus pour répondre aux divers besoins des services de gestion de multimédias reposant sur CoMMediA (des services d'annotation, des services de partage de contenu, etc.).

La Figure 64 illustre les deux cas d'utilisation envisagés pour *ContextAnnotator* :

- *Un usage destiné aux services d'annotation et d'organisation sur des ordinateurs de bureau*. Ces services, couplés à des outils de gestion de multimédia, vont utiliser *ContextAnnotator* pour enrichir les métadonnées de chaque nouveau document multimédia. Une pré-condition d'utilisation de *ContextAnnotator* est l'existence de métadonnées initiales qui comprennent au moins la date et l'heure de création des documents et une information de localisation géographique (des coordonnées géographiques, le nom du lieu de création, l'ID de la cellule GSM, etc.). En utilisant ces informations initiales, *ContextAnnotator* peut produire de nouvelles métadonnées qui pourront ensuite être validées par les utilisateurs, et exploitées dans le processus d'indexation des documents multimédias.

- *Un usage destiné aux services mobiles de partage de documents multimédias*. Lors de la création de documents multimédias, les personnes n'ont pas forcement besoin de connaitre les métadonnées associées à leurs documents, car à ce moment précis le centre d'intérêt des personnes est surtout la création du document multimédia lui-même [Sarvas *et al.*, 2004a]. Ce comportement change lorsque les personnes souhaitent partager leurs documents multimédias [Ames *et al.*, 2007]. Les métadonnées, produites par *ContextAnnotator*, peuvent être exploitées par les services de partage, à la fois pour la recommandation des annotations qui doivent être publiées avec le document, et pour

la génération des messages de partage de contenu (tels que le texte qui accompagne une image dans un MMS).

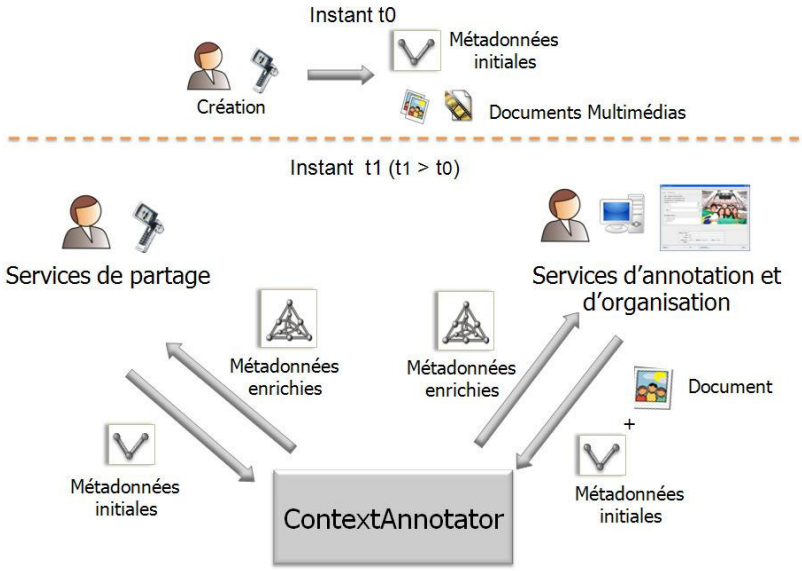


Figure 64 - Cas d'utilisation de ContextAnnotator.

Suite à ces cas d'usage, nous pouvons constater que ContextAnnotator n'a pas besoin de dériver en boucle des informations relatives au contexte des personnes. Contrairement à d'autres architectures d'acquisition et d'inférence de contexte (Chapitre 3 et 5), lors de la création d'un contenu, un ensemble réduit d'informations doit être capturé et, ultérieurement, ces métadonnées nécessitent d'être enrichies. Par conséquent, les nouvelles métadonnées seront générées sur demande explicite des services (mobiles ou non) à la plate-forme *ContextAnnotator*.

6.3.1 Architecture de ContextAnnotator

Afin de répondre aux besoins de ces deux cas d'utilisation de *ContextAnnotator*, nous modélisons les processus d'enrichissement comme des compositions séquentielles de plusieurs services Web, dont chaque service génère un certain nombre de métadonnées. La structuration du processus d'enrichissement profite du couplage faible fourni par la technologie de services Web pour garantir la flexibilité et extensibilité de la plate-forme *ContextAnnotator,* car différentes compositions de services peuvent être proposées plus facilement. L'usage des services Web permet également l'accès à ces mécanismes depuis les plates-formes mobiles aussi bien que depuis des ordinateurs de bureau (connecté à l'internet).

Nous définissons d'abord la structure générique d'un service d'enrichissement de *ContextAnnotator*. La Figure 65 illustre cette structure qui doit être adoptée par les concepteurs de services de la plate-forme. Deux catégories de services sont prévues sur la plate-forme : i) les services d'enrichissement qui reposent uniquement sur les métadonnées (*MedataFromMetadataService*), et ii) les services qui utilisent en plus le contenu du document (MetadataFromContentService). Chaque service d'enrichissement de métadonnées de *ContextAnnotator* possède au moins deux méthodes : *generateMetadata* et *changeConfigurations*.

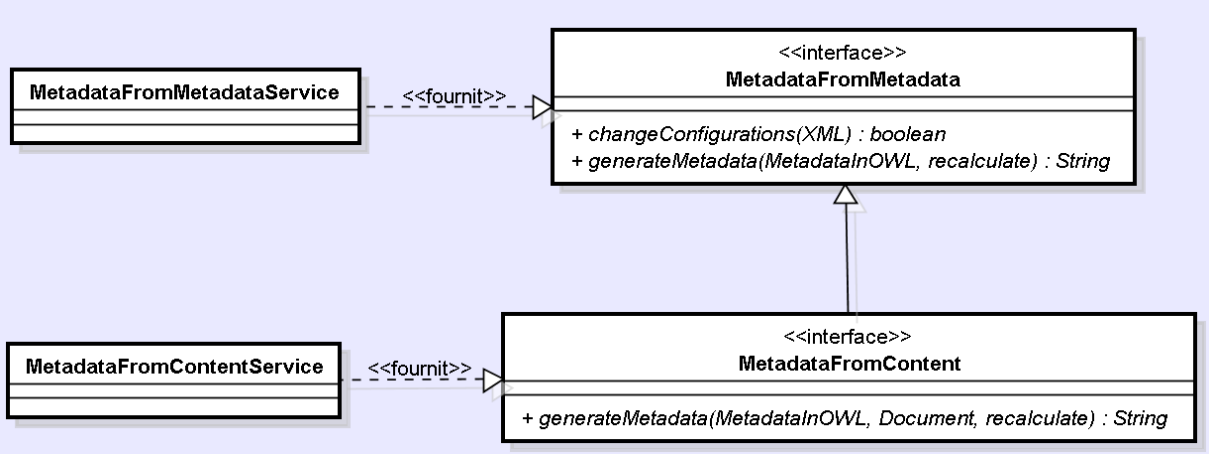


Figure 65 - La modélisation de services d'enrichissement de métadonnées

Le mécanisme d'enrichissement de métadonnées d'un service est déclenché par l'appel de la méthode *generateMetadata*. Le client du service fournit une ontologie (une instance de ContextMultimedia) contenant les métadonnées d'un document multimédia qui ont été déjà capturées ou générées. Chaque service tente d'augmenter ces métadonnées et représente le résultat sous-forme d'ontologie. En fait, une chaîne de caractères représentant la sérialisation d'une instanciation de l'ontologie *ContextMultimedia* est un des paramètres d'entrée de la méthode *generateMetadata*. Un service de *ContextAnnotator* doit puiser ses paramètres d'entrée parmi les métadonnées déjà disponibles sur l'ontologie.

Le résultat de la méthode doit être stocké dans l'ontologie et retourné au client du service. Par exemple, un service qui génère une adresse à partir de coordonnées géographiques doit extraire ces coordonnées dans l'ontologie initiale et doit stocker dans le fichier de cette ontologie l'adresse correspondant aux coordonnées. En l'absence des paramètres d'entrée ou s'il est impossible de dériver de nouvelles métadonnées, le résultat de la méthode doit être sans effet pour l'ontologie initiale, afin de ne pas compromettre une composition de services d'enrichissement dont ce service fait partie.

Ce choix de modélisation offre un appel standard de tous les services d'enrichissement de *ContextAnnotator*. De plus, la sortie d'un service d'enrichissement peut être utilisée comme paramètre d'entrée d'un autre service, facilitant ainsi leur composition. L'autre paramètre de la méthode *generateMetadata* est une variable booléenne (*recalculate*) qui signale la nécessité de recalculer ou non les métadonnées susceptibles d'être générées par le service. Si la valeur de ce paramètre vaut *faux*, le service doit vérifier d'abord la présence du type de métadonnées qu'il génère. En cas de présence, le service ne doit pas les recalculer. Par exemple, le service de génération d'une adresse vérifie si une instance du concept *Address* est déjà présente dans l'ontologie de départ. Si la valeur est *vrai*, le service doit générer la nouvelle valeur du concept, et effacer l'existante dans l'ontologie.

Au delà de l'ontologie et de la variable booléenne, les services fournissant des métadonnées à partir du contenu d'un document multimédia ont besoin également de ce document comme paramètre d'entrée. Bien que les services doivent avoir une pré-configuration minimum de fonctionnement, des aspects de configuration liés à la sémantique de chaque service peuvent être modifiés par la méthode *changeConfigurations*. L'objectif est de fournir à la fois un fonctionnement standard de chaque service, et une adéquation de son fonctionnement au besoin du client. Un service d'enrichissement peut être appelé directement par le client ou par le biais de la plate-forme *ContextAnnotator*.

La Figure 66 illustre l'architecture de *ContextAnnotator*. Une entité est chargée de traiter les réquisitions (*Annotation Broker*) et de décider le service ou de la composition séquentielle à utiliser pour l'enrichissement. Une composition séquentielle est elle-aussi un service

d'enrichissement contenant la méthode *generateMetadata*. Un client de méthodes d'enrichissement appelle directement un service par son URI, ou informe la plate-forme le concept de l'ontologie qu'il compte enrichir (par exemple, *Address* pour générer l'adresse postale). *Annotation Broker* consulte son registre de services qui stocke une mise en correspondance entre les concepts de l'ontologie et les services (ou les compositions) qui peuvent les générer. *Annotation Broker* choisit ensuite le service ou la composition à déclencher. Nous souhaitons ainsi isoler du client du service la connaissance précise du service qui réellement génère les métadonnées. Le concepteur de plate-forme peut plus facilement modifier ce service étant donné que son interface (par exemple, *MedataFromMetadata*) reste inchangée.

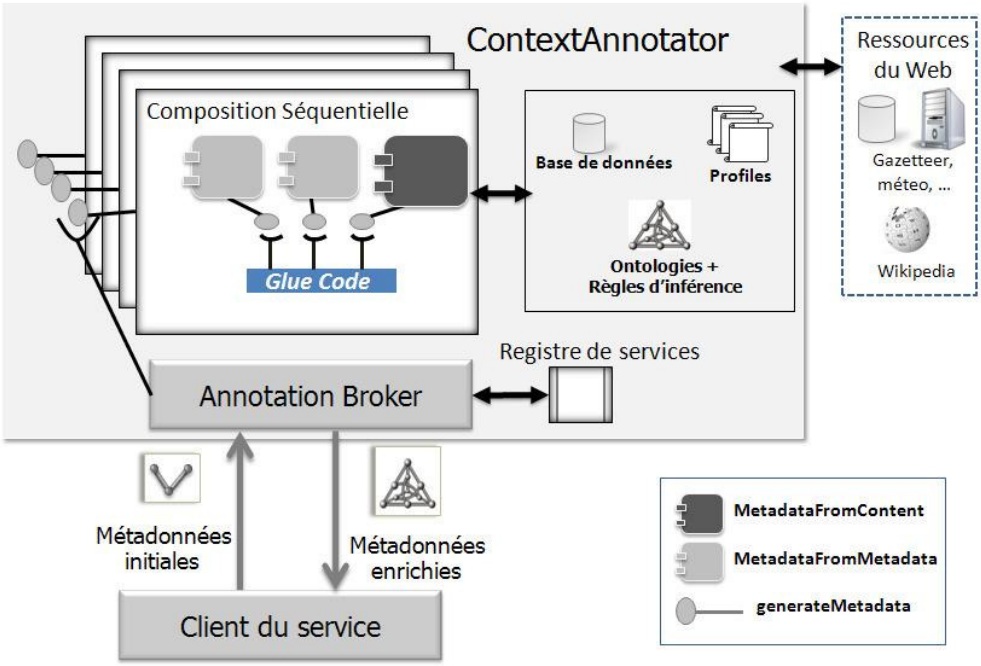


Figure 66 - Architecture de la plate-forme ContextAnnotator.

Un concepteur qui souhaite ajouter un nouveau service d'enrichissement, soit modifie les compositions existantes, soit ajoute une nouvelle composition en modifiant le registre de services d'*AnnotationBroker*.

Les services de notre plate-forme exploitent deux catégories de mécanismes d'enrichissement de métadonnées : **i) méthodes d'interprétation du contexte de création et du contenu du document**, et ii) **méthodes de dérivation de métadonnées**.

Les **méthodes d'interprétation du contexte de création et du contenu du document** proposent un accroissement des métadonnées à partir des premières informations symboliques acquises par les capteurs du dispositif mobile lors de la création d'un document multimédia personnel. Ces informations sont combinées à des sources de données du Web (gazetteer, services de météorologie, base de données de points d'intérêt, etc.) qui permettent découvrir des informations d'un niveau sémantique plus élevé (tel que la transformation des coordonnées géographiques en adresse, les noms des points d'intérêts à proximité, la période de la journée par rapport au coucher/au lever du soleil, etc.). Des méthodes classification/analyse de contenu (par exemple, la détection d'objets ou visages sur des photos) peuvent également être ajoutées à cette étape d'enrichissement de métadonnées. Les processus d'interprétation déjà disponibles sur notre plate-forme sont décrits dans la section 6.3.2.

Une fois établie une large quantité d'information relative au contexte de création, **des méthodes de dérivation de métadonnées** sont mises en place pour inférer de nouvelles relations sémantiques entre les concepts instanciés de l'ontologie. Il s'agit principalement de calculs des relations spatiales, et de l'exécution d'un ensemble des règles d'inférence relatives au contexte « social » du document. Par exemple, l'établissement des relations sémantiques entre le créateur du contenu et les personnes présentes lors de la création du document. Nous décrivons ces méthodes de dérivation dans la section 6.3.3.

6.3.2 Processus d'interprétation du contexte de création

Les services d'interprétation disponibles sur *ContextAnnotator* génèrent de nouvelles métadonnées liées au contenu du document et à trois dimensions du contexte (Spatiale, Spatio-temporelle et Temporelle). Quatre métadonnées sont fondamentales pour l'exécution de ces services : i) l'identificateur du créateur, ii) une localisation, iii) une date, et iv) une heure. Ces informations doivent être fournies par les clients des services d'interprétation et doivent être disponibles sur l'ontologie initiale. La date et l'heure représentent l'instant de création du document (par exemple, le début d'enregistrement pour une vidéo), alors que la localisation établit la position géographique du dispositif mobile lors de la création du document. Dans le prochain chapitre, nous décrivons des méthodes pour capturer ces informations initiales.

Nous présentons à présent les services d'interprétation. Ces services sont décrits selon la dimension contextuelle à laquelle ils font référence.

6.3.2.1 Dimension spatiale du contexte

Sept services sont disponibles pour la génération de métadonnées spatiales. Ils sont illustrés par la Figure 67. Les services de la catégorie *Geocoding* : « MyAddress », « NearbyPOI » et « MyPlaces » cherchent dans l'ontologie d'entrée une instance du concept *Place* appartenant au contexte de création du document. Plus spécifiquement, ces services ont besoin des coordonnées géographiques associées à l'instance du concept *Place* qui décrit le lieu de création du document (propriété *georss:point*).

Le service *MyAddress* fournit l'adresse administrative la plus proche de ces coordonnées géographiques. Il utilise deux méthodes pour le calcul. La première méthode exploite une base de données géographique contenant le contour de certaines villes de l'Europe[87]. La deuxième méthode utilise le service *Find Nearest Address* du site Geonames[88]. Le service de Geonames offre le nom de la ville (et de son Pays) dont le centre est le plus proche des coordonnées géographiques. En conséquence, le service Geonames peut présenter des erreurs dans ces résultats, car les points aux abords des villes peuvent être identifiés comme appartenant à des villes voisines, dont le centre est plus proche des coordonnées géographiques. Malgré cet inconvénient, Geonames offre davantage de données sur pratiquement toutes les villes du monde. Initialement, le service *MyAddress* utilise la première méthode et, en cas de résultat nul, le service Geonames est invoqué. Les résultats du service *MyAddress* sont stockés dans les propriétés de données (*DataProperties*) du concept *Address* de l'ontologie ContextMultimedia.

[87] Nous utilisons une base PostGis générée à partir des fichiers MIF/MID exploités dans le projet HyperCarte. La base de données contient, par exemple, tous les contours des villes de la France métropolitaine.

[88] http://www.geonames.org/maps/reverse-geocoder.html#findNearestAddress

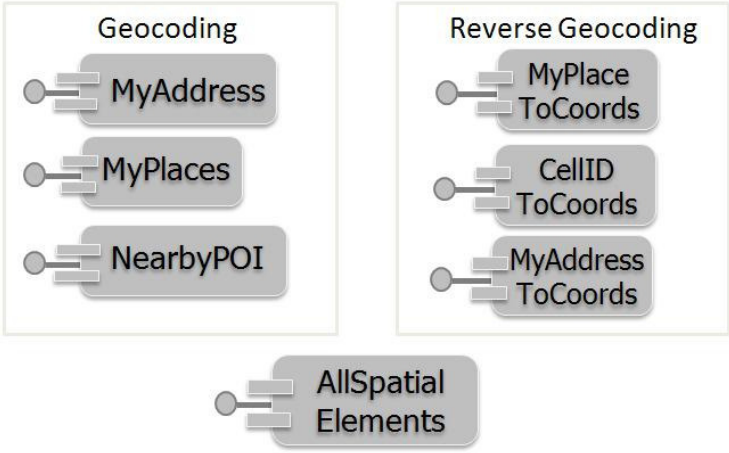


Figure 67 - Services de la dimension spatiale.

Le service *NearbyPOI* remplit l'ontologie avec des informations sur les dix plus proches points d'intérêt dans un rayon d'un kilomètre des coordonnées géographiques. En fait, ce service utilise le Wikipedia comme fonds documentaire de monuments touristiques. Ces descriptions sont utiles pour enrichir les annotations, puisque, dans certains cas, elles sont directement liées aux objets présents sur les photos et les vidéos créés par les utilisateurs. Le service *NearbyPOI* fait appel au service *Find Nearby Wikipedia Entries*[89] *pour avoir* accès uniquement aux rubriques Wikipedia géoréférencées.

Les résultats sont inclus dans l'ontologie en utilisant le concept *Wikipedia_Entry* (Figure 55). La distance du rayon de recherche et le nombre de points d'intérêt retenus sont modifiables par la méthode de changement de configurations du service (changeConfigurations).

Le troisième service de la catégorie *Geocoding* tente de trouver une description personnelle du lieu de création du document. Ce service, nommé *MyPlaces*, repose sur la construction d'un registre de descriptions (dénominations) de lieux créées par les utilisateurs eux-mêmes. L'objectif du service est de fournir une description du lieu de création plus proche du sens attribué par un utilisateur ou par une communauté à ce lieu. La Figure 68 illustre un exemple d'interface graphique d'insertion de lieux personnels. L'utilisateur peut dessiner des régions sur carte de la terre et leur attribuer des descriptions personnelles. Par exemple, la personne peut définir le contour d'un bâtiment et le nommé « mon travail ». Cette information peut être également renseignée sur une application mobile qui associe les coordonnées géographiques du dispositif mobile (moyennant un rayon d'erreur) au nom de lieu indiqué par l'utilisateur.

Lorsque le service *MyPlaces* est invoqué, il identifie le créateur du document et cherche dans une base de données géographique les descriptions des lieux personnels de cet utilisateur qui englobent les coordonnées géographiques initialement liées au document multimédia. Les descriptions de lieux qui satisfont cette requête sont stockées par l'ontologie *ContextMultimedia* en tant qu'objets géoréférencés (instances du concept *Georeference_Object*). La relation spatiale topologique « à l'intérieur de » (propriété *insideOf*) est introduite entre le lieu de création (instance du concept *Place*) et la nouvelle instance de *Georeferenced_Object*. Le service *MyPlaces* peut être configuré pour insérer également les lieux personnels qui sont les plus proches du lieu de création du document multimédia (par exemple, le service établit qu'une photo a été prise à moins d'un kilomètre du lieu personnel « mon travail »).

[89] http://www.geonames.org/export/wikipedia-webservice.html#findNearbyWikipedia

Figure 68 - Un exemple d'interface d'insertion de lieux personnels. Les polygones dessinés sur la carte indiquent le lieu de travail (« mon travail ») et objet géoréférencé (« BU Sciences ») à proximité.

La mise en œuvre de ces trois services d'interprétation exige, de la part du client du service, la découverte des coordonnées géographiques du lieu de création du document multimédia. Cependant, une application mobile de création de documents multimédias n'a pas toujours un accès direct à ces informations spatiales. Par exemple, l'absence d'un GPS intégré dans le dispositif mobile, ou bien son fonctionnement incorrect, empêche l'acquisition des coordonnées géographiques et, par conséquent, l'utilisation de ces trois services d'interprétation. Afin de contourner cet obstacle, la plate-forme *ContextAnnotator* propose trois services pour offrir des coordonnées géographiques aux services mobiles à partir d'autres informations spatiales. Ces services appartiennent à la catégorie de service *Reverse Geocoding* et sont illustrés dans la Figure 67.

Le service CellIDToCoords, par exemple, transforme l'identification d'un relais cellulaire GSM en coordonnées géographiques (les coordonnées du centre de la cellule). Pour accomplir cette tâche, le service CellIDToCoords utilise le registre Telin[90] qui contient des coordonnées géographiques associées à des identificateurs de plus de 120 000 cellules distribuées dans 59 pays[91]. L'ontologie produite par le service CellIDToCoords contient une instance du concept *Place* avec les coordonnées du centre de la cellule et une géométrie associée. La géométrie est en fait un cercle représentant le relais cellulaire. Cette information peut être utile car le rayon de la cellule GSM peut être de plus d'un kilomètre, et produire ainsi des coordonnées très distantes du lieu de création du document.

Le service AddressToCoords et MyPlacesToCoords génèrent des coordonnées géographiques, respectivement à partir d'une adresse postale et d'une description d'un lieu personnel. AddressToCoords utilise un service Google[92] pour la transformation d'une adresse en coordonnées. MyPlacesToCoords exploite la base de données de lieu personnel. Ce service indique les coordonnées géographiques du centre géométrique du lieu et le polygone représentant le lieu personnel.

Le dernier service de la dimension spatiale est, en réalité, une composition de services. Cet enchaînement, nommé *AllSpatialElements*, tente de générer le maximum de métadonnées

[90] http://cellid.telin.nl/wasp/jsp/CellStats.jsp

[91] Statistiques de Septembre 2009.

[92] http://code.google.com/apis/maps/documentation/services.html#ReverseGeocoding

spatiales que la plate-forme peut produire. Dans un premier temps, ce service teste la présence des coordonnées géographiques. En ayant ces informations, ce service déclenche en séquence les services d'enrichissement de la catégorie *Geocoding*. En cas d'absence des coordonnées géographiques, *AllSpatialElements* cherche, dans l'ontologie, une autre information spatiale associée à la localisation de création du document (par exemple, une adresse, un nom de lieu). Ce service utilise ensuite un des services de la catégorie *Reverse Geoconding* pour produire les coordonnées géographiques. L'obtention des coordonnées géographiques permet ensuite d'appeler les services d'enrichissement spatiaux.

La Figure 69 illustre un exemple d'exécution du service *AllSpatialElements*. Une requête est envoyée au module *AnnotationBroker* (étape I) par une application mobile de partage de contenu. Cette requête contient les métadonnées initiales et la demande de production de nouvelles métadonnées concernant les éléments spatiaux du contexte (le client du service informe qu'il souhaite la génération de tous les sous-concepts de *SpatialElement*). Dans le registre de service, le concept *SpatialElement* est associé au service *AllSpatialElements*. Par conséquent, le module *AnnotationBroker* met en œuvre la composition de service *AllSpatialElements* et informe l'annotation initiale envoyée par l'application mobile (étape II).

Les métadonnées initiales ne contiennent pas de coordonnées géographiques associées au lieu de création du document multimédia. Uniquement, le nom de ce lieu est informé (« mon travail ») sur l'ontologie. La composition de service commence ainsi par l'appel du service MyPlaceToCoords. Les coordonnées produites par ce service sont ensuite utilisés pour dériver d'autres lieux personnels (« Bu Sciences »), l'adresse (« Saint Martin d'Hères, France ») et les points d'intérêt à proximité (« Université de Grenoble-I »). À chaque appel d'un service, l'ontologie résultat du service précédent est passée comme paramètre d'entrée du service suivant. À la fin du processus, l'ontologie produite par le dernier service contient toutes les métadonnées générées dans le processus d'enrichissement. L'ontologie est envoyée au module *AnnotationBroker* (étape III). Celui-ci la repasse au client du service (étape IV).

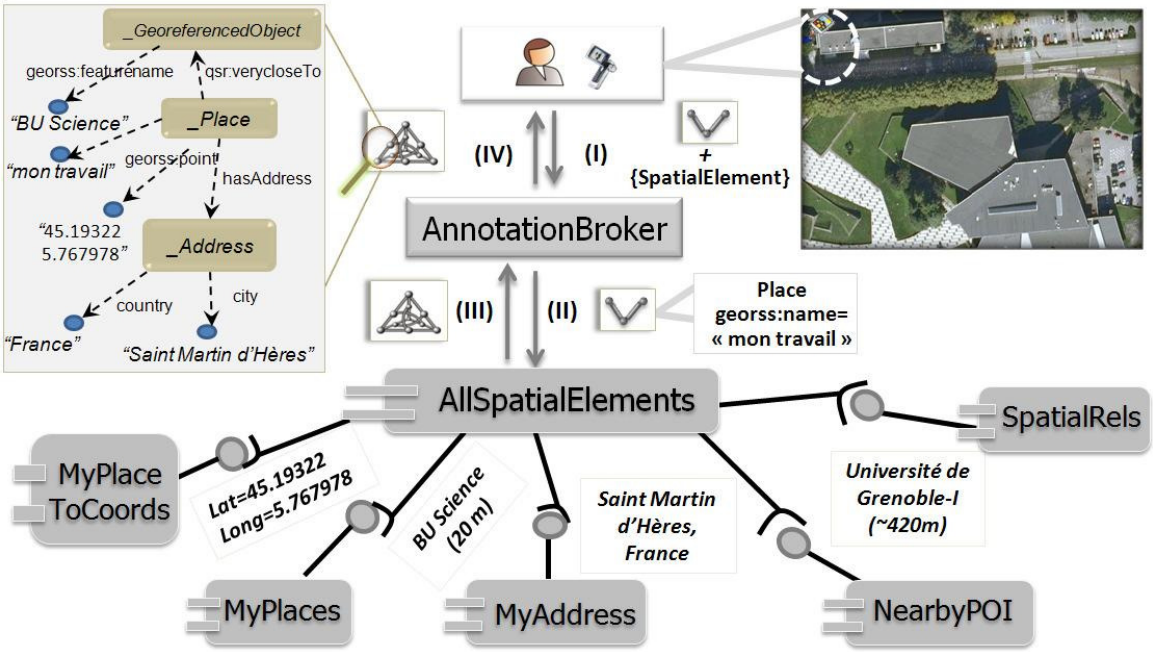


Figure 69 - Un exemple d'utilisation du service AllSpatialElements.

6.3.2.2 Dimension temporelle du contexte

Le service d'interprétation le plus simple de *ContextAnnotator* est celui qui enrichit la dimension temporelle du contexte. Ce service a besoin uniquement de la description de la date et de l'heure de création du document. À partir des ces données, il sépare les divers composants temporels tels que le jour de la semaine, le mois et l'année. Une valeur décrivant le moment de la journée est également calculée en prenant en compte uniquement l'heure de création du document. Nous considérons quatre moments de la journée: le *matin* (6:00 à 12:00), l'*après-midi* (12:00 à 17:00), le *soir* (17:00 à 21:00) et la *nuit* (21:00 à 6:00). Ces tranches horaires peuvent être modifiées dans les configurations du service. Tous les attributs calculés sont stockés dans la dimension temporelle de *ContextMultimedia* en utilisant les propriétés de données (*DataProperties*) telles que *time:dayofweek*, *time:month, time:year* et *time:hasDurationDescription.*

6.3.2.3 Dimension spatio-temporelle du contexte

ContextAnnotator fournit trois services pour la génération de métadonnées spatio-temporelles. Ces services génèrent des métadonnées sur l'état de l'environnement physique lors de la création du document, en utilisant plusieurs services Web gratuits. Pour le renseignement sur les conditions climatiques, par exemple, le service NearByWeather utilise les services Web *Weather Underground*[93] et *Geonames Weather*[94]. Nous optons pour l'usage de deux services afin d'avoir une alternative lorsqu'un des services tombe en panne (relativement fréquent sur les services du site *Geonames…*) ou lorsqu'un des services ne dispose pas de l'information concernant le lieu de création du document[95]. Ces deux services Web offrent des informations relatives à la température, la vitesse du vent, l'état des nuages, l'humidité et la météo (pluie, soleil, orages, etc.).

ContextAnnotator dispose également d'un service de calcul de la période de la journée par rapport au coucher/au lever du soleil. Ce service a d'abord besoin de découvrir les heures de lever et de coucher du soleil du lieu et date de création du document. Notre service d'interprétation utilise ainsi le service Web *Sunrise and Sunset Times*[96]. À partir de la date et des coordonnées géographiques, ce service renvoie l'heure du lever et du coucher du soleil à cette date précise. Avec le résultat fourni par EarthTools, notre service d'interprétation détermine en quelle période le document a été créé. Comme pour le moment de la journée, nous divisons la journée en tranches horaires : « Lever du soleil » (une demi-heure avant le lever du soleil à une demi-heure après), « Jour » (une demi-heure après le lever du soleil jusqu'à une demi-heure avant le coucher du soleil), « Coucher du soleil » (une demi-heure avant le coucher du soleil à une demi-heure après), et « Nuit » (une demi-heure après le coucher du soleil jusqu'à une demi-heure avant le lever du soleil). Ces tranches horaires peuvent être également modifiées dans les configurations du service.

Le troisième service spatio-temporel détermine la saison en utilisant la date et les coordonnées géographiques. Nous considérons la définition météorologique des saisons qui est basée sur la notion de périodes chaudes (été) et froides (l'hiver) et exprimées en intervalles de dates. Pour établir les périodes de chacune des quatre saisons[97], nous adoptons les dates du

[93] www.weatherunderground.com/

[94] http://ws.geonames.org/findNearByWeather

[95] Ce qui est fréquent sur *Weather Underground* pour les villes non capitales de l'Europe..

[96] http://www.earthtools.org/

[97] D'autres définitions de dates existent pour le début et la fin des saisons telle que la définition basée strictement dans l'observation astronomique dont le solstice d'été (le 21 juin dans l'hémisphère nord) devrait marquer le milieu de l'été et non le début.

calendrier français (21 juin, 22 septembre, 21 décembre, 20 mars). Nous appliquons un simple calcul des dates, combiné à la découverte de l'hémisphère où le document a été crée. Par exemple, l'hiver est adopté comme métadonnée lorsque deux conditions sont valides : i) *21 juin <= date <22 septembre* et ii) *latitude <= 0°*.

Tous ces attributs dérivés sont stockés dans les propriétés de données (*DataProperties*) du concept *Physical_Environnement* telles que *season, light status, windspeed, temperature_value. ContextAnnotator* dispose également d'une composition de ces trois services d'enrichissement de métadonnées spatio-temporelles. Au démarrage de ce service, il teste la présence des coordonnées géographiques dans l'ontologie. En cas d'absence de ces informations, les services spatiaux de la catégorie *Reverse Geocoding* sont exploités avant d'appeler les services associés à dimension spatio-temporelle du contexte.

6.3.2.4 Métadonnées sur le contenu

Des méthodes de classification de contenu peuvent être également intégrées à la chaîne d'enrichissement de métadonnées en utilisant le modèle de service *MetadataFromContent*. Afin d'illustrer cette intégration, *ContextAnnotator* propose un service de localisation/détection de visages sur des images. Ce service de localisation de visages nécessite l'accès au document multimédia personnel. Nous avons utilisé la méthode de classificateurs du type Haar pour la détection de visages, en raison de sa vitesse d'exécution [Wilson *et al.*, 2006]. Cette méthode de détection d'objets, visages inclus, proposé par Viola *et al.* [Viola *et al.*, 2001] repose sur un enchaînement en cascade de classificateurs qui utilisent l'algorithme AdaBoost [Freund *et al.*, 1999]. Cet algorithme d'apprentissage agressif construit un classificateur fort à partir d'un ensemble de classificateurs faibles. Dans la proposition de Viola *et al.*, les classificateurs AdaBoost n'utilisent pas les valeurs d'intensité d'un pixel. Ces classificateurs sont basés sur la variance de contraste des régions de pixels qui permettent la détermination de régions sombres et claires d'une image. Ces régions de pixels sont dénommées les *Haar-features*. Ensuite, un processus d'entrainement est réalisé pour déterminer si chacune des régions ne contient pas un visage (deux ensembles de régions exemple sont fournis pour l'entrainement). Le résultat du processus est un arbre de décision que peut être utilisé pour la détection de visages sur de nouvelles images. L'utilisation des classificateurs Haar est censée apporter une exactitude de 95%, cependant le nombre de faux positifs peut attendre jusqu'à 25% [Wilson *et al.*, 2006].

Nous avons implémenté ce service d'interprétation à l'aide de deux bibliothèques : OpenCV(*Open Source Computer Vision*)[98] et FAINT (*Face Annotation INTerface*)[99]. OpenCV fournit l'implémentation de la méthode. La bibliothèque FAINT offre un accès JNI (*Java Native Interface*) à OpenCV et un arbre de décision déjà entrainé. La méthode se montre rapide avec une détection en moins de 5 secondes par image testée sur un ordinateur portable[100]. Cependant, les résultats sont moins bons que ceux annoncés par les auteurs. Ceci est dû probablement à la qualité des photos prises par des téléphones mobiles qui est encore nettement inférieure à celle des appareils photos numériques classiques. La Figure 70 illustre des exemples de détection de visages sur quatre images prises par un téléphone mobile Nokia N95.

Les régions détectées par le service sont décrites par l'ontologie *ContextMultimedia* en utilisant les concepts provenant de l'ontologie Mpeg7. Chaque visage est une instance du concept StillRegion comme dans l'exemple de la Figure 53. En raison de la détection de faux

[98] http://sourceforge.net/projects/opencvlibrary/

[99] http://faint.sourceforge.net

[100] **Dell Latitude E6500 -** processeur Intel® CoreTM 2 Duo, mémoire DDR2 bicanal5 offrant 4 Go et 800 MHz de bande passante mémoire.

positifs, nous avons ajouté une propriété pour indiquer l'état de validation des métadonnées (*is Validated* = *false*). Un outil d'annotation pourra utiliser les résultats du service pour le filtrage d'images contenant des personnes, et proposer à l'utilisateur d'identifier le nom de personnes présentes sur l'image.

D'autres classificateurs sont disponibles sur Open CV (par exemple, pour la détection de photos prises à l'intérieur et à l'extérieur). Ces classificateurs peuvent être également intégrés dans *ContextAnnotator* pour augmenter le nombre des métadonnées concernant le contenu de documents multimédias.

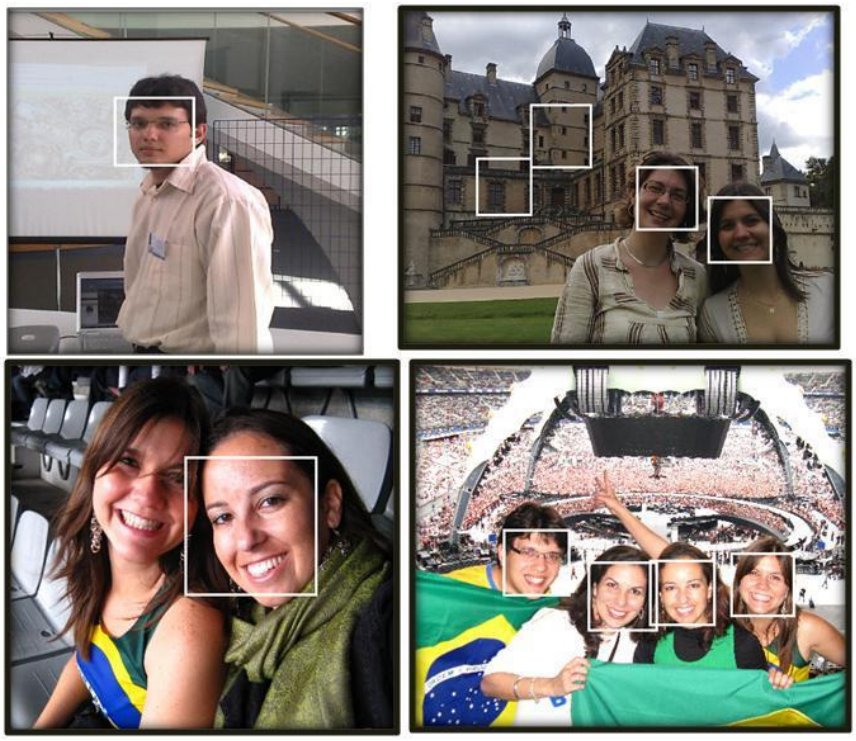


Figure 70 - Exemples de résultats du service de détection de visages.

6.3.3 Processus d'inférence de métadonnées

L'ontologie ContextMultimedia offre un riche vocabulaire pour stocker les métadonnées générées par les services d'interprétation de ContextAnnotator. Cette ontologie est également perçue comme une base de connaissances associée aux documents multimédias dont les mécanismes d'inférence peuvent être appliqués pour inférer des informations (principalement, des relations entre les individus) qui étaient, auparavant, implicites dans l'ontologie.

Dans cette section, nous décrivons deux services de ContextAnnotator qui exploitent des mécanismes d'inférence pour l'enrichissement des métadonnées. Le premier service infère des métadonnées concernant le contexte social de création du document multimédia. Le deuxième service calcule les relations spatiales qualitatives entre les objets géoréférencées et le lieu de création du document multimédia.

6.3.3.1 Service pour l'inférence du contexte social

L'ontologie ContextMultimedia offre des concepts pour la représentation des personnes présentes lors de la création d'un document multimédia (le concept *Person* associé au concept *foaf:Person* de l'ontologie FOAF). L'ontologie ContextMultimedia introduit également des propriétés (par exemple, *wifeOf*) décrivant les relations entre ces personnes et le créateur du

contenu (concept *Creator*). Pour chaque document multimédia, toutes ces informations peuvent être remplies manuellement par les utilisateurs à l'aide, par exemple, d'un outil d'annotation tel que le logiciel OntoAlbum (Chapitre 3). Cependant, un des objectifs de notre approche est précisément de réduire cet effort d'annotation des utilisateurs en proposant de générer automatiquement ces informations et de transformer, par conséquent, la tâche d'annotation en une tâche de validation de métadonnées. Ainsi, nous proposons un service pour inférer automatiquement le contexte social des documents multimédias.

Nous supposons qu'un système de gestion de multimédias décrit le réseau social d'une personne en utilisant l'extension de FOAF proposée par ContextMultimedia (section 6.2.2). De nombreux systèmes multimédias Web 2.0 (Flickr et Picasa Web) offrent déjà des interfaces permettant la description des relations de connaissances (utilisateur A connait utilisateur B). Ces interfaces doivent être étendues afin de permettre la qualification de ces relations entre les utilisateurs (par exemple, meilleur amie, mère, cousin, etc.). Pour le fonctionnement de notre service, ce réseau social doit être représenté en utilisant ContextMultimedia. Dans cette représentation, chaque personne possède un fichier (une ontologie) représentant son profil qui est identifié par une URI. La Figure 71 illustre une représentation de la superposition des réseaux sociaux de plusieurs personnes.

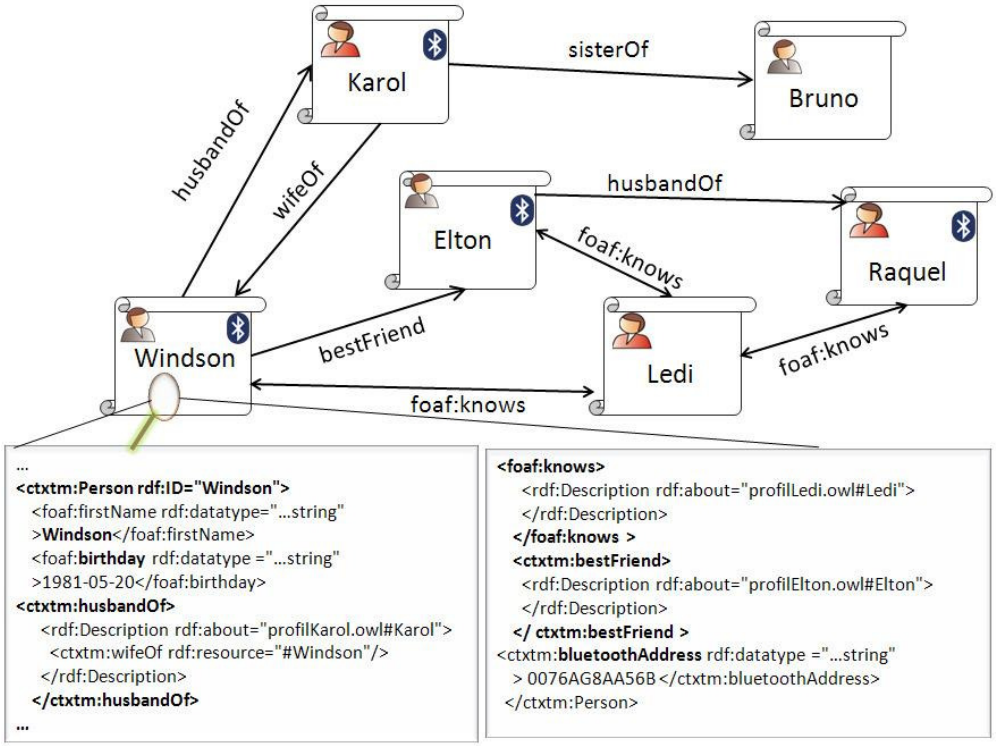


Figure 71 - Exemple de description d'un réseau social.

L'ontologie d'un utilisateur contient des informations personnelles le concernant et les relations entre son profil et les profils d'autres personnes identifiées par des URI. En regardant exclusivement le profil de l'utilisateur Windson sur la Figure 71, nous pouvons connaître les personnes appartenant à son réseau social (Karol, Elton, Ledi), néanmoins nous n'avons pas d'accès à davantage d'informations sur eux. Par exemple, restent inconnus la date d'anniversaire de son épouse et le réseau social de celle-ci. Ces informations sont découvertes en navigant à travers les relations de connaissances du profil de Windson (dans le cas de l'exemple : la relation *husbandOf*) qui indiquent l'emplacement du profil de la personne appartenant à son réseau social. Ainsi, le réseau social présente une représentation distribuée. Un profil peut être stocké et maintenu par un autre système de gestion de multimédias.

Pour garantir le succès de notre processus d'inférence du contexte social, le système de gestion de multimédias doit permettre à un utilisateur d'associer les adresses Bluetooth de dispositifs mobiles à son profil personnel et à des profils d'autres personnes. Ceci peut être fait automatiquement par une application mobile qui capture l'adresse Bluetooth de son dispositif hôte et l'envoi au service de gestion de profils d'un système d'annotation.

Le principe du processus d'inférence du contexte social est de découvrir si les dispositifs Bluetooth stockés dans le contexte informationnel d'un document multimédia appartiennent à des personnes décrites dans le réseau social du créateur de ce document. De plus, ce processus d'inférence doit établir les relations sociales existantes entre ces personnes et le créateur du contenu.

Malheureusement, la représentation de ce processus ne peut pas être faite simplement en utilisant le langage OWL, car les mécanismes d'inférence de ce langage sont basés sur l'établissement de la subsomption et sont ici peu adaptés. Nous optons pour l'utilisation du langage SWRL (*Semantic Web Rule Language*) qui combine OWL DL et OWL Lite avec le langage RuleML (*Rule Markup Language*). SWRL est un langage recommandé par le W3C pour enrichir la sémantique d'une ontologie définie en OWL. SWRL, contrairement à OWL, permet de manipuler des instances par des variables (?x, ?y, etc.). En fait, les axiomes du langage OWL sont étendus à l'aide de clauses Horn réduites aux prédicats unaires et binaires. Ce langage offre ainsi la possibilité de décrire des règles d'inférences qui peuvent être associées aux concepts, individus, et propriétés d'une ontologie OWL. Une règle d'inférence est un jugement contenant un ensemble de hypothèses (ou prémisses) suivi d'une conclusion. **Si** toutes les hypothèses sont satisfaites **alors** la conclusion est également satisfaite pour ce domaine logique. Le détachement (*modus ponens*[101]) est un exemple de raisonnement qui peut être mis en œuvre en utilisant des règles d'inférence :

$$p \vdash q,\ \vdash p \rightarrow \vdash q\ ;$$

dont $p \vdash q$ est une règle d'inférence et $\vdash p$ un axiome.

Sur SWRL, les hypothèses sont nommées les *antécédents*, et la conclusion, le *conséquent*. Une règle SWRL est construite suivant le schéma : *antécédent(s)* → *conséquent*. La syntaxe de SWRL peut être résumée à :

$$C(x) \wedge P(x, y) \wedge \ldots \rightarrow G(x, z)$$

C : prédicat unaire (un concept de l'ontologie)

P, G : prédicat binaire (propriétés de l'ontologie)

x, y, z : des variables, des instances ou de littéraux (nombres, chaine de caractères, etc.)

Le Tableau 9 présente un des règles SWRL que nous utilisons pour dériver le contexte social d'un document multimédia. Cette règle établit la présence d'une personne sur le lieu de création du document. Une personne appartient au contexte social d'un document (*ctxt :hasContextElement(?ctxt, ?person)*) si cette personne est dans le réseau social (*foaf :Knows(?creator, ?person)*) de créateur du document (*cxtmm :Creator(?creator)*). Cette personne doit également être le propriétaire d'un des dispositifs Bluetooth décrits dans le contexte informationnel de l'ontologie (*hasContextElement(? ctxt, ?btDV)*). Cette règle est appliquée sur les instances de l'ontologie ContextMultimedia dont les individus sont considérés comme des « faits ». Nous tenons à souligner que les personnes possédant une relation héritant de *foaf:knows* (wifeOf, husbandOf, etc.) et qui étaient présentes lors de la création du document sont également répertoriées par la règle du Tableau 9. En fait, le moteur d'inférence suppose que si une des relations héritant de *foaf:knows* existe entre le créateur du document et une personne, alors la relation *foaf:knows* est aussi satisfaite.

[101] *Modus ponens* est une méthode raisonnement reposant sur l'implication. Elle consiste à affirmer une implication (« si A alors B ») et à poser ensuite l'antécédent (« A est *vrai* ») pour en déduire le conséquent (« donc B »)

Tableau 9- Règle SWRL pour inférer la présence des amis du créateur du document.

```
cxtmm:BTDevice(?btDV) ∧ cxtmm:Creator(?creator) ∧
ctxt:Context(?ctxt) ∧ cxtmm:Person(?person) ∧
foaf:knows(?creator, ?person) ∧
ctxt:hasContextElement(?ctxt, ?btDV) ^
cxtmm:bluetoothaddress(?person, ?btadd) ∧
cxtmm:bluetoothaddress(?btDV, ?btadd2) ∧

swrlb:equal(?btadd, ?btadd2)

→ ctxt:hasContextElement(?ctxt, ?person)
```

Identification des variables

Découverte des adresses Bluetooth

Comparaison des adresses Bluetooth

Contrairement à l'approche proposée par Monaghan *et al.* [Monaghan *et al.*, 2006] décrite dans le Chapitre 3, nous ne retenons comme métadonnées que les personnes présentes lors de la création du document qui possèdent effectivement une relation sociale avec le créateur du contenu. Ce choix réduit notre espace de recherche, car nous découvrons les personnes uniquement dans le réseau social du créateur du document. Ainsi, nous n'avons pas besoin de chercher dans une base de données « universelle » contenant toutes les adresses Bluetooth qui serait plus difficile à construire. La Figure 72 illustre le fonctionnement de notre service d'inférence du contexte social.

L'ontologie d'entrée doit contenir l'identificateur du créateur et le contexte informationnel. Un module du service récupère le profil du créateur du document et insère dans l'ontologie des instances de *Person* représentant les personnes de réseau social du créateur. Cette ontologie augmentée est introduite dans le moteur d'inférence avec les règles d'inférence. À la fin du processus d'inférence, les instances du concept Person qui n'appartiennent pas au contexte social (c'est-à-dire, celles pour lesquelles la relation *hasContextElement(?ctxt, ?person)* n'est pas satisfaite) sont exclues de l'ontologie afin de réduire sa taille et de ne conserver que les métadonnées vraiment liées au document multimédia. La Figure 72 illustre un exemple de ce processus. Les personnes L et E sont introduites dans l'ontologie au début du processus et sont retirées lors de la dernière étape, car le contexte C1 ne contient pas ces personnes. Cependant la personne K est conservée dans l'ontologie, puisque la relation *hasContextElement(Context_C1, K),* représentée par une flèche pointillée, a été inférée.

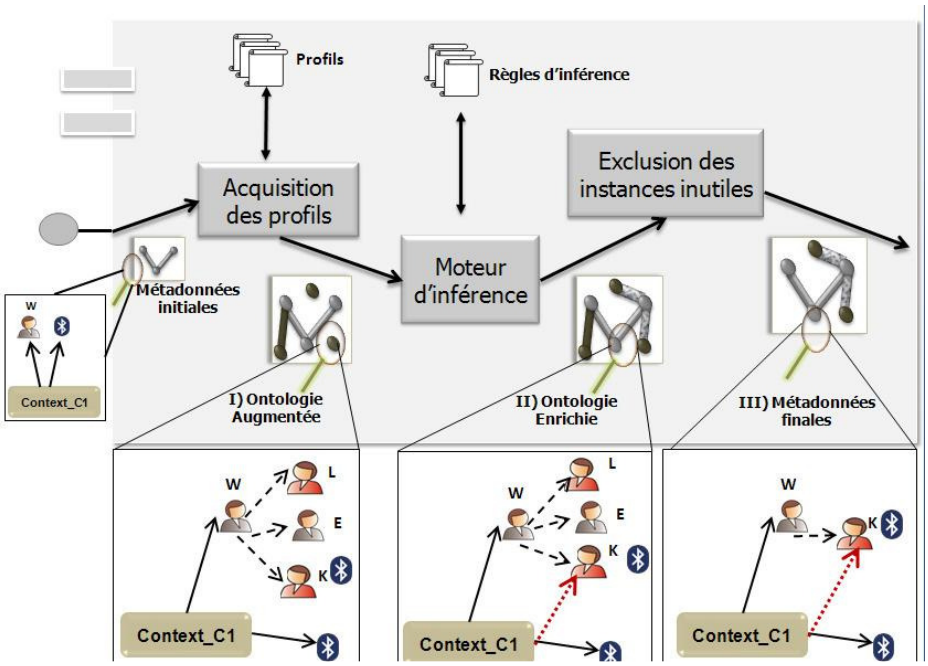


Figure 72 – Processus d'inférence du contexte social.

D'autres règles peuvent être introduites dans ce processus pour inférer des métadonnées relatives au document. Ces règles peuvent faire référence à tous les concepts de l'ontologie et leurs résultats peuvent ajouter de nouvelles relations entre les individus décrits dans les métadonnées. Les règles SWRL sont déclaratives, ce qui facilite l'extension du processus d'inférence lui-même. De plus, leur ordre d'exécution n'altère pas les résultats du processus d'inférence, puisque le moteur d'inférence exécute les règles en boucle jusqu'à qu'aucune règle ne soit satisfaite. Les Tableau 10, Tableau 11 et Tableau 12 illustrent trois autres règles.

Tableau 10- Règle SWRL pour attribuer un mot-clé au document si sa date de création est le jour de l'anniversaire du créateur.

```
Tag(inferredTag) ∧ Creator(?creator) ∧ Context(?ctxt) ∧
Personal_Multimedia(?doc) ∧ Temporal_Element(?inst) ∧      } Identification des variables
hasContextElement(?ctxt, ?creator) ∧
hasContextElement(?ctxt, ?inst) ∧                          } Découverte de l'instant de création
hasContext(?doc, ?ctxt) ∧
time:hasDurationDescription(?inst, ?duration) ∧
foaf:birthday(?creator, ?birhtdayCreator) ∧                } Découverte de l'anniversaire du créateur

time:days(?duration, ?durationInDays) ∧
time:months(?duration, ?durationInMonths) ∧
time:days(?birthdayCreator, ?birthInDays) ∧                } Transformation en jours et mois
time:months(?birthdayCreator, ?birthInMonths) ∧

swrlb:equal(?birthInDays, ?durationInDays) ∧               } Comparaison des dates
swrlb:equal(?birthInMonths, ?instantInMonths)

→ hasTextualAnnotation(?doc, birthday) ∧
hastag(inferredTag, "My Birhtday")
```

La première règle illustre l'ajout d'une annotation textuelle signalant que la date de création du document coïncide avec le jour d'anniversaire du créateur du document. Avant

l'exécution de cette règle un mot-clé vide est insérer dans l'ontologie augmentée (instance *inferredTag*). Le conséquent de cette règle produit exactement une valeur pour ce mot-clé (« My birthday ») et l'associe au document multimédia (*hasTextualAnnotation(?doc, inferredTag)*). Le mois et l'année de a date d'anniversaire sont comparés à ces mêmes attributs de l'instant de création du document multimédia. Une variation de cette règle est illustrée dans le Tableau 11. Elle attribut un mot-clé au document si la date de création correspond à l'anniversaire d'une des personnes présentes qui ont été inférées par la règle du Tableau 9. La valeur du mot-clé est composée du prénom de la personne (*foaf : firstName*) suivi de « 's Birthday ».

Le Tableau 12 illustre une réglé d'inférence qui ajoute dans le contexte social de l'ontologie les personnes présentes qui sont amis des amis du créateur du document. Pour la correcte exécution de cette règle, l'ontologie augmentée au commencement du processus d'inférence doit contenir en plus les personnes appartenant au réseau social de chaque ami du créateur du document. Les personnes inférées sont ajoutés au contexte (*hasContextElement(?ctxt, ?person)),* néanmoins il n'existe pas dans l'ontologie de relations entre ces personnes et le créateur du document.

Tableau 11- Règle SWRL pour attribuer un mot-clé au document si sa date de création est le jour de l'anniversaire d'une personne présente lors de la création du document.

```
Tag(inferredTag) ∧ Creator(?creator) ∧ Context(?ctxt) ∧ Person(?person)
Personal_Multimedia(?doc) ∧ Temporal_Element(?inst) ∧swrlb:notEqual(?creator, ?person)
hasContextElement(?ctxt, ?creator) ∧ hasContextElement(?ctxt, ?person)
hasContextElement(?ctxt, ?inst) ∧                    } Découverte de l'instant de création
hasContext(?doc, ?ctxt) ∧
time:hasDurationDescription(?inst, ?duration) ∧      } Découverte de l'anniversaire
foaf:birthday(?person, ?birhtdayPerson) ∧            } de la personne
time:days(?duration, ?durationInDays) ∧              ]
time:months(?duration, ?durationInMonths) ∧          ] Transformation en jours et mois
time:days(?birthdayPerson, ?birthInDays) ∧           ]
time:months(?birthdayPerson, ?birthInMonths) ∧       ]
swrlb:equal(?birthInDays, ?durationInDays) ∧         ] Comparaison des dates
swrlb:equal(?birthInMonths, ?durationInMonths)       ]
foaf:firstName(?person,?firstname)∧                  } Géneration de la valeur
swrlb:stringConcat(?newTag, ?firstname, "'s birhtday") } du mot-clé

→ hasTextualAnnotation(?doc, inferredTag)
∧hastag (birthday, ?newTag)
```

Toutes les informations dérivées lors de ce processus d'inférence sont stockées dans l'ontologie résultante du processus et peuvent être exploitées pour la gestion de documents multimédias. Par exemple, les noms de personnes inférées comme étant présentes lors de la prise d'une photo peuvent être suggérées pour identifier les visages détectés par le service d'interprétation du contenu (section 6.3.2.4).

BTDevice(?btDV) ∧ Creator(?creator) ∧ Context(?ctxt) ∧ Person(?person) ∧ Person(?person2) ∧	Identification des variables
foaf:knows(?creator, ?person) ∧ foaf:knows(?person, ?person2) ∧	Découverte des amis des amis
hasContextElement(?ctxt, ?btDV) ∧ bluetoothaddress(?person2, ?btadd) ∧ bluetoothaddress(?btDV, ?btadd2) ∧	Découverte des adresses Bluetooth
swrlb:equal(?btadd, ?btadd2)	Comparaison des adresses Bluetooth
→ **ctxt:hasContextElement(?ctxt, ?person)**	

Tableau 12 - Règle SWRL pour inférer la présence des amis des amis du créateur du document.

6.3.3.2 Service pour l'inférence de relations spatiales qualitatives

Notre proposition d'ontologie d'annotation comporte la représentation des relations spatiales qualitatives entre le lieu de création (concept *Place*) du document et les objets géoréférencés stockés dans les métadonnées.

Selon Grütter *et al.* [Grütter *et al.*, 2007], ces relations spatiales ne peuvent pas être calculées en utilisant les méthodes de raisonnement classiques de OWL ni les règles d'inférence SWRL, en raison de la nature très algébrique de ces calculs dérivés des méthodes RCC-8. Cependant, un système hybride peut être construit pour calculer ces informations. Une base de données géographiques et un raisonneur spatial peuvent être utilisés pour calculer ces informations et, ensuite, les représenter sur l'ontologie.

Pour inférer les relations topologiques et de distance, nous utilisons des méthodes d'une base de données géographique. Une fois que les objets géoréférencés et les lieux personnels sont introduits dans l'ontologie, notre service d'inférence spatiale peut être activé. Les informations disponibles sur l'ontologie sont représentées sur une base de données spatiale et les fonctions de mesure (telles que la distance entre deux géométries, l'intersection, l'inclusion, etc.) sont exploitées pour le calcul de ces relations spatiales qualitatives. Ce sont ces méthodes, par exemple, qui permettent d'inférer qu'un document multimédia a été créé à l'intérieur ou à l'extérieur dans lieu personnel.

Pour le calcul de relations de distance, nous établissons, par défaut, que les objets géoréférencés et les lieux personnels se trouvant à moins de 300 mètres du lieu de création possèdent la relation très proche (*qsr :veryclose*) alors que ceux se trouvant entre 300 mètres et 1km possèdent la relation proche (*qsr:close*). Ces paramètres sont réglables via la configuration du service. L'exemple de la Figure 69 illustre le calcul de cette relation de distance, la composition de services *AllSpatialElements* invoque, à la fin du processus d'enrichissement, le service d'inférence spatiale (*SpatialRels*). La relation *qsr:veryclose* est ajoutée entre le lieu de création du document (une instance du concept *Place*) et le lieu personnel « BU Sciences », car leur distance euclidienne est de 20 mètres.

Pour le calcul des relations de direction, nous utilisons le raisonneur spatial développé dans notre équipe, appelé ONTOAST (Miron *et al.*, 2007), qui peut être utilisé pour exploiter les informations décrites dans les ontologies OWL grâce à sa compatibilité avec le langage OWL 2 DL. Construit comme extension du système de Représentation des Connaissances par Objets AROM[102], ONTOAST est à la fois un environnement de modélisation d'ontologies

[102] http://www.inrialpes.fr/romans/arom/

spatio-temporelles et un système d'inférence, capable de raisonner sur des connaissances spatiales, temporelles et thématiques. Le système ONTOAST est capable d'interpréter les descriptions spatiales quantitatives et qualitatives réalisées conformément à des ontologies telles que GeoRSS-Simple que nous utilisons également sur ContextMultimedia. Ces descriptions sont traduites vers un formalisme propre à ONTOAST afin qu'elles puissent être exploitées dans des raisonnements. Des règles écrites en ONTOAST permettent l'établissement des relations de direction entre les objets géoréférencés et le lieu de création du document (telles que « au nord de », « au sud de »). À la suite de cette étape d'inférence, les relations spatiales qualitatives résultantes sont traduites en OWL afin qu'elles puissent être mises à la disposition d'autres services de ContextAnnotator.

6.4 Conclusion

Le premier pas pour assurer une meilleure gestion de documents multimédias personnels est de fournir des métadonnées associées à ces documents. Dans ce chapitre, nous avons présenté une ontologie pour la représentation des métadonnées concernant le contexte de création d'un document et son contenu. Nous avons également élaboré une plate-forme qui génère la majorité de ces métadonnées, sans exiger en contrepartie un grand effort de la part de l'utilisateur.

L'ontologie ContextMultimedia répond au besoin d'un vocabulaire formel et extensif pour représenter les métadonnées d'un document multimédia personnel. L'ontologie que nous proposons offre des concepts et des relations pour la description de trois catégories de métadonnées :

- **Les descripteurs textuels du document.** Ces concepts permettent l'introduction de mots-clés, et de descriptions textuelles qui font référence au document multimédia.
- **Les descripteurs de son contexte de création.** Nous offrons des concepts et des relations pour décrire des métadonnées du contexte selon cinq dimensions qui sont définies sur l'ontologie ContextTop : spatiale, temporelle, sociale, informationnelle et spatio-temporelle. Nous définissons un ample vocabulaire englobant toutes les métadonnées contextuelles proposées par les outils de gestion de multimédias décrites dans l'état de l'art. Ce vocabulaire, fourni par ContextMultimedia, repose sur l'extension et la réutilisation d'un ensemble d'ontologies qui sont déjà largement utilisées par des approches du Web Sémantique pour la description des aspects spatiaux (GeoRSS), temporels (OWL-Time), et sociaux (FOAF) .
- **Les descripteurs du contenu du document**. Ces concepts et les relations dérivés d'une ontologie représentant le langage MPEG7 en OWL offrent la description de propriétés de bas niveau du contenu d'un document multimédia. Ces concepts peuvent également être reliés à d'autres métadonnées de l'ontologie ContextMultimedia, par exemple, pour identifier une personne sur un segment du document multimédia.

La plate-forme ContextAnnotator est le module de CoMMeDiA qui assure l'enrichissement des métadonnées initialement capturées par le dispositif mobile lors de création d'un document multimédia. Conçue principalement pour un usage Web, ContextAnnotator offre, aux services mobiles de partage de contenu et aux outils de gestion de multimédias, une liste de services d'enrichissement de métadonnées qui reposent sur trois mécanismes :

- **L'accès aux sources de données du Web**. Les services d'interprétation de ContextAnnotator utilisent des gazetteers, des services de météorologie, et le Wikipedia pour augmenter les métadonnées concernant le contexte de création d'un document.
- **L'accès aux données stockées et gérées par la plate-forme elle-même**. La plate-forme possède une base de données contenant les lieux personnels décrits par les utilisateurs, leurs préférences en termes de configuration de services d'enrichissement, et leurs profils qui incluent leur réseau social. Ces informations sont également exploitées par les services d'interprétation et d'inférence des métadonnées.
- **L'exécution de processus d'inférence et d'analyse du contenu du document multimédia**. ContextAnnotator supporte l'exécution de règles d'inférence SWRL pour la découverte de métadonnées associées au contexte social du document multimédia. Un service de ContextAnnotator illustre également comment les méthodes d'analyse de contenu peuvent être intégrées sur la plate-forme. Ce service offre la détection de visages sur des images.

La plate-forme ContextAnnotator est conçue de manière à être flexible, car un client de service d'enrichissement peut choisir quelles métadonnées il souhaite inférer ou acquérir. Un client de services de ContextAnnotator a l'option d'invoquer un seul service d'enrichissement, ou toute une composition de services. L'usage de services Web et d'un modèle de service dont l'ontologie est à la fois le paramètre d'entrée et de sortie, garantit également l'extensibilité de la plate-forme, puisque les compositions de services peuvent être étendues plus facilement.

À l'aide de ContextMultimedia et de ContextAnnotator, nous proposons la représentation et l'acquisition automatique d'une large liste de métadonnées. Dans le chapitre 8, nous présentons des mécanismes d'exploitation de ces métadonnées pour l'organisation, le partage et la recherche textuelle de documents multimédias créés par des dispositifs mobiles.

7 DEPLOIEMENT ADAPTATIF ET ACQUISITION DE CONTEXTE

Tout système sensible au contexte, y compris ceux destinés à la gestion de documents multimédias, reposent sur une capture initiale d'informations contextuelles. Dans nos propositions, le dispositif mobile de l'utilisateur exerce cette activité d'acquisition d'information, car nous faisons l'hypothèse que le dispositif mobile sera, dans un proche avenir, le principal dispositif de production et partage de documents multimédias. Dans l'état de l'art, nous avons relevé, néanmoins, les difficultés du développement et du déploiement d'applications mobiles en raison, principalement, de l'hétérogénéité des dispositifs mobiles, et du support divergent des bibliothèques sur les plates-formes mobiles de programmation.

Dans ce chapitre, nous présentons un ensemble de solutions pour l'adaptation d'applications mobiles et un intergiciel d'acquisition de contexte. À travers ces propositions, nous souhaitons diminuer le besoin de développer différentes versions d'une application mobile sensible au contexte pour chaque couple potentiel (modèle de dispositif, plate-forme) en raison des dissemblances qui peuvent exister en termes de caractéristiques matérielles et logicielles des dispositifs cibles. Les différents stades d'évolution de nos propositions ont été rapportés dans [Viana *et al.*, 2009c] et [Viana *et al.*, 2009d]. Nos propositions ont été également validées dans un projet de collaboration franco-brésilien (FAPESP-CNRS) [Viana *et al.*, 2008f][Bergeret *et al.*, 2009].

Nous présentons d'abord une approche générale d'adaptation applicable sur tout type d'applications mobiles. Nous proposons un mécanisme de déploiement adaptatif d'applications mobiles décrites par un assemblage de composants. Le déploiement nous semble être le moment idéal pour adapter une application mobile, principalement, lors des phases de configuration et d'installation. En fait, on peut penser que les portails d'applications mobiles vont devenir le principal moyen de distribution d'applications sur toutes les plates-formes mobiles. Introduire sur ces portails des mécanismes qui reconnaissent le dispositif de l'utilisateur et qui génèrent automatiquement une version adaptée d'une application est bénéfique à la fois à l'utilisateur, qui n'a plus besoin de choisir manuellement les applications fonctionnant exclusivement sur son modèle de dispositif, et le concepteur d'une application, qui peut définir les mécanismes d'adaptation de cette application conformément aux caractéristiques des dispositifs cibles.

Le principe de notre approche de déploiement adaptatif est de décrire les applications mobiles comme un assemblage de composants qui communiquent enter aux par des services.

Un service, dans notre proposition, représente une fonctionnalité implémentée par un composant et fournie par le biais de son interface. Ainsi, un même service peut avoir plusieurs implémentations, chacune adaptée à un modèle de dispositif mobile spécifique. Un mécanisme de sélection est donc mis en place pour générer la combinaison de composants idéale pour le modèle de dispositif de l'utilisateur. Dans la section 7.1, nous décrivons le modèle de description de ces applications, l'approche de détection des caractéristiques d'un dispositif mobile, une proposition d'infrastructure à composants, et le mécanisme de déploiement.

Dans ce chapitre, nous proposons également un intergiciel d'acquisition de contexte, le *Device-Aware Context toolkit (DevAC)*. Notre objectif avec DevAC est de concevoir un intergiciel d'acquisition de contexte qui s'adapte aux caractéristiques matérielles et logicielles des dispositifs mobiles et aux services sensibles au contexte s'exécutant au dessus de lui (l'intergiciel). Dans l'état de l'art, nous avons pu constater l'inadaptation de la plupart des architectures sensibles au contexte aux environnements d'exécution des dispositifs mobiles en raison de la consommation excessive de ressources et des difficultés de portabilité de leur code. Nous souhaitons avec DevAC limiter ces inconvénients en utilisant deux mécanismes d'adaptation : un déploiement adaptatif et une reconfiguration dynamique des services d'acquisition de contexte. Nous détaillons les caractéristiques de DevAC dans la section 7.2.

À la fin de ce chapitre, nous présentons un mécanisme de déploiement adaptatif d'applications mobiles et sensibles au contexte. Ce mécanisme simultanément l'application mobile et l'intergiciel d'acquisition de contexte.

7.1 Déploiement adaptatif d'applications mobiles

Notre approche de déploiement tente de s'approcher du déploiement idéal d'une application mobile (voir Section 5.5.2). Nous souhaitons principalement automatiser ce processus et offrir des mécanismes d'adaptation des applications aux caractéristiques des dispositifs mobiles. Notre approche est fondée sur cinq principes :

- **Programmation orientées composant (POC) pour le développement des applications mobiles**. Les applications mobiles sont construites à partir d'un assemblage de composants qui communiquent entre eux par des services. POC a pour caractéristique principale la séparation entre l'interface et l'implémentation des composants, qui sont l'unité de distribution et de développement de ces systèmes. L'utilisation des composants, outre la possibilité de réutilisation et la réduction de complexité qu'elle offre, permet d'intégrer des mécanismes d'adaptation (tels que l'échange de composants, le paramétrage) dans toutes les phases de déploiement d'une application.
- **Métadonnées pour la description de dépendances et de directives de déploiement**. Les composants possèdent une description de leurs dépendances d'exécution (version de la machine virtuelle, bibliothèques nécessaires, services attendus d'autres composants, etc.) et de leurs directives de déploiement (les directives décrivent les conditions pour déployer le composant).
- **Méta-applications comme un modèle générateur de versions**. Nous supposons, à l'exemple des auteurs [Ayed *et al.,* 2008] [Donsez, 2006] [Fjellheim, 2006], qu'un service (c'est-à-dire la fonctionnalité implémentée par le composant et fournie par le biais de son interface) peut posséder différentes implémentations (la même interface implémentée par plusieurs composants), et que, chaque implémentation est adaptée à un contexte d'utilisation spécifique (réduit ici au duo dispositif mobile, plate-forme de

programmation). Une méta-application est ainsi un assemblage de services (les interfaces) contenant plusieurs alternatives d'implémentation (les composants). La méta-application décrit le comportement fonctionnel idéal de l'application, et ses alternatives proposent une dégradation de ses fonctionnalités afin de permettre son exécution sur une plus grande quantité de dispositifs mobiles.

- **Détection automatique de dispositifs mobiles**. Pour automatiser le processus de déploiement adaptatif, nous proposons un mécanisme de détection des modèles de dispositifs mobiles reposant sur l'extension du registre de profils WURFL.
- **Génération de versions lors de l'installation**. La méta-application et la description des caractéristiques du dispositif mobile (plate-forme de programmation incluse) sont les paramètres d'entrée d'un processus de génération de versions de l'application mobile qui exploitent ces informations pour produire une version adaptée de l'application lors de son installation. Ce processus peut être vu également comme un générateur de configurations de l'application.

Notre proposition de déploiement adaptatif suit le schéma générique des approches de déploiement adaptatif d'applications mobiles présentées dans l'état de l'art (voir section 5.5.2.2, Figure 43). Cependant, au contraire de ces approches, notre mécanisme de détection de contexte d'utilisation (dispositif mobile et plate-forme de programmation) est mis en œuvre sans l'installation préalable d'un intergiciel au dessus de la plate-forme d'exécution. Ce mécanisme repose sur l'usage d'un navigateur Web pour démarrer le processus d'installation.

La Figure 73 illustre notre processus de déploiement adaptatif d'une application mobile. Le processus d'installation est démarré par l'utilisateur à partir d'un navigateur mobile simple, qui existe déjà sur la plupart des dispositifs mobiles. L'utilisateur accède à un portail d'applications et choisit une application. Le serveur de CoMMediA utilise l'entête de la demande WAP ou HTTP afin d'identifier automatiquement le dispositif comme dans les approches d'adaptation d'applications Web décrites dans l'état de l'art (voir section 4.1.1).

La valeur de l'entête *User-Agent* de la demande est la clé d'accès du registre de profils WURFL. Nous avons étendu ses profils avec une ontologie de description de dispositifs mobiles. En effet, les propriétés des profils WURFL font plutôt référence à des caractéristiques du navigateur Web et aux propriétés matérielles du dispositif. L'objectif de cette extension est d'augmenter la liste de caractéristiques des plates-formes en incluant des informations sur les capteurs intégrés au dispositif et la description de ces plates-formes de programmation. Nous décrivons en détail notre extension de WURFL et le processus de détection d'un dispositif mobile dans la Section 7.1.1.

Une fois le modèle de dispositif identifié, et son profil étendu obtenu, le portail d'applications met en place l'adaptation de la méta-application requise par l'utilisateur. La méta-application contient la liste des composants à déployer et ceux à activer dès le début de l'exécution de l'application. Cette liste est organisée en deux parties : la première possède les composants sans lesquels l'application ne peut être exécutée, et la seconde de la liste des composants optionnels. La méta-application contient également une liste de services et leurs alternatives d'implémentation.

Chaque composant contient la description des bibliothèques de la plate-forme mobile qu'il utilise (par exemple, bibliothèque de prise de photos), les conditions minimales d'exécution (par exemple, qualité de l'appareil photo supérieure à 2Mb), les services fournis, et les services qu'il doit éventuellement utiliser. Pour la description des services à utiliser, le concepteur peut les qualifier à l'aide des méta-informations *obligatoire* et *optionnel*.

Lors de l'installation, les informations qui font référence aux attributs statiques d'un modèle de dispositif sont évaluées. Elles permettent au gestionnaire de déploiement de choisir

les composants les mieux adaptés au contexte d'utilisation, ou d'être informé d'un échec du déploiement (par exemple, lorsqu'aucune des implémentations d'un service dont depend un composant ne peut être exécutée). Le modèle de description d'une méta-application est présenté dans la Section 7.1.2.

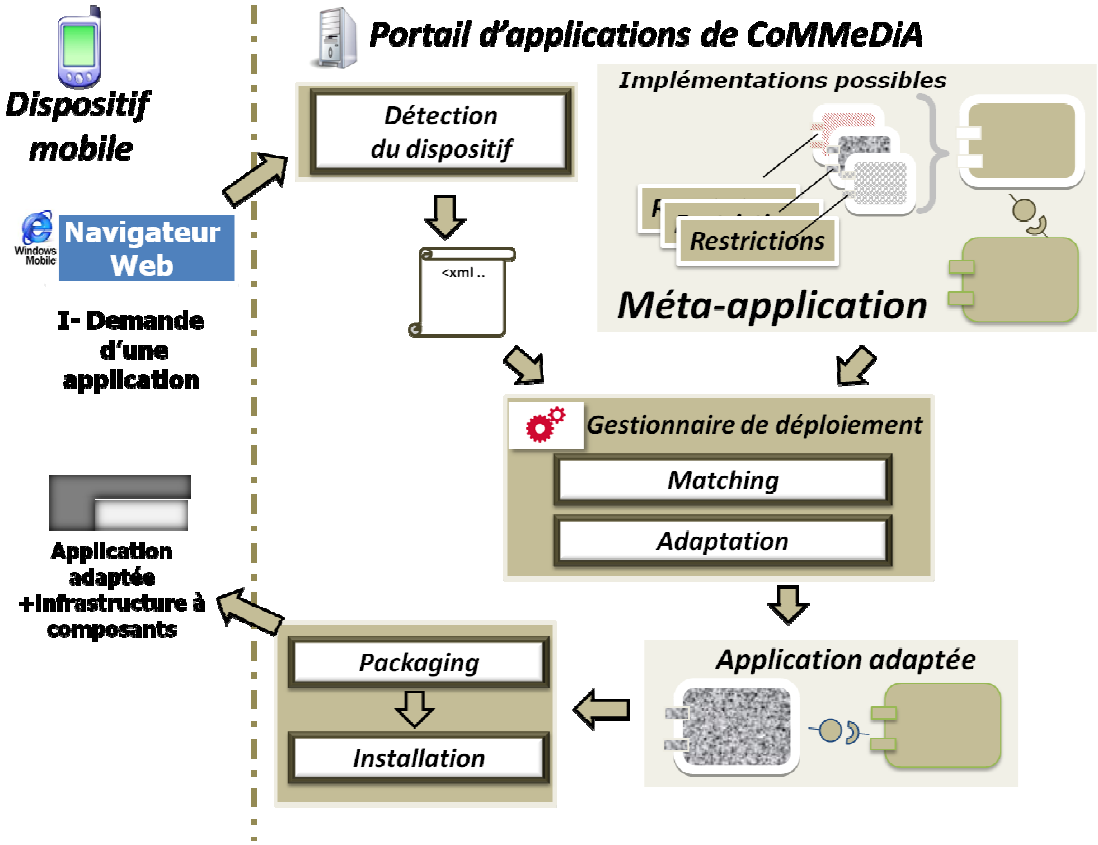


Figure 73 - Schéma du déploiement adaptatif.

En utilisant la description de la méta-application et du profil étendu du dispositif mobile, le portail d'applications de CoMMediA met en place un processus de mise en correspondance afin d'établir la version la mieux adaptée au contexte d'utilisation de l'application. La liste de composants de la méta-application est filtrée afin de ne contenir finalement que les composants susceptibles d'être activés sur le dispositif mobile. Les composants qui utilisent des bibliothèques absentes sur la plate-forme disponible sur le dispositif sont éliminés. Ensuite, les directives qui font référence aux attributs statiques (tels que « qualité de l'appareil photo supérieure à 2Mb ») sont évaluées. Enfin, un graphe de dépendances des composants est créé. Le système vérifie pour chaque composant qu'il existe au moins un autre composant pour fournir le service dont il dépend obligatoirement.

À la fin du processus, le système de déploiement vérifie si les composants qui appartiennent à la catégorie des composants obligatoires de l'application peuvent être déployés, sinon un message d'échec de déploiement est envoyé à l'utilisateur. Une fois la version adaptée déterminée, CoMMediA réalise l'assemblage du code et envoie au dispositif l'application générée. Après l'installation de l'application finale (composants de l'application + infrastructure à composants), le conteneur active les composants nécessaires pour l'exécution. Le processus de mise en correspondance est décrit dans la Section 7.1.4.

7.1.1 Description des profils de dispositifs mobiles

La découverte des caractéristiques d'un dispositif mobile est fondamentale pour l'automatisation du processus de déploiement. Plus on dispose d'informations sur le dispositif mobile, plus riche est la description de la méta-application. Dans l'état de l'art, nous avons vu deux types d'approche de description de profils : celles utilisant un registre de profils et celles basées sur un vocabulaire propre (souvent décrites par des ontologies). Les approches basées sur des registres (WURFL, Marjory, ..) offrent la découverte automatique des dispositifs mobiles, néanmoins elles possèdent un vocabulaire des propriétés des modèles limité pour l'approche d'adaptation que nous proposons. Les approches basées sur un vocabulaire propre ([Indulska *et al.*, 2003], [Wagelaar, 2005], etc.) permettent une description extensible des caractéristiques. Cependant ces approches n'offrent pas la capacité de découverte automatique de dispositifs fournie par les registres. Nous proposons donc une approche hybride qui utilise les infrastructures des registres pour découvrir le modèle du dispositif de l'utilisateur, et qui exploitent ensuite des ontologies pour enrichir la description stockée dans les registres.

Bien que, par nature, dédiée à l'adaptation d'applications mobiles Web, nous avons opté pour l'usage de WURFL en raison de l'extension possible offerte des profils de dispositifs mobiles. De plus, l'usage de l'extension conserve l'efficacité du mécanisme d'identification des modèles de dispositifs qui est réalisé en utilisant l'entête d'une demande Web.

7.1.1.1 WURFL étendu

Le mécanisme d'extension du registre (le WURLF *Patchs*) permet d'associer un fichier d'extension aux profils stockés. Nous proposons ainsi d'associer une ontologie décrivant d'autres attributs du dispositif, et vue comme une extension de son profil. Nous donnons la priorité à la description des plates-formes préinstallées et des ressources matérielles intégrées (en incluant les capteurs), car ces attributs sont très importants pour l'adaptation d'une application mobile et sensible au contexte. La Figure 74 et Figure 75 présentent les concepts de notre ontologie.

Cette ontologie est, en fait, une extension de l'ontologie *Context_Top* dont le contexte est réduit aux éléments informationnels qui identifient le futur contexte d'exécution d'une application. Ces concepts sont inspirés de l'ontologie de [Wagelaar, 2005] (Chapitre 5). Cependant, notre ontologie permet de représenter les attributs dynamiques d'un dispositif mobile et les attributs statiques du modèle du dispositif. L'objectif est de disposer d'une ontologie qui peut être exploitée à la fois par un processus d'adaptation lors de l'installation d'une application (les attributs statiques sont alors mis à profit), et par un processus d'adaptation dynamique lors de l'exécution (les attributs dynamiques sont alors observés et exploités).

Une instance du concept « MobileDevice » représente le dispositif de l'utilisateur. Ce dispositif possède une description de son modèle (« MobileDeviceModel ») et de ses ressources matérielles dont les propriétés changent au long de l'exécution d'une application (« DynamicHardwareResource »). Le concept « MobileDeviceModel » décrit les ressources matérielles et logicielles du dispositif (comme sur le profil CC/PP). Nous pouvons savoir, par exemple, si le dispositif possède un GPS, un écran tactile. Concernant les ressources logicielles, nous pouvons décrire dans le détail les plates-formes disponibles sur le dispositif mobile (version, bibliothèques, propriétés, limitations). La Figure 76 montre un exemple d'ontologie de description d'une plate-forme mobile.

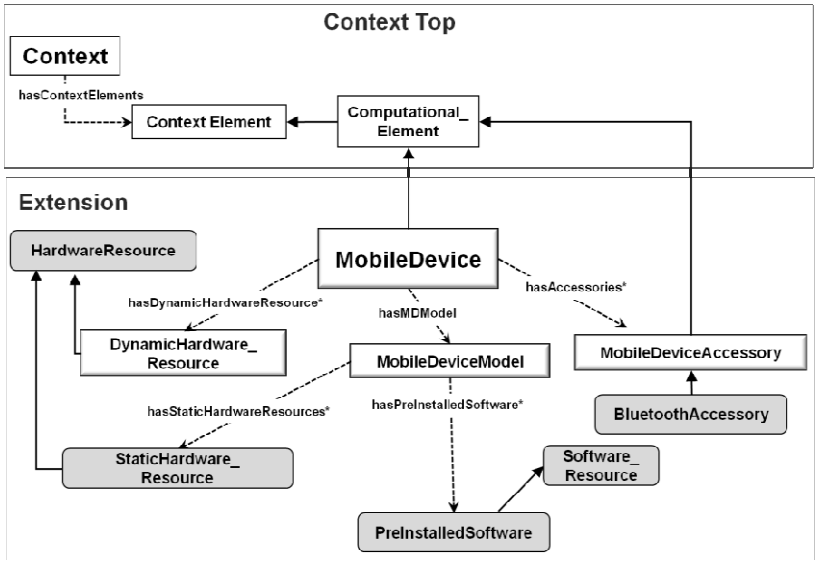


Figure 74 - Ontologie pour la description d'un dispositif mobile.

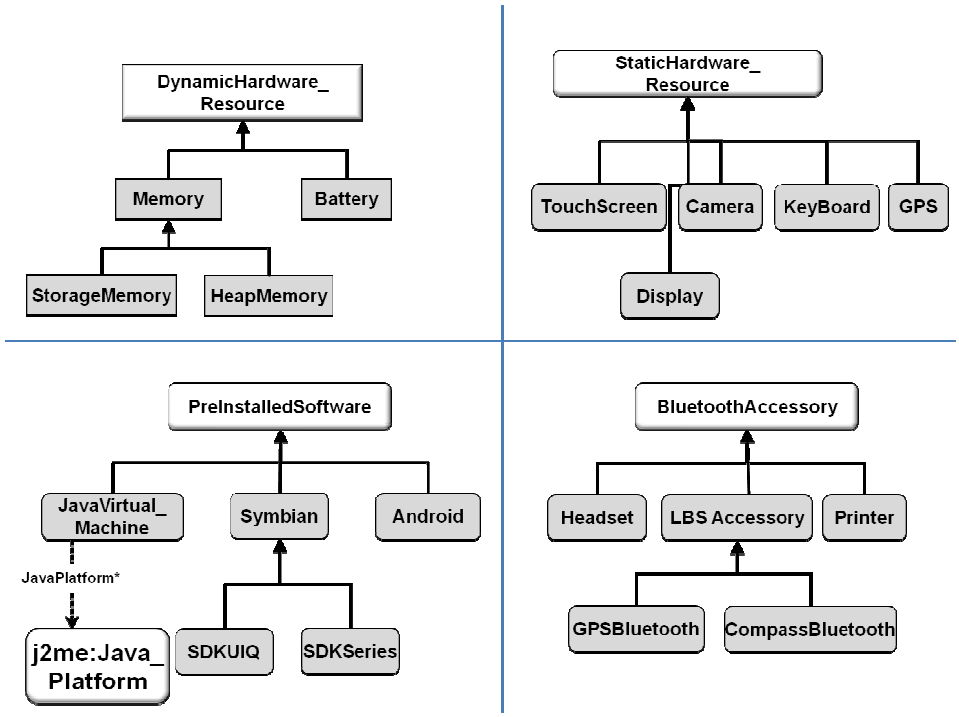


Figure 75 – D'autres concepts de l'ontologie pour la description d'un dispositif mobile.

Cette ontologie, appelé *J2MEOntology*, décrit les plates-formes Java, plus particulièrement, la plate-forme J2ME qui représente le meilleur exemple de fragmentation de bibliothèques. Nous pouvons, à l'aide de *J2MEOntology,* expliciter les bibliothèques supportées par la version de la machine virtuelle du dispositif de l'utilisateur. Par exemple, pour la plate-forme J2ME MIDP, il est possible de connaître les caractéristiques de la machine virtuelle (par exemple, J2ME MIDP 2.0), la liste des bibliothèques optionnelles supportées (telles que JSR 82 qui donne accès à l'interface Bluetooth) et leurs fonctions (par exemple, l'échange de messages par Bluetooth). L'héritage entre les versions de machine virtuelle (par exemple, MIDP 2.0 hérite de MIDP 1.0) explicite que tout dispositif possédant la version 2.0 peut exécuter également une application qui nécessite de la version 1.0 pour

fonctionner. Des ontologies similaires peuvent être construites en suivant le même schéma pour les autres plates-formes de programmation.

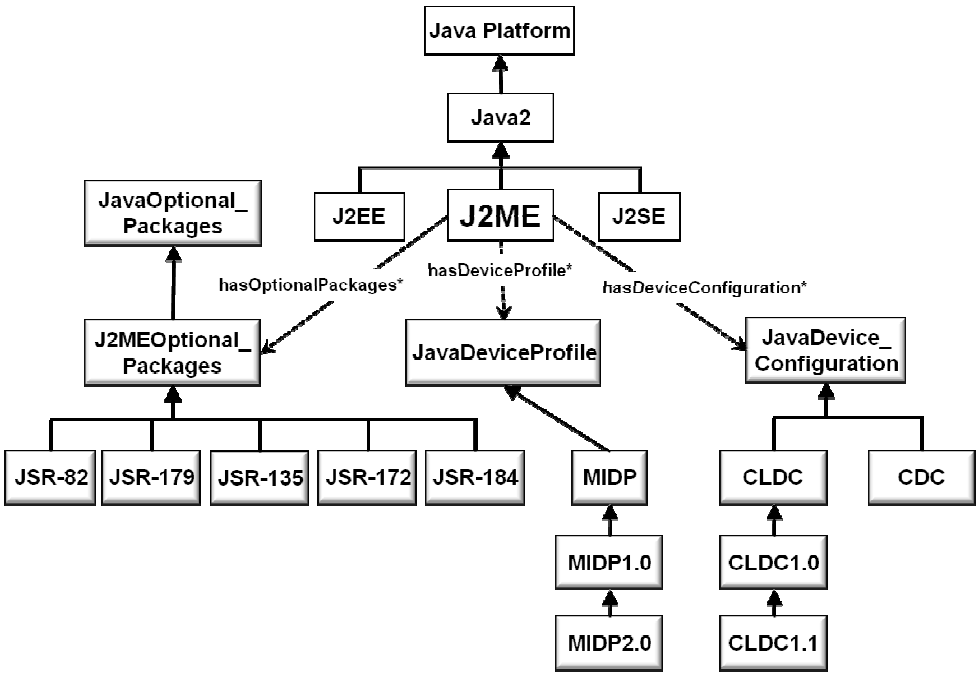


Figure 76 - Ontologie de la plate-forme mobile J2ME.

7.1.1.2 Création de profils

La création de profils WURFL est assurée par sa propre communauté de développeurs qui insèrent les nouveaux profils à partir de fichiers UAPROF. Cependant, ces profils ne contiennent pas les informations contenus dans notre extension. Afin d'éviter de décrire manuellement ces informations, nous proposons l'usage d'applications de capture de descriptions. Un développeur souhaitant créer le profil d'un nouveau dispositif, envoie une demande au portail d'applications de CoMMeDiA à partir du nouveau dispositif. Si l'extension du profil n'existe pas sur le registre, le gestionnaire de déploiement fait la lecture du profil WURFL classique afin de découvrir les noms des plates-formes de programmation disponibles. Il déploie ensuite une application, appelée « Sniffer », qui utilise uniquement les bibliothèques standard de la plate-forme de programmation (par exemple, une application J2ME MIDP1.0) pour capturer les informations de l'extension. Une fois l'application « Sniffer » exécutée, le profil du modèle du dispositif est généré et stocké par le WURFL Patch. Le Sniffer peut être étendu pour supporter de nouveaux attributs supportés par notre ontologie.

Le Tableau 13 présente des fonctions exécutées par le Sniffer J2ME et les attributs capturés. Par exemple, la fonction « System.getProperty("video.snapshot.encodings") » retourne les caractéristiques de l'appareil photo telles que la possibilité d'enregistrer de vidéos, la résolution de prise de photo en pixels, le format, …. En absence d'appareil photo intégré, la méthode retourne *null*. Les valeurs découvertes par cette fonction sont stockées en utilisant le concept Camera de notre ontologie.

Tableau 13 - Code de découverte des propriétés de la plate-forme mobile J2ME.

Propriété	Méthode
Capteur de Localisation (JSR-172)	System.getProperty (”microedition.location.version”)
Bluetooth (JSR-82)	System.getProperty(”bluetooth.api.version”)
Camera (JSR-135)	System.getProperty("video.snapshot.encodings")
SVG (JSR-266)	System.getProperty (”microedition.m2g.version”)
3D (JSR-184)	System.getProperty (”microedition.m3g.version”)
Accès aux fichiers (JSR -75)	System.getProperty (”microedition.io.file.FileConnection.version”)

7.1.1.3 Détection d’un dispositif

La détection d’un dispositif mobile est similaire à celle du registre WURFL. L’utilisation envoie la demande d’une application, et le gestionnaire de déploiement du portail utilise l’entête de la demande Web pour localiser le profil du dispositif et son extension. En cas d’absence de l’extension, le déploiement d’un Sniffer peut être mis en œuvre. La Figure 77 illustre le diagramme de séquences dans cette situation. À la fin du processus de génération du profil, le Sniffer demande une mise à jour, et le gestionnaire de déploiement répond avec l’application demandée par l’utilisateur qui a finalement pu être adaptée.

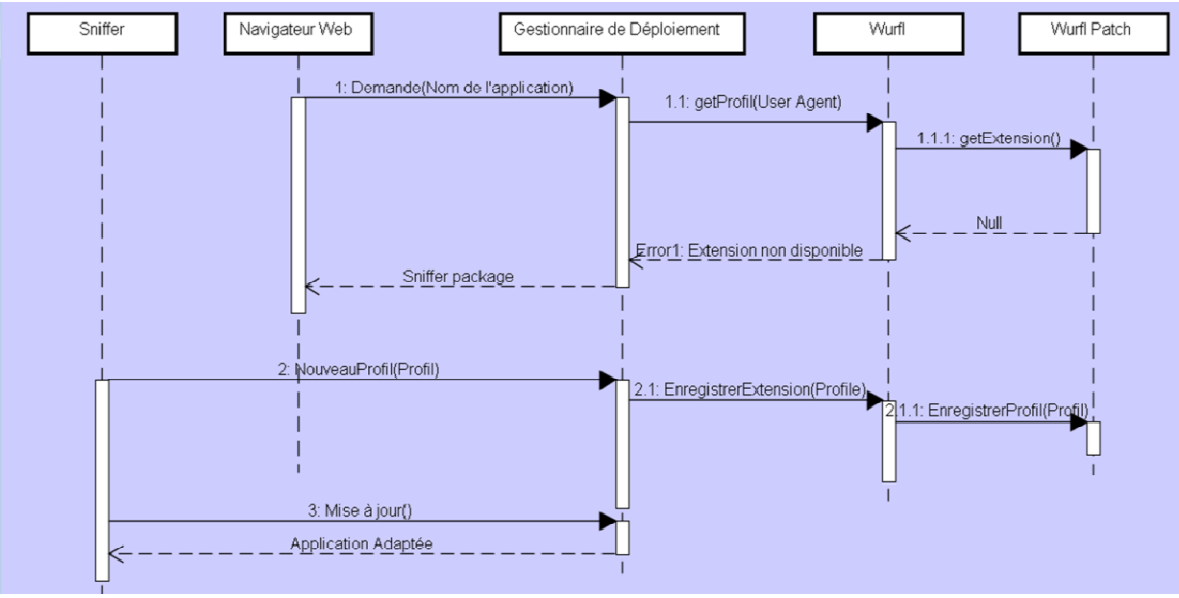


Figure 77 - Diagramme de séquences décrivant l'installation d'une application lorsque l'extension de description du dispositif n’existe pas.

7.1.2 Description d’une méta-application

Dans cette section, nous présentons un modèle de description de versions d’une application mobile et orientée composants. Ce modèle permet la description d’une méta-application qui peut être vue comme un plan de déploiement simplifié, car les composants vont être déployés sur un même emplacement (le dispositif mobile). La méta-application décrit la variabilité des fonctionnalités d’une application et de ses configurations possibles. L’application instance est un sous-ensemble de la méta-application qui changera selon les caractéristiques du dispositif de l’utilisateur et de la plate-forme de programmation.

En nous appuyant sur les travaux d’Ayed *et al.* [Ayed *et al.,* 2008] et Chefrour [Chefrour, 2005], nous identifions trois points de variabilité d’une application mobile qui sont influencés par les propriétés de son futur contexte d’exécution :

- **L'assemblage de composants**. Une méta-application doit décrire une liste idéale des composants qui doivent être déployés. Cependant, une partie de ces composants peut ne pas être installée, car ses dépendances d'exécution (une bibliothèque, la présence d'un capteur, …) ne sont pas satisfaites par le modèle du dispositif mobile de l'utilisateur.
- **Les paramètres/propriétés d'un composant**. La configuration d'un composant peut également varier selon le contexte d'exécution. Les propriétés fonctionnelles et non fonctionnelles peuvent être associées aux caractéristiques du dispositif mobile. Par exemple, un composant d'accès à l'appareil photo changera la résolution de prise de photos selon les caractéristiques de l'appareil intégré au dispositif mobile.
- **L'implémentation principale d'un service**. Un concepteur peut définir plusieurs implémentations d'un même service, et établir dans quel contexte chaque implémentation est la plus appropriée. Les dépendances de composants doivent être analysées pour éliminer les composants qui ne peuvent pas fournir le service. Ensuite, en utilisant le profil du dispositif, le gestionnaire de déploiement doit choisir parmi les implémentations restantes la mieux adaptée.

La Figure 74 présente notre modèle de description d'une méta-application.

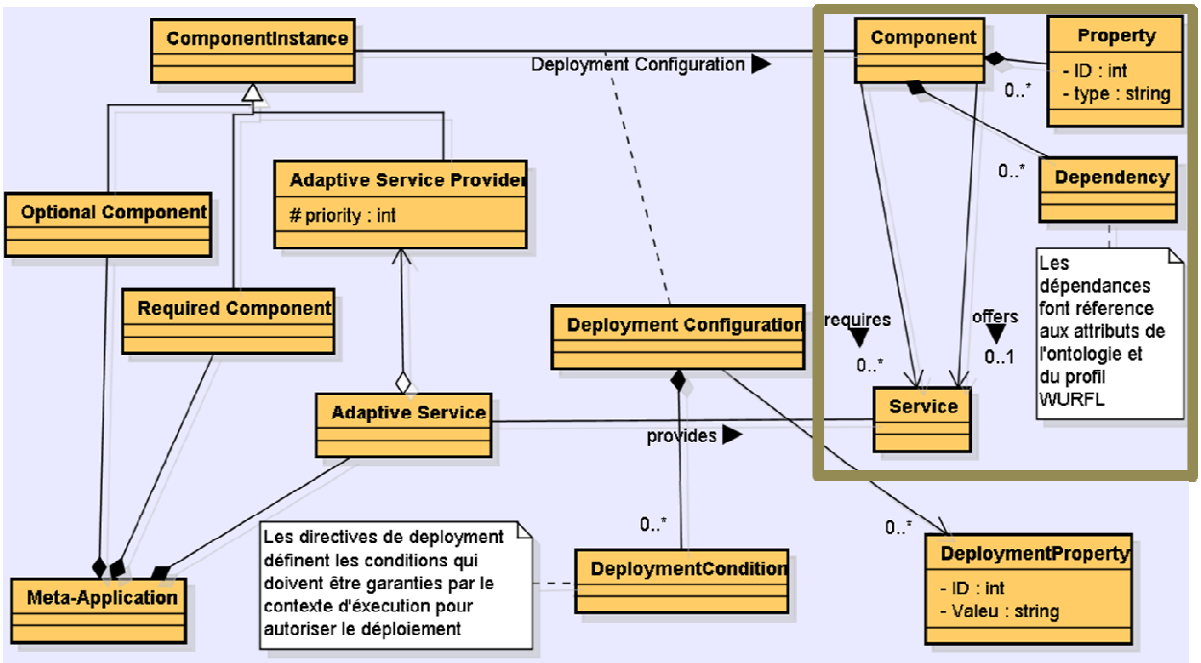


Figure 78 - Description d'une Méta-application.

Nous séparons la définition d'un composant (*Component*) de la configuration qu'un composant possède dans une application (*Component Instance*). Un composant (*Component*) offre des services (*Services*). Il peut également utiliser des services, et peut même exiger la présence d'un service pour s'exécuter correctement. La description d'un composant définit également des propriétés (*Property*) qui peuvent être attribuées par le concepteur avant son activation, et les dépendances d'exécution (*Dependency*). Ces dépendances définissent les bibliothèques exigées pour l'exécution, et les caractéristiques minimales que la plate-forme et le dispositif mobile doivent posséder pour permettre la mise en œuvre correcte du composant. Les dépendances peuvent faire référence aux attributs du profil WURFL et de son extension décrite par notre ontologie. Nous décrivons notre modèle de description d'un composant et proposons une infrastructure à composants pour les plates-formes mobiles dans la prochaine section.

Une méta-application est décrite en trois parties : les composants obligatoires (*Required Component*), les composants optionnels (*Optional Component*), et les services adaptatifs (*Adaptive Service*).

Les composants optionnels et obligatoires sont, en fait, les instances d'un composant (*Component Instance*) et possèdent une configuration de déploiement. Cette configuration est un ensemble de métadonnées que indiquent au gestionnaire de déploiement comment le composant doit être configuré et quand il est autorisé à être déployé. La configuration exprime les valeurs des propriétés (*Deployment Property*) que l'instance du composant doit affecter aux propriétés du composant (*Property*) pour cette application en particulier. Le concepteur de la méta-application peut lier les valeurs de propriétés de déploiement à des attributs du futur contexte d'exécution de l'application. Par exemple, un *composant A* qui affiche une carte possède la propriété *taille de la carte* (largeur, longueur). Un concepteur d'une méta-application établit qui la carte doit remplir l'écran du dispositif en attribuant à cette propriété la valeur « #resolution_height, #resolution_width » à l'aide de *Deployment Property*. Le gestionnaire de déploiement remplace ces paramètres par les valeurs de l'écran du profil du dispositif découvert par notre processus de détection.

La configuration de déploiement d'un composant définit également des directives de déploiement (*Deployment Conditions*). À l'instar des approches d'adaptation présentées dans l'état de l'art, les directives définissent les conditions minimales devant être satisfaites pour autoriser le déploiement d'un composant. Une directive est en fait une règle qui utilise des opérateurs tels que >,<,=, et ≠ et fait référence aux attributs du futur contexte d'exécution de l'application. Nous tenons à souligner que ces directives sont différentes des dépendances d'exécution. Les dépendances contextuelles sont des restrictions du composant qui ont été définies par son développeur et doivent être satisfaites pour n'importe quelle application (par exemple, un *composant A* dépend d'un GPS pour fonctionner). Tandis que les directives de déploiement sont des conditions établies par le concepteur de la méta-application. Elles définissent simplement les caractéristiques qui doivent être garanties par le futur contexte exécution de l'application afin d'autoriser le déploiement du composant (par exemple, le *composant A* doit être déployé uniquement si le dispositif possède plus de 2MB de mémoire vive et un écran tactile, ceci est un choix du concepteur de la méta-application, car ces restrictions n'empêchent pas le composant de fonctionner).

La description des composants optionnels et obligatoires est utilisée par le gestionnaire de déploiement pour éliminer les composants qui ne peuvent pas fonctionner et ceux qui ne doivent pas être déployés dans ce contexte d'exécution. Nous avons également introduit un autre mécanisme de variabilité dont l'objectif n'est plus d'éliminer un composant, mais plutôt de choisir le mieux adapté pour fournir un service. Le concept *AdaptiveService* décrit ce mécanisme de variabilité. Il représente un service adaptatif qui peut être vu comme un point de flexibilité de l'application. Un service adaptatif décrit les diverses alternatives d'implémentation d'un service. Ces alternatives sont des instances d'un composant et, par conséquent, possèdent des conditions de déploiement et de dépendances d'exécution. Ces alternatives sont également classées par le concepteur en utilisant un ordre de priorité. L'objectif est d'établir le composant à déployer lorsque diverses alternatives sont satisfaites par les propriétés du contexte d'exécution.

La Figure 79 présente un exemple d'une méta-application qui définit une application dont la fonction principale est la prise de photos affichant une carte avec la position du lieu de création des documents multimédias. Le composant PhotoMap est le composant principal de l'application. Il a besoin du service ShowMap pour fonctionner et peut utiliser le service fournit par RecVideo si celui-ci est présent. Ce service offre l'enregistrement de vidéos. RecVideo possède une dépendance d'exécution établie par le développeur du composant (la présence d'une bibliothèque d'exécution et enregistrement de vidéos), et une directive de

déploiement établie par le concepteur de la méta-application (plus de 10MB de mémoire de stockage). ShowMap est un service adaptatif implémenté par deux composants : SVGMap et JPEGMap. Le composant SVGMap a la priorité, et il nécessite de la présence de la bibliothèque d'affichage de fichiers SVG (*Scalable Vector Graphics*). JPEGMap, quant à lui, n'a pas de dépendances d'exécution.

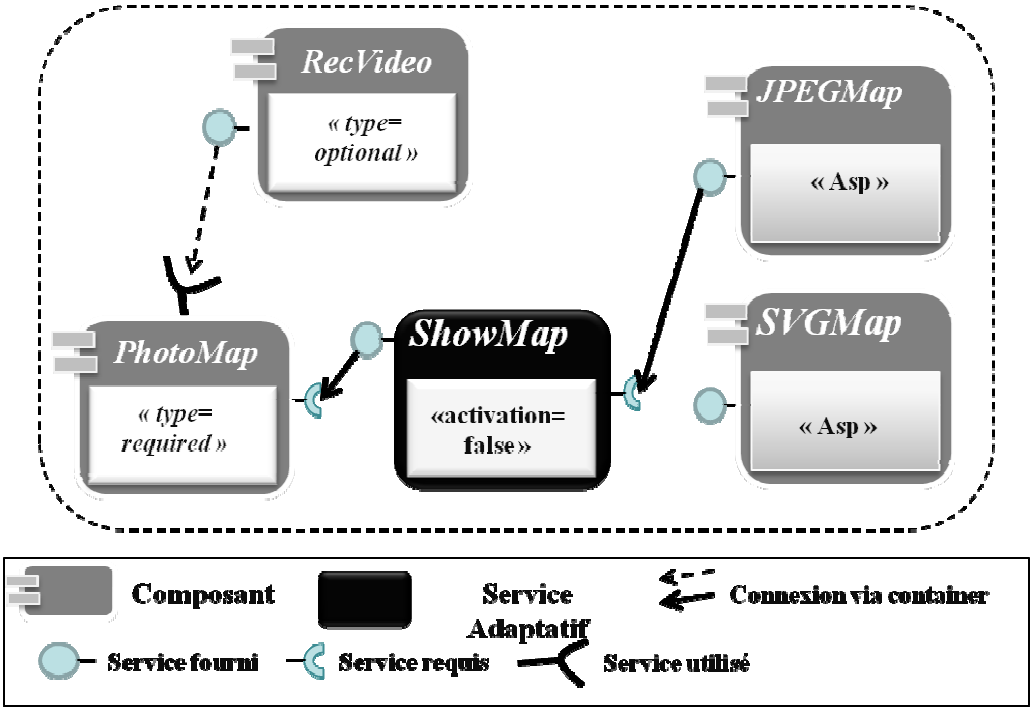


Figure 79 - Exemple visuel d'une méta-application.

Nous présentons, sur la Figure 80, la représentation en XML de cette méta-application. Le concepteur affecte les paramètres du composant PhotoMap en utilisant les propriétés de l'écran du dispositif mobile (*<parameter id="sizemap" value="#display_height, #display_width" />*) et établit également que ce composant doit être activé dès le début de l'exécution de l'application. La condition de déploiement de ce composant fait référence à notre ontologie, et indique que le composant doit être déployé si un GPS est présent sur le dispositif mobile (*<condition contextClass="MobileDeviceModel" contextpropertie="hasGPS" operator="equals" value="true" />*).

Plusieurs versions peuvent être générées à partir de cette méta-application :

- **Version contenant PhotoMap, SVGMap**. Lorsque la plate-forme du dispositif possède la bibliothèque de lecture de SVG, néanmoins, elle ne peut pas enregistrer des vidéos. Dans cette situation, SVGMap et JPEGMap peuvent être déployé, cependant uniquement SVGMap est installé en raison de sa priorité.
- **Version contenant PhotoMap, JPEGMap, et RecVideo**. Lorsque le dispositif ne possède pas la bibliothèque de lecture de SVG, néanmoins il peut enregistrer des fichiers vidéo, et il possède plus de 10MB de mémoire de stockage.
- **Version message d'échec**. Lorsque le dispositif ne dispose pas d'un GPS.

```
<?xml version="1.0" encoding="UTF-8"?>
<meta-application>
      <!–
          <compinstance>
           Component instance properties :
           - name -> it defines the name of the component package file without the extension
           - classpath-> it defines the path of the component package
           - type -> one of {required, optional, asp}
                 #mandatory : if the component can not be activated, the program can't
                 be installed.
                 #optional : even if the component can not be activated, the program can
                 be installed.
                 #asp : an Adaptive Service Provider.
           - activation -> if true the component is activated in the beginning of the
application execution
          Compinstance subelements: condition, parameter
           <condition>
           Deployment condition properties :
             -contextclass -> ontology concept or "WURFL"
             -contextproperty -> concept property or WURFL property
             -operator -> one of {more, less, equal, moreequal, lessequal]
             -value -> the value to be compared
           <parameter>
           Component Parameter (DeploymentProperty) properties :
             -id -> parameter identifier
             -value -> the value of the parameter
    -->
  <compinstance name="PhotoMap" classpath="/components/ PhotoMap /"  type="mandatory" activation="true">
    <parameter id="sizemap" value="#resolution_height, #resolution_width" />
    <condition contextClass="MobileDeviceModel" contextproperty="hasGPS" operator="equals" value="true" />
  </ compinstance >
  <compinstance name="RecVideo" classpath="/components/ ViewMap /"  type="optional" activation="false">
    <condition contextClass="WURFL" contextproperty="j2me_storage_size" operator="more" value="10240" />
  </ compinstance >
  <compinstance name="JPEGShowMap" classpath="/components/ShowMap/"  type="asp" />
  <compinstance name="SVGShowMap" classpath="/components/ShowMap/"  type="asp" />

    <!–adaptive service properties :
       <adaptiveservice>
      - type -> one of {required, optional}
                 #mandatory : if any component can not provide the service, then the program
can't be installed.
                 #optional : even if any component can not provide the service, the program
can be installed.

     - install -> one of {all, onlyone}
                 #all : all components should be installed if the mobile device can
                 activate them.
                 #onlyone : only the high priority component should be installed

           Warning: The order of components determines
           the priority.
    -name -> the publication name of the provided service
    -interface -> the service interface
    -activation -> if true the component is activated in the beginning of the application
execution
     <asp>
     -name -> the name of the Adaptive Service Provider –>
  <adaptiveservice name="showmap" type="mandatory" install="onlyone" activation="false"
interface="org.steamer.maps.ShowMap">
        <asp name="SVGView" >
        <asp name="JPEGView" >
  </ adaptiveservice >
</meta-application>
```

Figure 80 – Représentation en XML de méta-application.

7.1.3 L'infrastructure à composants de CoMMeDiA

Nos processus d'adaptation et d'acquisition de contexte reposent sur la description des applications mobiles sous forme de composants interopérables. Dans cette section, nous présentons notre modèle de composants.

7.1.3.1 Avant-propos

Plusieurs infrastructures à composants existent pour les applications d'entreprises telles que CORBA, EJB (Enterprise JavaBeans) et StarCCM[103]. Dans l'état de l'art, nous avons vu que ces infrastructures à composants ne sont pas complètement appropriées aux environnements d'exécution limités des dispositifs mobiles. Une telle infrastructure nécessite de grandes quantités de ressources (mémoire, batterie, communication, processeur) qui ne sont pas toujours disponibles sur les appareils mobiles d'aujourd'hui.

Nous souhaitons une infrastructure à composants légère qui fournisse au moins un annuaire de services et un modèle strict de connexion par le biais d'interfaces (qui, pour nous, représentent les services). Des services non-fonctionnels (tels que le contrôle de transaction, sécurité, persistance) ne sont pas nécessaires à ce stade pour l'approche que nous défendons, car la plate-forme doit servir simplement de base pour l'exécution de notre approche d'adaptation.

Nous avons donc choisi de proposer notre propre infrastructure à composants. L'essentiel de notre plate-forme à composants est inspiré de la plate-forme OSGi et elle est également conçue pour un usage sur des plates-formes de programmation Java. Dans l'état de l'art, nous avons vu qu'une mise en œuvre de la spécification OSGi R3 de 80KB, appelé Concierge, fonctionne sur des dispositifs mobiles qui supportent la machine virtuelle Java CDC (par exemple, certains *smartphones* possédant les systèmes d'exploitation *Android* et *Wince*). Cependant, l'objectif de CoMMeDiA est d'assurer la portabilité de nos applications pour le plus grand nombre possible d'appareils mobiles. Or, la plupart des plates-formes Java actuellement disponibles sur les dispositifs mobiles, telles que J2ME MIDP et DOJA, n'offrent pas les mécanismes d'introspection nécessaires pour mettre pleinement en œuvre les spécifications OSGi.

Nous avons ainsi créé un modèle de composants plus simple qui peut être implémenté sur ces plates-formes et également sur Android et WinCe. Si la mise en œuvre sur J2ME/MIDP et DOJA a le bénéfice de la portabilité, l'architecture perd, en revanche, en fonctionnalités. En fait, le téléchargement et l'installation de composants lors de l'exécution d'une application sans que celle-ci soit interrompue ne sont pas possibles sur ces plates-formes. Le chargement dynamique d'une classe est possible uniquement si la classe est déjà présente sur le paquet d'installation de l'application, car le téléchargement d'un composant n'est pas autorisé par la machine virtuelle. Seul le téléchargement complet d'une application ou d'une mise à jour est autorisé. Malgré ces restrictions, l'adaptation lors de l'installation et la reconfiguration dynamiques des composants préalablement installés sont supportés par notre architecture.

Nous décrivons à présent notre infrastructure à composants qui comporte trois parties : un modèle de composant, un conteneur d'exécution et un modèle de connexion entre les composants.

7.1.3.2 Modèle de composant Ilet

Dans notre infrastructure, un composant, appelé *Ilet*, est une unité de déploiement qui peut être branchée sur notre conteneur. Une *Ilet*, tout comme un *bundle* OSGi, ne peut pas

[103] http://starccm.sourceforge.net/

invoquer des méthodes d'autres *Ilets* directement. La communication entre les *Ilets* est possible par le biais de services qui sont publiés dans l'annuaire de services du conteneur et sont accessibles uniquement en l'utilisant.

Un service sur notre plate-forme correspond à une *Interface Java* qui est partagée par le fournisseur et le client du service, qui sont eux-mêmes, des *Ilets* enregistrées sur la plate-forme. Une *Ilet* peut donc :

i) exécuter du code Java MIDP, Android ou Doja sur le dispositif mobile (même avec des interfaces visuelles) ;

ii) publier des services ;

iii) rechercher et accéder à des services disponibles sur l'annuaire du conteneur qui sont fournis par d'autres *Ilets* ou par le conteneur lui-même.

Le cycle de vie de l'*Ilet* est contrôlé par le conteneur et possède cinq états possibles : installé, résolu, non résolu, actif, et pause. Ces états intègrent certaines particularités des plates-formes mobiles que nous adressons. La Figure 81 montre le diagramme des états du cycle de vie et les commandes exécutées par le conteneur qui modifient l'état du composant.

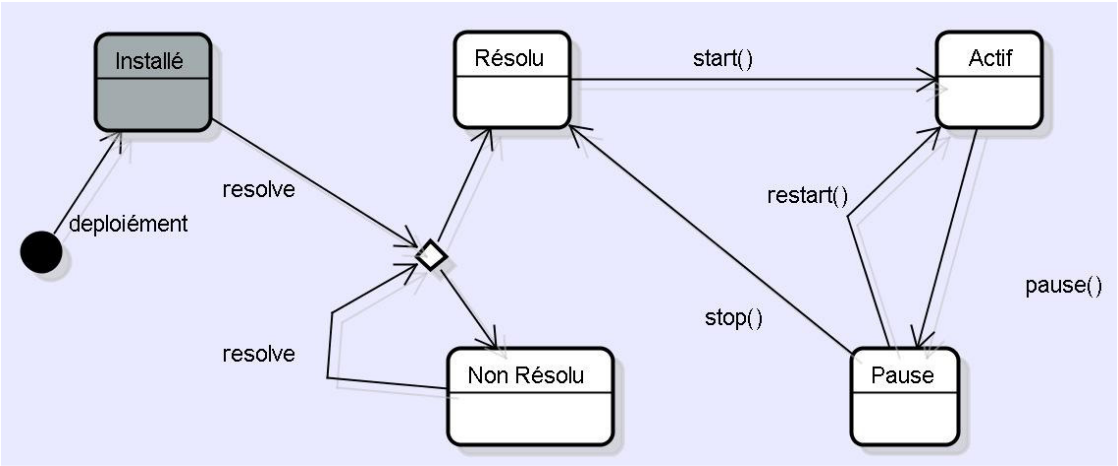


Figure 81 - Cycle de vie des composants *Ilets*.

Tout d'abord, les composants doivent être inscrits (installés) sur la plate-forme. Ensuite le conteneur vérifie les dépendances de l'Ilet. Le *Manifeste* de l'Ilet décrit les services qui sont indispensables pour exécuter l'Ilet et les dépendances d'exécution. Si ces restrictions sont résolues, ou si ce service est déjà disponible sur la plate-forme, l'Ilet passe à l'état de résolu et peut être instancié par la plate-forme. Normalement, ce processus de vérification de restrictions a été réalisé par le gestionnaire de déploiement. Cependant ,nous souhaitons établir une certaine indépendance de l'infrastructure à composants dans notre approche de déploiement. Ainsi, ce processus est également réalisé par le conteneur.

Si un paquet n'existe pas ou s'il existe un service dont il a besoin pour l'utilisation et qui n'a pas encore été publié, le composant passe dans l'état non résolu. Cela évite que l'*Ilet* occupe inutilement la mémoire vive. Le conteneur vérifie, à chaque nouvelle installation d'un *Ilet*, si les services qu'elle fournit ne permettent pas de résoudre les dépendances des autres *Ilets*.

Lorsque la commande de démarrage est exécutée, le conteneur crée une instance de la classe de l'Ilet qui implémente l'interface *IletActivator* (Figure 82). Cette classe doit contenir quatre méthodes : *start*(), *stop*(), *pause*() et *restart*(). Après la création de l'instance de l'activateur, le conteneur appelle la méthode *start*() de classe. Les méthodes *pause*() et *restart*() seront appelées lorsque l'application est placée en mode de pause par le système d'exploitation du dispositif mobile (opération commune sur les dispositifs mono-tâches, ainsi que sur les plates-formes J2ME MIDP et DOJA, par exemple lors d'un appel téléphonique). Le conteneur retire l'Ilet de la mémoire lorsque la méthode *stop*() du composant est invoquée.

Nous n'avons pas ajouté d'état de désinstallation, car l'exclusion d'un code déployé sur un dispositif mobile n'est pas disponible sur les plates-formes que nous adressons. Les opérations de mise à jour d'un composant sont également absentes en raison de ces mêmes limitations.

La communication entre le composant et le conteneur est assuré par la classe *ILetExecEnvironment* qui donne accès, par exemple, au registre de services et à l'affichage sur l'écran du dispositif.

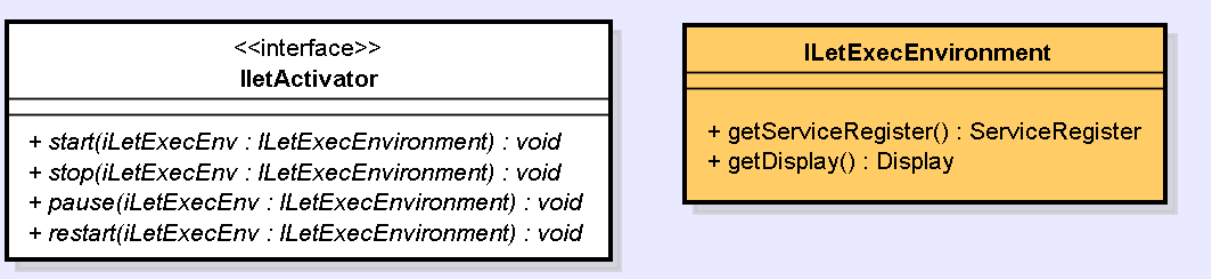


Figure 82 - Les classes permettant la communication entre le conteneur et le code d'un composant.

Les *Ilets* sont distribuées dans des paquets JAR et elles doivent contenir au moins un fichier *manifeste* et une classe qui implémente l'interface *IletActivator* (Figure 82). Le manifeste décrit les dépendances de l'*Ilet*, les services accessibles et/ou fournis, et le nom de la classe exécutable de l'*Ilet*. La Figure 83 présente le manifeste du composant SVGMap. Ce composant dépend de la présence de la bibliothèque SVG (« j2me_svgt ») pour fonctionner et offre le service ShowMap.

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
  <dependends contextClass="WURFL" contextproperty="j2me_svgt" operator="equal" value="true" />
  <dependends contextClass="WURFL" contextproperty=" j2me_midp_2_0" operator="equal" value="true" />
  <dependends contextClass="WURFL" contextproperty=" resolution_width" operator="more" value="240" />
  <dependands contextClass="WURFL" contextproperty=" resolution_height" operator="more" value="320" />
  <service name="ShowMap" classpath="org.steamer.maps.ShowMap" type="offers"/>
  <iletactivator classpath="org.steamer.maps.SVGMap.SVGMapViewIletActivator"/>
</manifest>
```

Figure 83 – Manifeste du composant SVGMap.

7.1.3.3 Modèle de connexion des Ilet

Les Ilets communiquent dans la plate-forme par le biais de services. Un service est un ensemble de méthodes décrites par une interface Java partagée par le fournisseur et le client du service. Pour fournir un service, le développeur doit créer deux classes. La première est l'interface du service qui doit être rendue publique et utilisée par le client lors du développement de son code. La deuxième classe doit implémenter l'interface du service et réaliser effectivement le comportement fonctionnel proposé par le service. La Figure 84 illustre un exemple d'usage d'un service partagé par deux Ilets : le service ShowMap (en réalité, « org.steamer.maps.ShowMap »). Le fournisseur du service ShowMap (le composant JPEGMap) publie ce service dans l'annuaire du conteneur (*ServiceRegister*) dont l'accès est obtenu par les méthodes de la classe *ILetExecEnvironment*. Le fournisseur enregistre en réalité une instance d'une classe qui implémente le service (JPEGMap). L'Ilet client acquiert le service en cherchant l'annuaire à partir du nom de publication (« ShowMap »). L'annuaire retourne une liste de composants qui implémentent ce service et leur priorité est utilisée pour les classer. L'annuaire du service offre également une méthode de découverte basée sur le nom de classe du service (par exemple, « org.steamer.maps.ShowMap »).

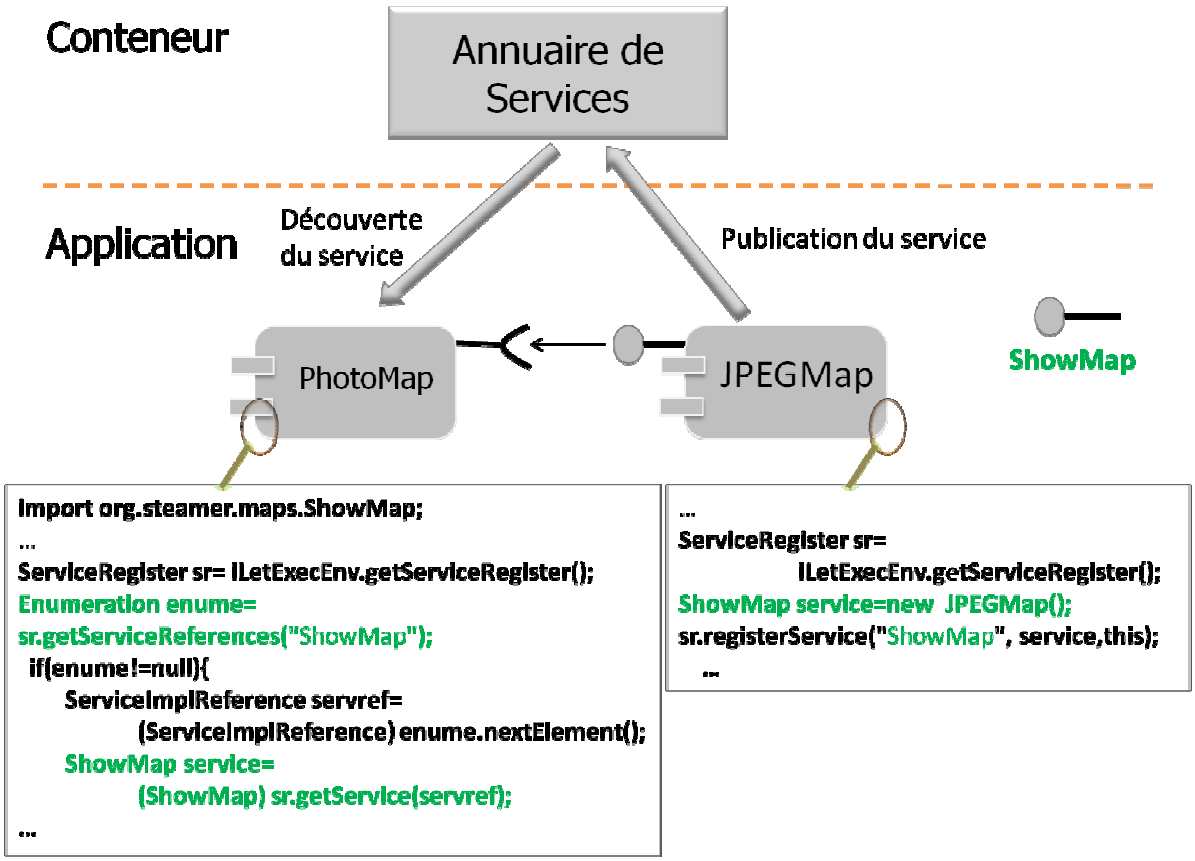


Figure 84 - Exemple de connexion entre les services. Le service ShowMap fournit par Ilet JPEGMap est utilisé par l'Ilet PhotoMap.

Une fois que le client a obtenue un « pointeur » vers l'implémentation du service, il peut appeler les méthodes spécifiées par l'interface. Le conteneur n'intervient plus dans cette communication entre les composants. Ce modèle de connexion est un **modèle synchrone** d'appel de méthodes. Cependant, un développeur peut utiliser le patron de conception observateur[104] ou le modèle *white-board*[105] afin d'assurer un **modèle asynchrone** de communication entre les composants.

Lorsque la recherche d'un service échue et ne retourne aucune implémentation, le composant client peut s'enregistrer sur le conteneur afin d'être notifié lorsqu'un service implémentant l'interface désiré est activé. Il peut également chercher dans l'annuaire des composants inactifs qui offrent cette interface et les activer.

7.1.3.4 Le conteneur et le déploiement d'une application

Le conteneur contrôle le cycle de vie des *Ilets* et possède l'annuaire de services publiés par les composants. Le conteneur fournit également des services de base pour la communication, l'accès au système de fichiers et à l'affichage. Les *Ilets* peuvent utiliser le conteneur pour déclencher ou arrêter les autres *Ilets,* et également pour enregistrer, rechercher ou instancier des services. La classe *ILetExecEnvironment*, passée en paramètre à *IletActivator*, donne accès aux fonctions du conteneur.

Le conteneur contient trois registres pour enregistrer les différentes *Ilets*. Le premier enregistre toutes les *Ilets* installées et les services qui peuvent être activés, le deuxième enregistre toutes les *Ilets*

[104] http://www.research.ibm.com/designpatterns/example.htm

[105] http://www.osgi.org/wiki/uploads/Links/whiteboard.pdf

activées, et le dernier enregistre les *Ilets* qui désirent être à l'écoute d'activation des services. Ces registres sont accessibles à tout moment durant l'exécution du composant.

Le conteneur est également responsable de la lecture du plan d'installation de l'application mobile. Le plan d'installation définit les *Ilets* de l'application et leur ordre d'activation. Ce document, joint à l'application, est lu par le conteneur et enregistre chaque service et chaque Ilet dans le registre des composants installés. Toujours en fonction du plan d'installation, certains *Ilets* vont être activés à l'initialisation de l'application. À ce moment, certains parmi ces *Ilets* vont également publier des services. L'affichage est affecté à un composant qui affiche l'écran initial de l'application.

7.1.4 Génération d'une version d'application

La fonction principale du gestionnaire de déploiement de notre architecture est de transformer la méta-application en une version adaptée de l'application. La description de la méta-application est utilisée pour la construction d'un graphe de dépendances de composants (en anglais, *Component Dependences Graph-CDG*) [Li *et al.*, 2005]. Le CDG guide le processus de génération, principalement, lors de l'élimination des composants. Dans notre approche, le CDG est un graphe orienté dont les nœuds sont les composants (*ComponentInstance*) et les services adaptatifs (*AdaptativeService*). Les arcs représentent deux types de connexion. Le premier est une dépendance indiquant que le nœud d'origine est un composant et qu'il nécessite obligatoirement le service fourni par le nœud final de l'arc. Le deuxième type d'arc indique que le nœud d'origine est un service adaptif, et que le nœud final est une de ses alternatives. La Figure 85 illustre le CGD généré pour l'exemple de la Figure 79. Bien que le composant *RecVideo* fournisse un service à PhotoMap, il n'existe pas d'arc entre ces nœuds, car ce service est optionnel pour PhotoMap. La Figure 85 présente également la matrice CGD_{nxn}, où *n* est le nombre de nœuds du graphe. CGD_{nxn} est la représentation matricielle du CGD. Le tableau LC est manipulé lors du processus de génération pour décider des composants à éliminer. Au départ du processus, tous les composants possèdent la valeur 1 dans ce tableau, et la valeur attribuée à un service adaptatif est égale au nombre d'alternatives (*Adaptative Service Provider*) que ces services possèdent. La Figure 85 présente le LC au début du processus.

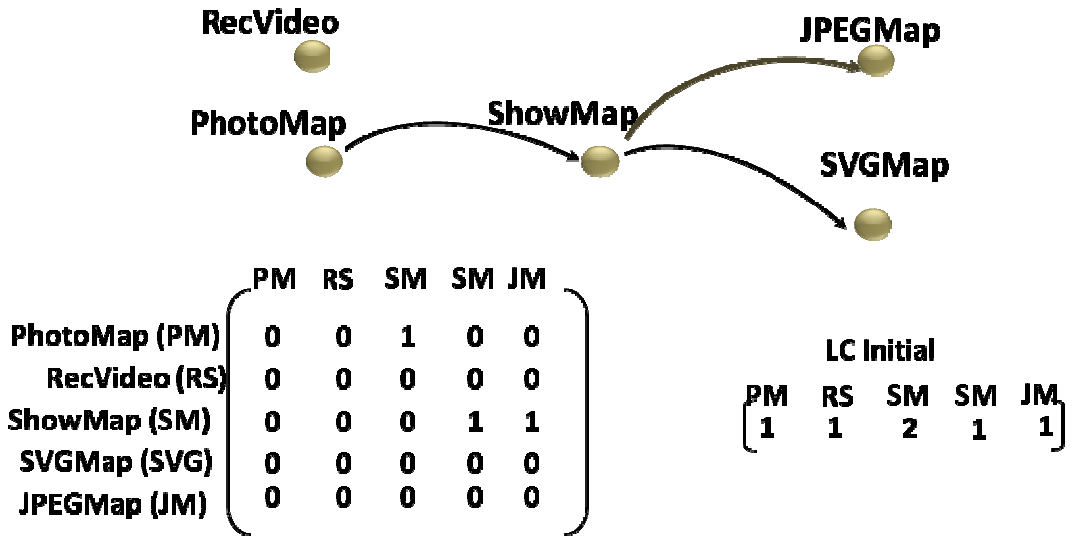


Figure 85 - CDG généré pour l'application de l'exemple de la Figure 79.

La Figure 86 présente le pseudo-code de notre processus de génération. D'abord, le gestionnaire exploite la représentation du profil du dispositif pour exécuter ce processus. En

fait, l'ontologie et le profil WURFL sont transformés en un arbre de propriétés dont les nœuds sont les « contextclass » et « contextproperties » référencés par le concepteur dans la méta-application. L'ontologie inclut également des individus dérivés à partir d'un processus d'inférence qui évalue l'héritage et d'autres constructions OWL (équivalence, etc.). Par exemple, si la machine virtuelle du dispositif est J2ME MIDP 2, elle est classée aussi comme J2ME MIDP 1, car toute application MIDP 1.0 peut s'exécuter sur une machine virtuelle MIDP 2 (ontologie Figure 76). D'autres règles d'inférences peuvent être introduites à cet instant afin de dériver davantage d'informations.

```
Algorithme : Génération d'une application
Donnée : Méta-application – MA, arbre de propriétés
Résultat : tableau LC ou null en cas d'échec

Début

CDG := construire CDG (MA) //CDG à partir de la méta-application
LC := construire LC(MA)//LC contient 1 pour un composant et n pour les services adaptatifs
Pour i Allant de 1 à Taille(LC)
  Si (Valider Dépendances d'exécution (LC[i]) = faux) ou (Valider Conditions de
Déploiement (LC[i]) = faux) Alors
        Éliminer Composant(LC,i)
  Fin Si
Fin Pour

Fin
//_______________
Procédure Éliminer Composant(LC,i,CGD)
  Si LC[i]>0 Alors //Éviter les cycles
    Si LC[i] est décrit comme obligatoire dans la méta-application Alors
      échec installation
    Sinon
      LC[i]:=LC[i]-1
      Si LC[i]=0 Alors
         Pour j Allant de 1 à Taille(LC)
                Si CGD[j,i]=1 Alors
                  Éliminer Composant(LC,j,CGD)
               Fin si
         Fin Pour
      Fin Si
    Fin si
  Fin si
Fin Procédure
```

Figure 86 - Pseudo-code du processus de génération d'une application.

Après la génération de l'arbre de propriétés, le gestionnaire de déploiement crée le CGD et le tableau LC en utilisant la représentation XML de la méta-application. Ensuite, les dépendances d'exécution et les conditions de déploiement sont évaluées. Une comparaison en utilisant les valeurs de l'arbre de propriétés est exploitée pour valider ces directives. En cas d'échéance d'une de ces validations, le gestionnaire démarre un processus d'élimination du composant, si ce composant est obligatoire, un échec de génération se produit. Sinon, la méthode va éliminer les composants qui dépendent de ce composant éliminé. Les services

adaptatifs sont éliminés uniquement lorsque toutes les alternatives ont été éliminées. À la fin du processus, le tableau LC décrit les composants à installer. Les composants possèdent la valeur zéro ne seront par dans le paquet final. Les services adaptatifs sont remplacés par les alternatives qui possèdent une valeur 1 dans le tableau LC. En cas de multiples options et de la présence de la imposée d'installation d'une seule alternative, la priorité établie par le concepteur est exploitée pour choisir parmi les alternatives, celle à installée.

Les paquets des composants sont ainsi regroupés en un unique Jar qui contient également le plan d'installation décrivant les composants et les services de l'application. Afin d'éviter de générer à nouveau un plan d'installation pour un même modèle de dispositif, un cache d'applications générées peut être mis en place pour alléger ce processus de déploiement adaptatif. Cependant, à chaque mise à jour d'un composant de la méta-application, le processus de génération doit être recalculé.

Notre approche de déploiement adaptatif repose essentiellement sur les propriétés statiques des dispositifs mobiles. Toutefois, d'autres informations susceptibles d'être découvertes avant l'installation de l'application peuvent également être introduites dans la méta-application afin de modifier les versions générées. Par exemple, des informations concernant l'utilisateur (âge, préférences, langage d'utilisation, …). Le concepteur doit décrire ces propriétés en utilisant l'ontologie Context_Top et assurer sa découverte avant la mise en œuvre du processus de mise en correspondance.

7.2 DevAC - Device Aware Context toolkit

DevAC (*Device-Aware Context Toolkit*) constitue notre proposition d'intergiciel d'acquisition de contexte. DevAC donne accès aux capteurs matériels d'un dispositif mobile (GPS, interface Bluetooth, etc.), aux capteurs logiciels qui surveillent l'état du dispositif (mémoire libre, niveau de batterie, etc.), et effectue la fusion de ces informations avec des données provenant d'autres sources (telles que des serveurs externes, les services de *ContextAnnotator*).

DevAC possède trois caractéristiques majeures :

I. **Installation adaptative des services d'acquisition de contexte selon les caractéristiques du dispositif mobile cible du déploiement**. DevAC est décrit à l'aide de notre modèle de définition de méta-applications, et repose sur un assemblage d'Ilets. Lors du déploiement d'une application qui utilise DevAC, une version adaptée de l'intergiciel est générée en respectant les limitations matérielles et logicielles du dispositif mobile cible du déploiement. Nous souhaitons augmenter, en conséquence, la portabilité de l'application, en évitant des problèmes de compatibilité entre DevAC et le dispositif mobile de l'utilisateur.

II. **Installation adaptative des services d'acquisition de contexte en fonction des informations contextuelles requises par les services de l'application**. DevAC a été conçu pour éviter l'approche du *tout-ou-rien* adoptée par les architectures d'acquisition de contexte. Le concepteur d'une méta-application peut spécifier les composants de l'application qui utilisent les services de DevAC, et expliciter les informations contextuelles qu'il souhaite découvrir. Le gestionnaire de déploiement déploie exclusivement les composants de DevAC qui fournissent ces informations. Nous souhaitons éviter de déployer et d'activer des composants d'acquisition de contexte qui ne seront jamais utilisés par l'application.

III. **Reconfiguration dynamique des services d'acquisition de contexte en fonction des services actifs de l'application et du fonctionnement des capteurs.** Nous avons conçu DevAC conformément à notre définition de contexte, en supposant que la zone d'intérêt d'un système change selon ses besoins. DevAC possède ainsi des mécanismes d'activation et de désactivation de modules de capture et d'agrégation d'informations contextuelles. L'activation et désactivation des services de DevAC sont réalisées automatiquement, en fonction des services actifs de l'application qui ont besoin d'informations contextuelles. L'objectif est d'acquérir uniquement les informations contextuelles utiles aux services de l'application à ce moment précis.

L'organisation structurale de cet intergiciel d'acquisition de contexte est inspirée du canevas d'adaptation proposé par [Henricksen *et al.*, 2004] (Chapitre 5), qui sépare le code métier de l'application, des méthodes d'acquisition de contexte et d'adaptation. Nous avons utilisé une approche similaire en utilisant notre infrastructure à composants pour coder les diverses couches fonctionnelles de l'intergiciel. De nouveaux mécanismes de découverte de service et de médiation ont été introduits dans l'infrastructure à composants afin de fournir les méthodes nécessaires au support des nouvelles caractéristiques de DevAC. Nous décrivons à présent les composants de DevAC et ces nouveaux mécanismes.

7.2.1 Les services d'acquisition de contexte

La Figure 87 illustre les composants de DevAC et certaines relations de dépendance entre ces composants.

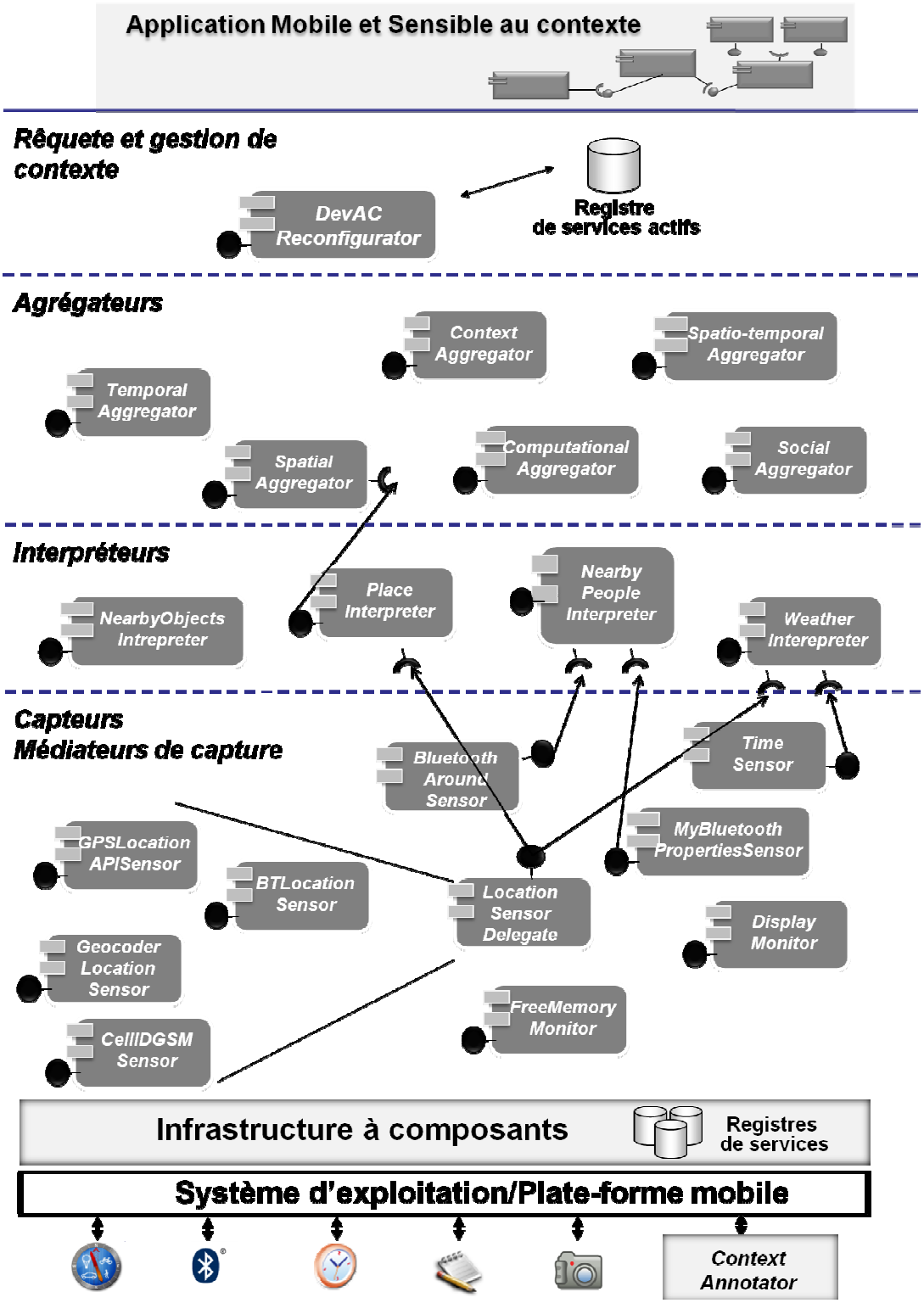


Figure 87 - La plate-forme d'acquisition de contexte DevAC.

Nous découpons DevAC en plusieurs couches : les capteurs physiques et logiciels, l'interprétation des données captées, l'agrégation des données de plusieurs capteurs ou interprétateurs, l'acquisition de contexte synchrone et asynchrone des agrégateurs, la représentation du contexte, et l'interface de requête. Chaque couche est composée d'une famille de composants implémentés en utilisant notre infrastructure à composants.

Nous identifions six types de composants :

I - Sensor

Chaque composant de cette catégorie donne accès à un capteur du dispositif. Nous avons défini un modèle général de capteur illustré dans la Figure 88. Un composant du type *Sensor* doit offrir deux services : configuration du capteur (*SensorConfigurationService)* et requête de données observées (*SensorService*). À l'intérieur du composant, une classe de configuration implémente le premier service et offre des méthodes d'activation et de désactivation du capteur (*SensorConfigurator*). Cette classe permet également de surveiller l'état du capteur (activé, déconnecté, erreur…). Une classe de capture (*Sensor*) effectue l'accès au capteur physique et publie le service qui fournit les données capturées (l'interface de requête *SensorInterface*). La classe *SensorIletActivator* est le point de communication entre le composant et le container.

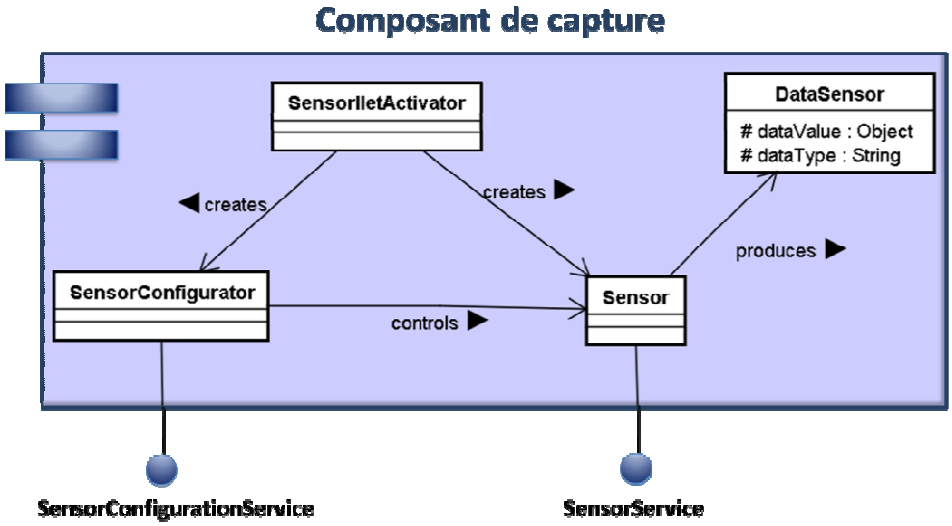


Figure 88 - Organisation interne d'un composant de capture de contexte (composant du type *Sensor*).

Divers composants de capture ont été créés suivant ce schéma pour capturer des informations nécessaires à l'exécution des services de la plate-forme ContextAnnotator (par exemple, la capture de la localisation et de l'identificateur Bluetooth du dispositif de l'utilisateur). Un *framework* a été créé pour faciliter ce processus de développement (Figure 89). Un capteur de ce *framework* peut fonctionner à la fois en mode synchrone ou asynchrone. Un client d'un capteur synchrone démarre l'acquisition de données en utilisant *SensorService*. Par exemple, un client du capteur *MyBluetoothPropertiesSensor* peut accéder à des propriétés du matériel Bluetooth intégré (adresse physique, nom de publication, etc.). La capture de ces informations est activée uniquement après une requête. D'autre part, en mode asynchrone, le client du capteur doit s'enregistrer, en utilisant *SensorService*, pour écouter les événements d'observation. Le client du composant de capture établit une condition de notification (une valeur ou un intervalle de valeurs) qui doit être valide pour déclencher la notification. À chaque nouvelle observation de ce capteur, les conditions de notification sont évaluées. Si la condition est satisfaite, le capteur notifie son client. Celui-ci utilisera l'interface de requête du capteur afin d'accéder aux nouvelles valeurs observées. La Figure 89 illustre un exemple de capteur asynchrone, le capteur *BluetoothAroundSensor* qui fournit les adresses physiques des dispositifs Bluetooth détectés à proximité de l'utilisateur.

Chaque composant de capture offre divers formats d'exportation des valeurs contextuelles observées, tels que des objets JavaBeans (*DataSensor*), du XML, et des instances OWL de l'ontologie *ContextTop*.

Nous tenons à souligner que les composants de capture sont les composants de DevAC qui possèdent le plus de dépendances d'exécution liées aux caractéristiques matérielles des dispositifs mobiles et à la présence des bibliothèques de la plate-forme Java. Lors de l'exécution du processus de déploiement, la non-satisfaction de ces dépendances élimine de fait ces composants de capture de la version à déployer de l'application.

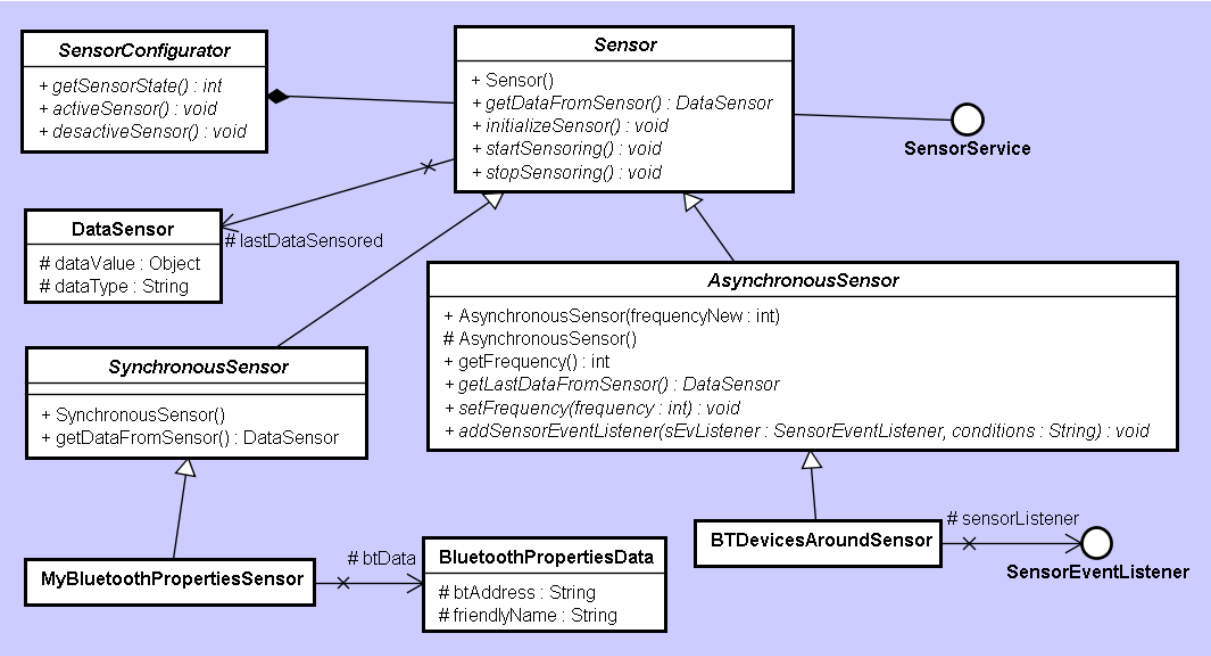


Figure 89- Framework de composants de capture.

II- Monitor

Ce type de composant suit un modèle similaire à celui des composants *Sensor*. Les composants de cette famille sont des capteurs logiciels qui fournissent des informations sur les propriétés dynamiques du dispositif, par exemple sur la mémoire vive, la position sur l'écran, le niveau de batterie. Ils peuvent également fonctionner en mode synchrone et en mode asynchrone.

III- Interpreter

Ces composants sont chargés d'augmenter le niveau sémantique d'une information contextuelle. Ils accèdent à des services externes (tels que les services de la plate-forme *ContextAnnotator*) pour inférer de nouvelles informations. Les interpréteurs dépendent d'un service de communication pour fonctionner et sont activés lorsque l'application souhaite connaître des informations contextuelles qui ne peuvent pas être découvertes par le dispositif mobile tout seul.

Le composant *NearbyPeopleInterpreter* est un exemple d'interpréteur qui identifie les noms et les profils des personnes présentes à proximité du dispositif mobile. Ce composant dépend des services des composants *MyBluetoothPropertiesSensor* et *BluetoothAroundSensor*. Cet interpréteur renseigne le service d´inférence sociale de *ContextAnnotator* sur l'identificateur du dispositif (fourni par *MyBluetoothPropertiesSensor*) et sur les adresses physiques des dispositifs Bluetooth détectés. Le service de *ContextAnnotator* répond en livrant les noms et l'URI des profils des personnes détectées à proximité, qui appartiennentt au réseau social du propriétaire du dispositif.

D'autres interpréteurs ont été ajoutés au DevAc pour accéder aux divers services d'interprétation et d'inférence de contexte proposé par *ContextAnnotator*.

IV- Aggregator

Les composants d'une application sensible au contexte peuvent accéder directement à chaque service contextuel fourni par les capteurs et interpréteurs, à partir d'un registre de services de DevAc qui est intégré à notre infrastructure à composants. Nous offrons également des services d'agrégation qui fusionnent des données originaires de divers capteurs et interpréteurs. Ces informations sont groupées et représentées en utilisant nos modèles de représentation de contexte. Nous avons établi des agrégateurs pour les cinq dimensions de contexte de l'ontologie ContextTop (spatiale, temporelle, spatiotemporelle, informationnelle, sociale). Ces agrégateurs peuvent fonctionner de deux manières : acquisition de contexte global, et acquisition de contexte local. En mode local, seuls les capteurs et interpréteurs qui n'exigent pas de communication sont exploités. En mode global, tous les interpréteurs et capteurs de chaque dimension, qui ont été déployés sur le dispositif, sont actionnés. DevAC possède également un agrégateur de toutes les informations contextuelles, le ContextAggregator. Cet agrégateur fusionne les données des autres cinq agrégateurs et fournit une représentation en OWL du contexte capturé à un instant précis.

Nous avons utilisé une agrégation dynamique des capteurs et des interpréteurs. À chaque demande d'un client du service d'agrégation, un agrégateur découvre les capteurs et les interpréteurs actifs de sa dimension de contexte, en utilisant le registre de services contextuels. Ainsi, l'ensemble d'informations agrégées peut changer tout au long de l'exécution de l'application, car les capteurs et les interpréteurs peuvent être désactivés ou tomber en panne. Un client d'un agrégateur doit être ainsi capable de traiter l'absence d'informations.

V- DevACReconfigurator

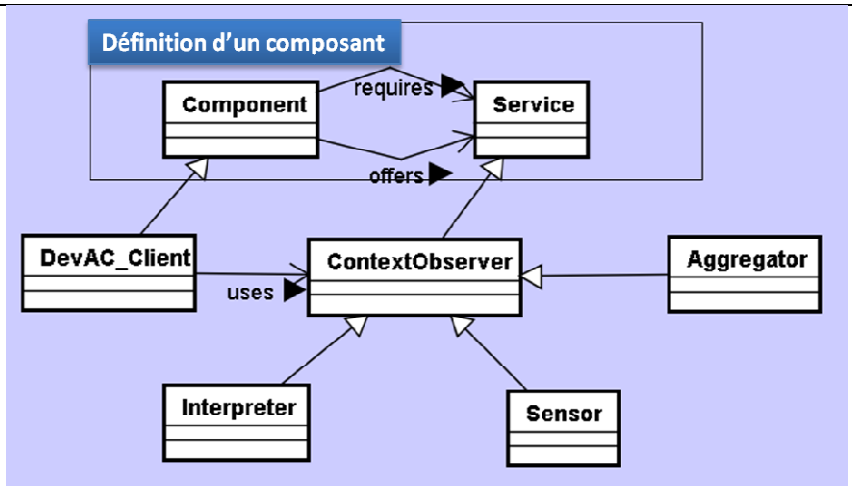
Ce composant contrôle l'activation et la désactivation des composants de DevAC, en fonction des services sensibles au contexte de l'application en cours d'exécution. Le DevAC peut caractériser le contexte de l'utilisateur en accédant à tous les capteurs et aux interpréteurs disponibles sur la version installée de l'intergiciel. Cependant, les composants actifs de l'application peuvent être concernés uniquement par un sous-ensemble du contexte de l'utilisateur, et, selon notre définition de contexte, cet ensemble peut varier au cours de l'exécution. Étant donné que l'acquisition d'informations contextuelles est consommatrice de ressources du dispositif mobile (mémoire, réseau et batterie), nous avons ajouté à DevAC un système de reconfiguration afin de contrôler dynamiquement l'utilisation des capteurs et des interpréteurs. Notre objectif est de diminuer la consommation d'énergie et d'améliorer la performance des applications.

Lors de la description du fichier *manifeste* d'un composant, un concepteur peut indiquer que celui-ci est un client de DevAC, et indiquer la liste des capteurs, des interpréteurs ou des agrégateurs qu'il utilise. La Figure 90 illustre cette extension de la description d'un composant. Le concepteur peut indiquer les services d'acquisition de contexte utilisés. De plus, lors de l'utilisation d'un agrégateur, le concepteur peut indiquer les composants qui doivent être agrégés.

La Figure 90 montre la description du composant TakePhoto qui utilise pour fonctionner les composants de capture de localisation des dispositifs Bluetooth à proximité, ainsi que la

date. Le composant TakePhoto utilise l'agrégateur ContextAggregator pour accéder ces informations.

Lors de l'activation de l'application, le DevACReconfigurator est informé par le conteneur de chaque installation d'un composant de ce genre. En effet, le DevACReconfigurator se transforme en « écouteur d'activation et de désactivation » de ces composants. Lorsque le conteneur doit activer un de ces composants, le conteneur notifie le DevACReconfigurator. Celui-ci active, par la suite, les composants asynchrones de DevAC qui fournissent les informations contextuelles souhaitées par le composant de l'application (par exemple, le capteur *BluetoothAroundSensor* est activé avant l'activation du composant TakePhoto). Le DevACReconfigurator utilise les interfaces de configurations publiées par les composants d'acquisition de contexte afin d'accomplir ce processus.




```
<?xml version="1.0" encoding="UTF-8"?>
<manifest type="DevACClient" >
   <dependends contextClass="WURFL" contextproperty="j2me_mmapi_1_1" operator="equal" value="true" />
   <service name="TakePhoto" classpath="org.steamer.photomap.TakePhoto" type="offers"/>
   <service name="ContextAggregator" classpath="org.steamer.cxtoolkit.agregator.ContextAggregator" type="uses"
only="BTAroundSensor,LocationSensorDelegate,DateTimeSensor" />
   <iletactivator classpath=" org.steamer.photomap.TakePhotoIletActivator"/>
</manifest>
```

Figure 90 – Définition d'un composant client du DevAC et l'exemple du Manifeste du composant TakePhoto.

Une fois que le composant client de l'application est désactivé, le DevACReconfigurator tente de désactiver les composants d'acquisition de contexte requis par le composant client désactivé. DevACReconfigurator accède au conteneur pour vérifier si d'autres composants d'application accèdent à ces composants du DevAc. En cas de réponse négative, les composants du DevAC entrent en processus de désactivation. Certains composants de capture de contexte, tels que celui d'accès au GPS, ne se désactive pas instantanément car les ressources et le temps exigés pour la réactivation de ces composants sont considérables. Dans ce cas, un minuteur est instancié, et uniquement après sont échéance, le composant est effectivement désactivé.

VII- ServiceDelegate

La description des services adaptatifs de notre processus de déploiement (Section 7.1.4) permet d'établir, lors de l'installation d'une application, quel est le composant le plus adapté aux caractéristiques du dispositif mobile de l'utilisateur. Cependant, plusieurs composants

offrant le même service peuvent fonctionner correctement sur le dispositif mobile, et chacun de ces composants peut être mieux adapté que les autres à un contexte d'exécution particulier. Des alternatives de composants peuvent ainsi être déployées sur le dispositif mobile et être inter-changées pendant l'exécution de l'application. Cette approche de reconfiguration dynamique est très utile pour la capture de contexte, car l'implémentation d'un capteur peut dépendre de ressources externes (un point d'accès, un accessoire intégré par Bluetooth) ou mieux fonctionner dans certaines conditions (le GPS ne fonctionne pas à l'intérieur des bâtiments). Ainsi, plusieurs implémentations d'un capteur peuvent être créées dont le coût d'observation ou la qualité des informations observées peut varier selon l'implémentation.

Nous avons introduit dans DevAC ce mécanisme d'échange sensible au contexte de composants par le biais du patron de conception *Business Delegate*[106]. La Figure 91 illustre l'utilisation de ce patron à l'aide de notre infrastructure à composants. Dans le cas où un service possède plusieurs implémentations, alors que les fournisseurs du service souhaitent cacher aux clients cette hétérogénéité, un médiateur de service (*ServiceDelegate*) est créé. Ce nouveau composant contient une classe qui implémente le même service (l'interface Java) et possède un accès à toutes les implémentations du service disponibles à travers le conteneur. Un client potentiel a accès au service seulement à l'aide du médiateur qui choisit, parmi les implémentations existantes, la plus appropriée selon le contexte d'utilisation. Un médiateur peut également démarrer l'échange de composants lorsque l'implémentation actuelle cesse de fonctionner ou qu'une implémentation de plus haute priorité reprend l'exécution. L'infrastructure à composants fournit un mécanisme de passage d'états des variables entre les implémentations, lorsqu'un échange de composants est mis en place (l'interface *IletReconfigurableService*). Des médiateurs ont été créés pour les composants de capture de localisation et pour les méthodes de communication.

La mise en œuvre d'un médiateur de service repose sur un ensemble de mécanismes : découverte de services, identification d'un médiateur lors de la découverte, et changement de variables d'état. Nous avons introduits ces mécanismes directement dans l'infrastructure à composants. Ainsi, les médiateurs peuvent être utilisés par le concepteur d'une application mobile qui souhaite utiliser le mécanisme d'échange transparent de composants. À ce jour, la décision du moment de l'échange et les critères pour la sélection de l'alternative la plus adaptée sont codés directement sur le médiateur.

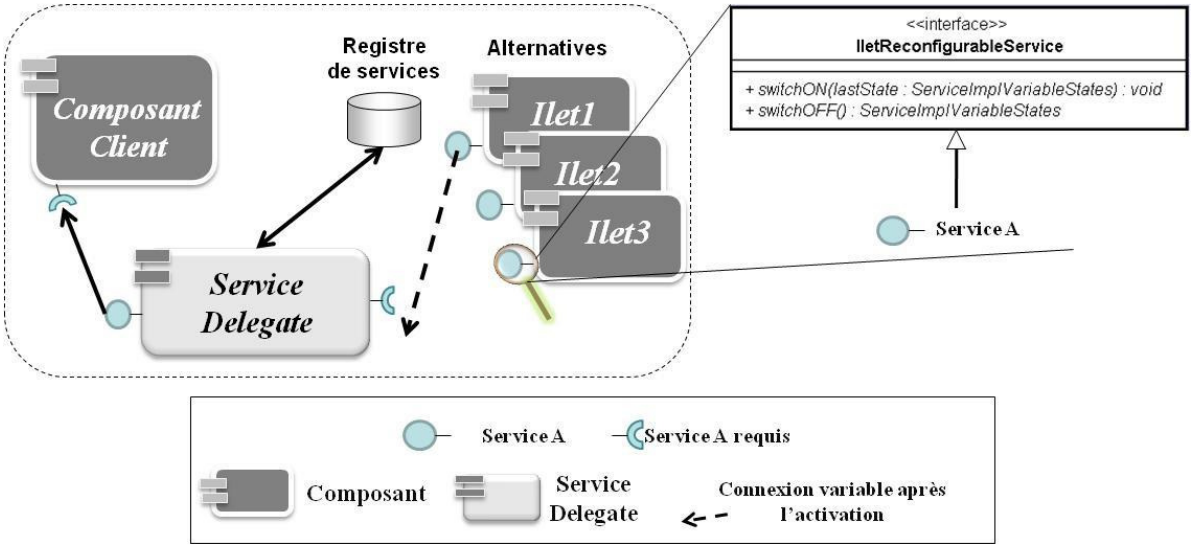


Figure 91 - Reconfiguration Dynamique à l'aide du patron de conception *Businness Delegate*.

[106] http://java.sun.com/blueprints/corej2eepatterns/Patterns/BusinessDelegate.html

Ces six types de composants du DevAC collaborent pour offrir une acquisition de contexte qui s'adapte au long de l'exécution de l'application. Nous décrivons à présent notre algorithme de déploiement adaptatif d'une application sensible au contexte.

7.2.2 Déploiement adaptatif d'une application sensible au contexte

Le plan de déploiement d'une méta-application sensible au contexte doit indiquer son intégration à l'intergiciel afin de bénéficier du déploiement adaptatif de celui-ci. Le concepteur exprime sur l'entête du fichier XML décrivant la méta-application que celle-ci est une cliente du DevAC. Cette information permet au gestionnaire de déploiement de démarrer un processus particulier de génération d'une version. Au long de cette section, nous utilisons l'application de la Figure 92 pour illustrer nos algorithmes de déploiement. La Figure 92 indique également les composants de DevAC utilisés par l'application. En fait, cette application est une extension de l'application exemple de la section 7.1.2 . Nous avons ajouté à l'exemple initial le composant TakePhoto (dont le fichier manifeste est présenté dans la Figure 90). Ce composant utilise quatre services d'acquisition de contexte représentés dans la Figure 92 (ContextAgregator, LocationSensorDelegate, BluetoothAroundSensor, DateTimeSensor). Ces informations contextuelles sont utilisées pour annoter les photos prises par ce composant. Le composant JPEGMap utilise également un service de DevAC : le DisplayMonitor.

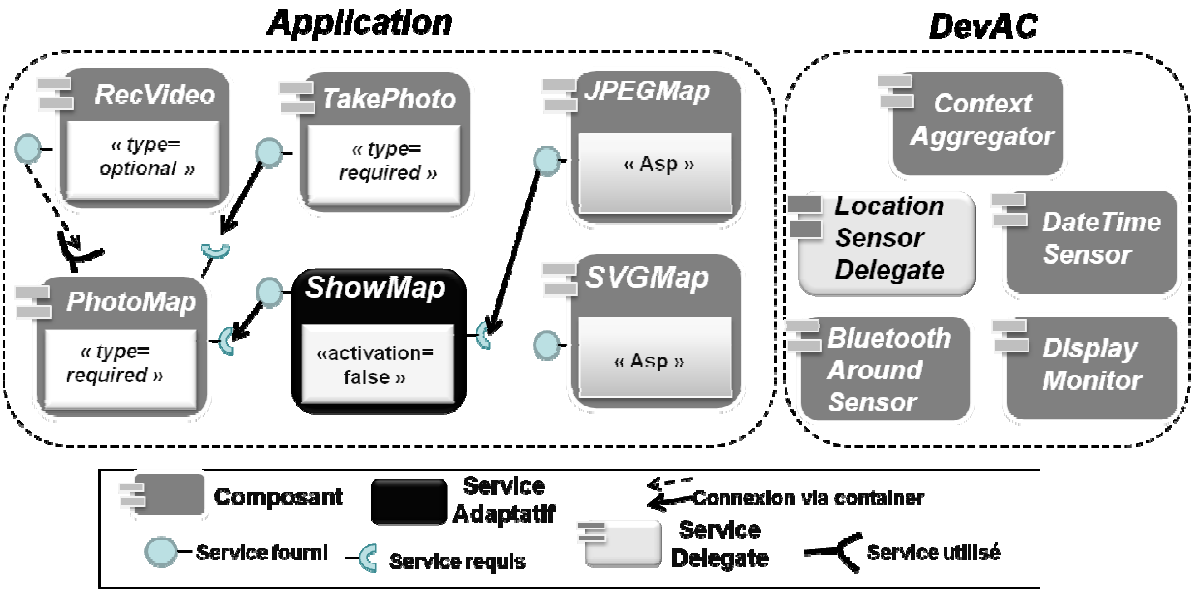


Figure 92 - Exemple d'application mobile et sensible au contexte.

Après la demande d'une application mobile et sensible au contexte, le gestionnaire de déploiement démarre le processus de déploiement illustré dans la Figure 93. Tout d'abord, il active la découverte du modèle de dispositif mobile (étape II) et, ensuite, il génère une version adaptée de DevAC en éliminant les composants dont les dépendances d'exécution ne sont pas satisfaites par les caractéristiques du dispositif mobile. Lors de ce processus, les composants du type ServiceDelegate sont considérés comme des services adaptatifs, c'est-à-dire qu'ils sont éliminés seulement lorsqu'aucune des alternatives ne peut être déployée. Un cache de versions du DevAC pour un modèle spécifique de dispositif mobile peut être créé afin d'éviter la répétition de ce processus pour d'autres applications.

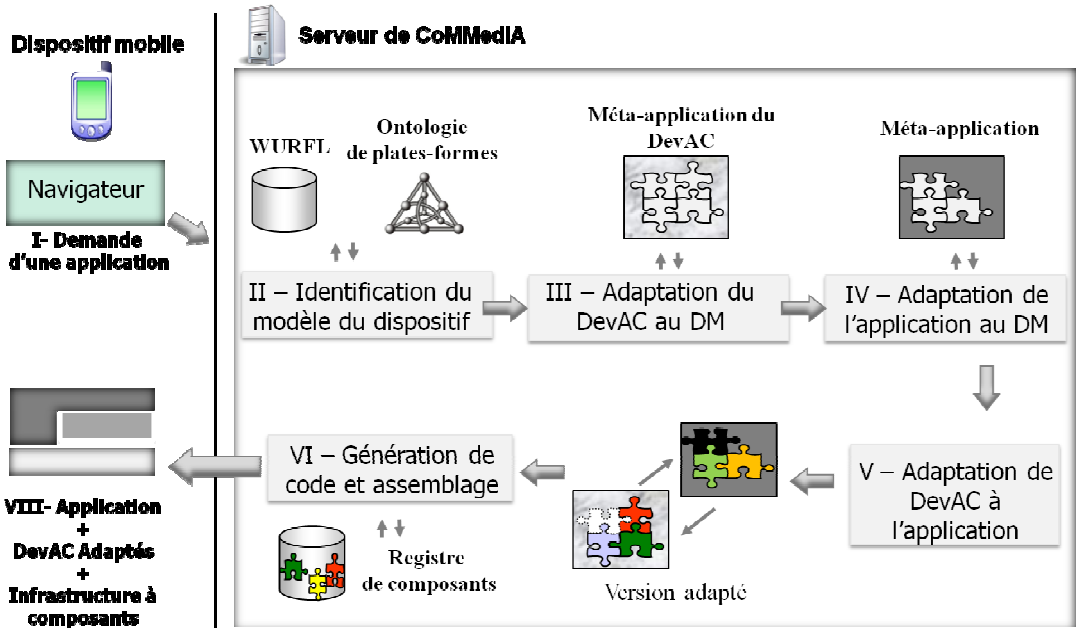


Figure 93 - Processus de déploiement adaptatif d'une application sensible au contexte.

Par la suite, un processus similaire génère une version adaptée de l'application. Le gestionnaire de déploiement utilise l'algorithme de la Figure 86 lors de ces deux processus (étape III et IV). Les relations entre l'application et DevAC ne sont pas considérées dans ces processus.

À partir des deux tableaux LC générés par ces processus, le gestionnaire de déploiement démarre un troisième mécanisme afin d'éliminer les composants de DevAC dont l'application n'a pas besoin pour s'exécuter. Ce processus exclut également les composants optionnels de l'application dépendants des services de DevAC, qui sont fournis uniquement par des composants éliminés lors de l'étape III.

La première activité de ce processus est de construire un CGD global. Ce graphe représente les composants de l'application qui souhaitent ou dépendent de l'exécution de composants de DevAC. Ce graphe est généré à partir de la description des composants de l'application. La Figure 94 illustre le graphe généré à partir de notre méta-application d'exemple. La Figure 94 illustre également le CDG de l'application et une simplification du CGD du DevAC.

Après la création du CGD global, le processus démarre la création d'un tableau (LCglobal) dont la taille est équivalente au nombre de composants de DevAC additionné au nombre de composants de l'application. Les composants de l'application possèdent la même valeur de LC de l'application, tandis que les composants de DevAC possèdent la valeur zéro. L'objectif de l'algorithme de déploiement est d'ajouter 1 à la valeur de composant de DevAC à déployer et 0 aux composants de l'application dépendants de composants de DevAC qui ne peuvent pas être exécutés sur le dispositif mobile. Le pseudo-code de l'algorithme de ce processus est présenté dans la Figure 95.

Pour chaque composant de l'application, nous vérifions s'il dépend d'un composant de DevAC. En cas d'absence de ce composant de DevAC, un processus d'élimination du composant est démarré. Par exemple, le composant JPEGMap dépend du DisplayMonitor pour fonctionner. Lorsque ce composant ne peut pas être déployé, le JPEGMap est éliminé du code final de l'application. Si le composant du DevAc est valide, nous changeons sa valeur du tableau LCglobal pour 1, et nous faisons de même pour les composants dont il dépend. Par exemple, TakePhoto dépend du composant ContextAggregator pour fonctionner, et celui-ci a besoin du déploiement des cinq agrégateurs. Ainsi, lorsque ContextAggregator doit être

déployé, ces cinq agrégateurs sont automatiquement inclus dans la version finale de l'application.

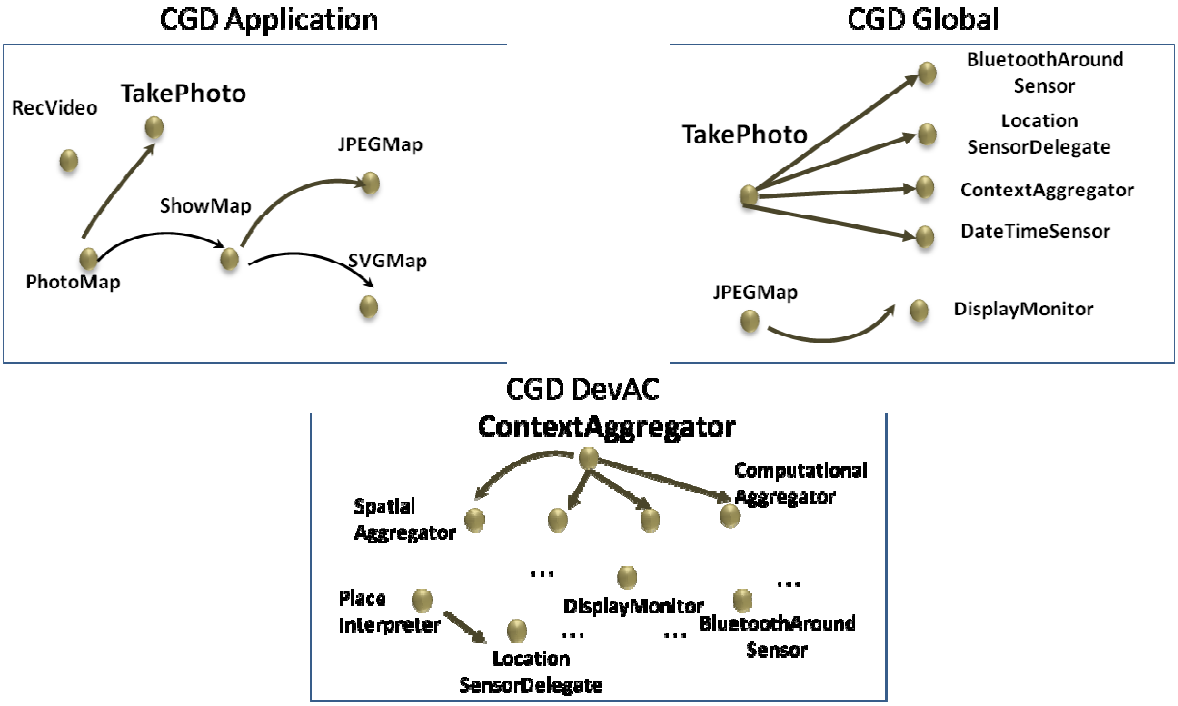


Figure 94- CGD générés pour le processus de déploiement d'une application sensible au contexte.

À la fin de ce processus, un nouveau LC est généré. Ce tableau définit les composants de l'application et du DevAC à déployer. Dans notre exemple, l'interpréteur Place Interpreter ne sera jamais déployé pour cette application. En utilisant ce tableau final, CoMMediA réalise l'assemblage du code et envoie au dispositif mobile une application générée qui est composée des composants propres à l'application, des composants sélectionnés de DevAC, et l'infrastructure à composants.

Les nombres de versions qui peuvent être générées automatiquement par cette méthode de déploiement augmentent sensiblement, notamment en raison des médiateurs de capture de localisation. Sur DevAc, le capteur de localisation possède cinq implémentations : localisation à l'aide de l'API Bluetooth, localisation à l'aide de la Location API, localisation à l'aide de GSM, localisation en accédant à un géocoder via ContextAnnotator, et le médiateur LocationSensorDelegate. Notre application exemple utilise le médiateur, de sorte que tous les composants fonctionnant sur le dispositif doivent être déployés et sont choisis lors de l'exécution. Par exemple, pour un déploiement sur un dispositif ne supportant pas la Location API, tel que le Sony Ericsson K750i, la version GPSLocationAPISensor est éliminée. En revanche, sur le Nokia N95, toutes les versions sont choisies, et celle accédant au GPS à la préférence. Cependant, en pénétrant dans un bâtiment (les coordonnées du GPS ne sont alors plus mises à jour et ont pour qualité 0), le médiateur change l'implémentation, pour le composant GSM et en cas de dysfonctionnement de celui-ci, le médiateur peut dégrader encore plus le service en utilisant le géocoder (une interface graphique s'affiche et demande à l'utilisateur d'indiquer sa localisation actuelle, ou la perception qu'il a de celle-ci…).

```
Algorithme : Génération d'une application sensible au contexte
Donnée : MA, méta-application
         CGDAppli, le CGD de l'application. CGDDevAC, le CGD du DevAC
         LCappli, le LC final de l'application. LCdevAC, le LC final de l'application
Résultat : tableau LCglobal ou null en cas d'échec

Début
CDGglobal := Construire Nouveau CDG (MA) //le nouveau CDG
LCglobal := Construire LCglobal (MA, Lcappli, LCDevAc)//LCglobal contient 0 pour les
composants du DevAC et les valeurs de LCappli pour les composants de l'application
Pour i Allant de 1 à Taille(LCappli)
 //Éliminer les composants de l'application qui dépend d'un service de DevAC qui a été
éliminé par le processus précédent
  Si (Exige Service DevAC (i, CDGglobal) = vrai) Alors
      Tse := Services de DevAC Exigés par (i, CDGglobal)
      Pour y Allant de 1 à Taille(Tse)
        Si  LCDevAC[Tse[y]]= 0 Alors //Composant de DevAC éliminé avant
            Éliminer Composant(LCglobal,i, CDGAppli)
        Sinon //Composant de DevAC présent
            Déployer Composant DevAC (LCglobal, LCDevAC,Tse[y], CDGDevAC)
        Fin Si
      Fin Pour
  Fin Si
  Si (Utilise Service DevAC (i, CDGglobal) = vrai) Alors
      Tse := Services de DevAC Utilisés par (i, CDGglobal)
      Pour y Allant de 1 à Taille(Tse)
        Si  LCDevAC[Tse[y]]> 0 Alors //Composant de DevAC non eliminé avant
              Déployer Composant DevAC (LCglobal, Tse[y], CDGDevAC)
        Fin Si
      Fin Pour
  Fin Si
Fin Pour
Fin
//_______________
Procédure Éliminer Composant(LC,i,CGD)
  Si LC[i]>0 Alors //Éviter les cycles
    Si LC[i] est décrit comme obligatoire dans la méta-application Alors
      échec installation
    Sinon
      LC[i]:=LC[i]-1
      Si LC[i]=0 Alors
         Pour j Allant de 1 à Taille(LC)
                Si CGD[j,i]=1 Alors
                  Éliminer Composant(LC,j,CGD)
               Fin si
         Fin Pour
      Fin Si
    Fin si
  Fin si
Fin Procédure
```

```
Procédure Déployer Composant DevAC(LCglobal,LCDevAC, i, t, CGDDevAC)
//L'objectif est d'affecter la valeur 1 aux composants à déployer
//t indique la position de LCGlobal qui fait référence aux composants de DevAC
  Si LCglobal[t+i]=0 Alors //Éviter les cycles
      LCglobal[t+i]:=1
      Pour j Allant de 1 à Taille(LCDevAC)
                Si CGD[i, j]=1 Alors //le composant i dépend du composant j
                  Déployer Composant DevAC (LCglobal,LCDevAC, j, t, CGDDevAC)
               Fin si
       Fin Pour
   Fin Si
Fin Procédure
```

Figure 95 - Pseudo-code du processus de génération d'une application.

7.3 Conclusion

La gestion sensible au contexte de documents multimédias dépend fondamentalement du développement d'applications mobiles capables d'acquérir des informations contextuelles lors de la création d'un document multimédia. Dans ce chapitre, nous avons présenté des mécanismes de déploiement adaptatif, et un intergiciel auto-adaptatif d'acquisition de contexte, qui visent à faciliter le développement de ce genre d'application.

Tout d'abord, nous avons proposé un mécanisme de déploiement adaptatif qui peut être appliqué à toute application mobile susceptible d'être activée sur divers modèles de dispositifs mobiles. En effet, les portails d'applications deviennent les moyens standards de déploiement d'applications mobiles. Nous avons décrit un modèle de déploiement adaptatif d'applications mobiles qui peut être intégré à ces portails d'applications. Notre approche de déploiement adaptatif offre :

- Un modèle de description de méta-applications qui représente un modèle de base générique pour la génération de versions.
- Une extension d'un mécanisme de découverte automatique des caractéristiques matérielles et logicielles d'un dispositif mobile. Ce mécanisme repose sur l'utilisation d'une description plus sémantique des caractéristiques des dispositifs mobiles.
- Une infrastructure à composants qui offre un support à notre processus de déploiement.
- Une méthode de génération de versions reposant sur l'élimination et la sélection de composants d'une méta-application.

Nous tenons à souligner que notre mécanisme de déploiement n'exige pas l'installation préalable de logiciels sur le dispositif mobile pour fonctionner. Il offre ainsi un modèle de déploiement applicable sur une grande variété de dispositifs mobiles.

Dans ce chapitre, nous avons également proposé un canevas d'acquisition de contexte conçu pour une exécution sur les dispositifs mobiles. Ce canevas repose sur notre infrastructure à composants, et fournit des composants de capture, d'interprétation et d'agrégation d'informations contextuelles. Il introduit des mécanismes de reconfiguration dynamique pour éviter la consommation excessive en termes de ressources du dispositif. Ce canevas possède également un mécanisme de médiateurs qui permet l'échange contextuel de composants de capture offrant un service semblable. Ces deux mécanismes combinés réalisent notre définition de contexte qui repose sur la variation des zones d'observation et zones

d'intérêt d'un système. Ce canevas orienté composants est adapté lors de l'installation d'une application mobile et sensible au contexte. Ce processus de déploiement adaptatif propose donc un usage de l'acquisition de contexte qui diffère de l'approche du tout-ou-rien adoptée par les plates-formes d'acquisition de contexte étudiées. Ainsi, la portabilité des applications sensible au contexte est facilitée.

8 COMMEDIA ET EXPLOITATION DES METADONNEES CONTEXTUELLES

Dans ce chapitre, nous présentons en détail la structuration et l'organisation de l'architecture CoMMeDiA qui intègre l'ensemble de nos propositions pour la gestion de documents multimédias, et pour le déploiement d'applications mobiles et multimédias.

Nous exposons également des mécanismes d'exploitation de métadonnées capturées et enrichies par CoMMediA. Nos propositions profitent des métadonnées formalisées par l'ontologie ContextMultimedia pour offrir de nouvelles manières d'indexer des documents multimédias et de les partager. Les différents stades d'évolution de ces propositions ont été rapportés dans [Viana *et al.*, 2008c] [Viana *et al.*, 2008d][Viana *et al.*, 2009a][Viana *et al.*, 2009b].

8.1 Architecture CoMMediA

En combinant une capture des métadonnées, lors de la création de documents multimédias, aux services de la plate-forme ContextAnnotator, un outil de gestion de multimédias personnels peut associer des annotations contextuelles, textuelles et de contenu aux documents multimédias qui sont gérés par cet outil. Ces métadonnées sont suffisamment riches pour décrire les documents et pour simplifier leur identification et leur partage.

L'apport principal de l'architecture CoMMediA est d'offrir les mécanismes nécessaires pour le développement d'outils de gestion de documents multimédias reposant sur ces métadonnées représentées à l'aide de l'ontologie ContextMultimedia. Les mécanismes proposés vont de la capture de contexte et de documents multimédias jusqu'à l'indexation sémantique de ces documents.

La Figure 96 expose les principaux modules de CoMMediA une fois qu'une application sensible au contexte est déployée sur le dispositif mobile. Une application mobile et multimédia peut utiliser DevAC pour capturer le contexte de création d'un document en utilisant uniquement les capteurs intégrés au dispositif mobile. Un processus *a posteriori* peut se charger d'enrichir le contexte, et exploiter, par exemple, ces informations pour l'indexation du document. CoMMediA offre également l'intégration directe de DevAC à la plate-forme ContextAnnotator. Cette intégration permet de disposer, sur le dispositif mobile, des informations contextuelles qui peuvent être exploitées par les services de partage de contenu.

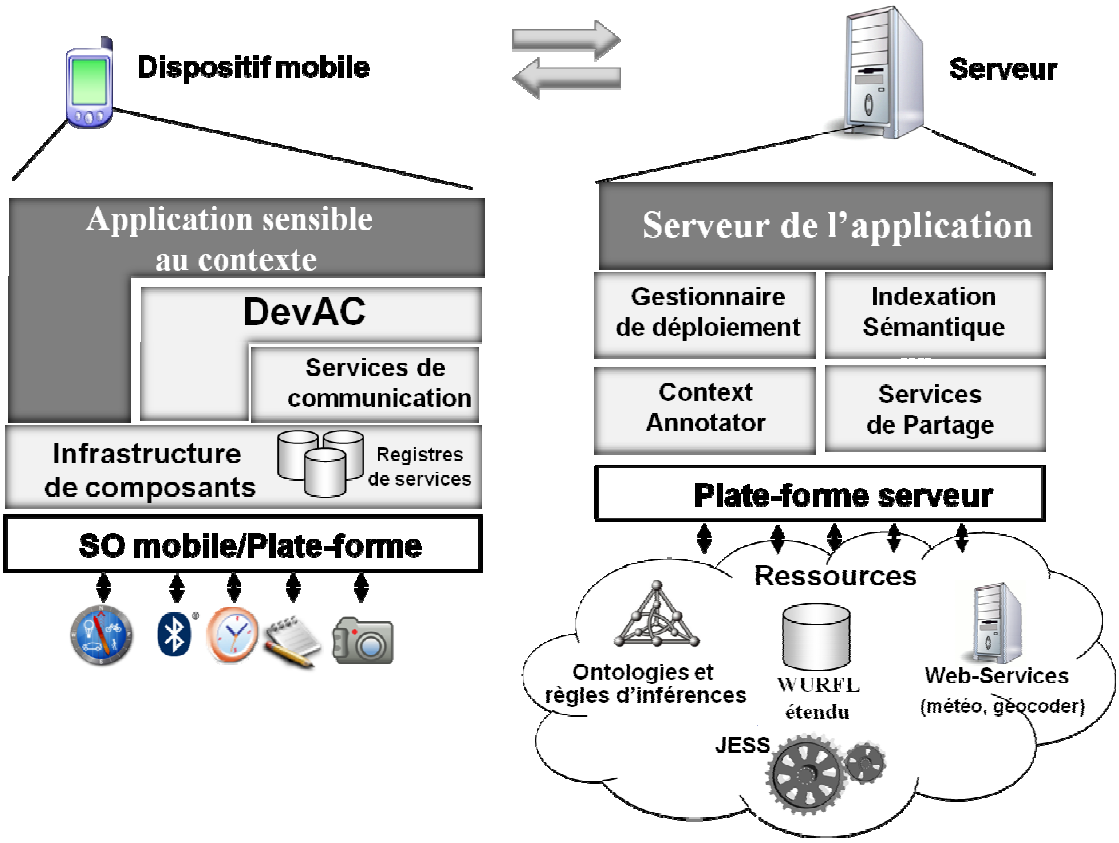


Figure 96 - Architecture CoMMediA après l'installation de l'application mobile.

La suite de ce chapitre est organisée de la façon suivante :

La section 8.2 présente notre proposition d'indexation sémantique et d'expansion de termes. Nous illustrons par un exemple, tout au long de cette section, l'indexation et l'expansion des termes spatiaux d'un document multimédia.

La section 8.3 expose nos propositions de partage sensible au contexte de multimédias enrichis par ContextMultimedia.

8.2 Indexation sémantique de multimédias personnels annotés par ContextMultimedia

8.2.1 Vers des systèmes de recherche contextuelle de documents multimédias

La plupart des systèmes de gestion de multimédias personnels ne prennent pas en compte les métadonnées contextuelles lors du processus de recherche à base de propriétés. Les systèmes d'annotation sensible au contexte exploitent plutôt l'information contextuelle pour l'organisation hiérarchique des documents multimédias, et pour offrir des interfaces de visualisation spatiale de collections de documents à l'aide de cartes. Seul le système MediAssist [O'Hare *et al.*, 2006] utilise l'annotation contextuelle dans la phase d'interrogation. Il offre une interface de recherche par texte et chaque métadonnée contextuelle dérivée est transformée en un terme d'indexation [O'Hare *et al.*, 2007]. L'inconvénient principal de cette approche est la non considération de la relation sémantique entre les documents multimédias et les index, lors de la phase d'interrogation. Il a y donc une perte de la sémantique des index. Or, celle-ci peut s'avérer utile pour supprimer certaines ambiguïtés. De plus, dans cette approche seule la mise en correspondance (« matching », en

anglais) syntaxique est effectuée entre les termes de la requête et les index d'un document multimédia.

Plus généralement, puisqu'il peut s'écouler un certain laps de temps entre la création et le moment de la recherche d'un document multimédia, les utilisateurs risquent d'oublier certaines informations sur le contexte de création. De plus, lorsqu'ils se souviennent de certaines informations sur le contexte de création, ces informations peuvent être partiellement incorrectes ou différentes de celles automatiquement produites par le système de gestion de multimédias. Dans ce type de requête, les comparaisons syntaxiques posent également certains problèmes. Par exemple, si une photo a été prise à la fin de l'hiver, les utilisateurs peuvent se souvenir de cette photo, en pensant qu'elle a été prise au printemps. Un autre aspect de requête imprécise est lié au manque de connaissance des utilisateurs par rapport aux attributs du contexte de création. Par exemple, il est possible de confondre la localisation d'une vidéo créée dans une ville (par exemple, Saint-Denis dans la banlieue parisienne) avec une ville plus connue à proximité (par exemple, Paris). Dans ce cas, les utilisateurs tenterront sans succès de rechercher la vidéo avec le terme « Paris ». En effet, ils n'obtiendront ce document, puisque dans les systèmes d'annotation contextuelle de documents multimédias (tels que ContextAnnotator, MediAssist, etc.), le lieu exact « Saint-Denis » est extrait en employant les coordonnées géographiques et des services « gazetteer » du Web.

Pour clarifier notre proposition d'indéxation, nous décrivons par la suite un scénario de recherche de documents multimédias qui intègre des annotations contextuelles. Dans ce scénario, un utilisateur mobile, un touriste américain Bob, prend une photo du Stade de France (un stade dans la banlieue parisienne). Il utilise une application mobile d'annotation qui accède ContextAnnotator pour dérivés des métadonnées contextuelles. Il se trouve à une distance de 200 mètres du stade. Son téléphone mobile accède au GPS intégré afin d'obtenir les coordonnées géographiques du dispositif au moment de la prise de la photo. Le système d'annotation les transforme en une adresse postale (par exemple, Avenue du Général de Gaulle, Saint-Denis, France). Le téléphone détecte également les dispositifs Bluetooth à proximité et, en utilisant ContextAnnotator, l'application mobile d'annotation peut inférer qu'un des dispositifs détectés appartient à l'amie de Bob appelée April. Supposons que Bob ajoute « stadium » en tant qu'annotation manuelle, et qu'il publie sa photo dans son espace personnel sur un site Web de partage de photos. Plus tard, Bob souhaite revoir la photo du Stade de France où April était présente.

Afin de formuler sa requête, Bob fournit « stadium », « April » et « Paris » comme requête. Nous considérons alors deux possibilités : 1) la recherche textuelle de photos utilise seulement les mots-clés ajoutés manuellement, et, 2) le moteur de recherche utilise l'approche de MediAssist [O'Hare *et al.*, 2007]. Dans le premier cas, le système pourra répondre avec des photos du Stade de France mélangées à des photos de stades où l'équipe de football Paris Saint-Germain a joué (c'est-à-dire, des photos annotées par les mots-clés « Paris » et « stadium »). Dans l'approche MediAssist, les résultats seront encore plus mauvais puisque le système fournira également des photos prises lors du mois d'avril (c'est-à-dire « April » en anglais), et les photos prises effectivement à Paris seront mieux classées dans ce résultat.

Maintenant, supposons que le système d'annotation incorpore pendant le processus d'indexation des métadonnées qui décrivent à la fois le contenu et le contexte de création d'un document multimédia. Ce système peut proposer maintenant des interfaces de requête avancée pour explorer la sémantique apportée par les métadonnées. Afin de raffiner des résultats, le système invite également l'utilisateur à spécifier des catégories de métadonnées pour chaque mot-clé de la requête (par exemple, contenu, contexte spatial, contexte temporel). Ainsi, Bob peut préciser au système que le mot-clé « Paris » décrit une étiquette de localisation, « stadium » indique une description textuelle et « April » correspond à une personne détectée près du lieu de création de la photo. De fait, certains problèmes d'ambiguïté

peuvent être résolus. De plus, si le système trouve peu de résultats, il peut utiliser l'expansion spatiale à partir du terme « Paris » afin de trouver des photos prise proches de Paris qui répondent partiellement à la requête. Par exemple, le système peut employer des ontologies administratives spatiales pour rechercher les villes voisines de Paris (par exemple, Saint-Denis). La photo d'April sera maintenant correctement classée dans le résultat, car le système considérera le lieu de création de la photo comme étant proche du terme « Paris ».

8.2.2 Notre proposition d'indexation sémantique

Les métadonnées contextuelles sont utiles dans le processus de recherche à base de propriétés de documents multimédias personnels, car ces métadonnées sont fréquemment liées aux souvenirs de personnes concernant leurs documents multimédias. Cependant, des métadonnées contextuelles ne doivent pas être uniquement transformées en de simples termes d'indexation.

Quatre aspects sont fondamentaux lors de la conception d'un moteur de recherche de multimédias qui intègre les métadonnées contextuelles :

IV. Les relations sémantiques entre les documents et chacun de ses mots-clés doivent être préservées lors de l'indexation pour éviter certains problèmes d'ambigüité des termes.

V. Le mécanisme de mise en correspondance document-requête doit intégrer cette sémantique.

VI. L'inclusion de mesures de similarité sémantique dans le processus d'indexation peut aider la conception de mécanismes d'expansion de résultats.

VII. Le modèle de requête et l'interface d'interrogation doivent utiliser la segmentation existante sur les métadonnées afin d'exploiter au maximum ces informations.

La Figure 97 illustre notre proposition d'indexation qui prend en compte ces quatre aspects. Nous voulons exploiter les métadonnées enrichies par ContextAnnotator afin de produire des index qui peuvent, ensuite, être utilisés pour l'organisation et pour la recherche à base de propriétés de documents multimédias personnels.

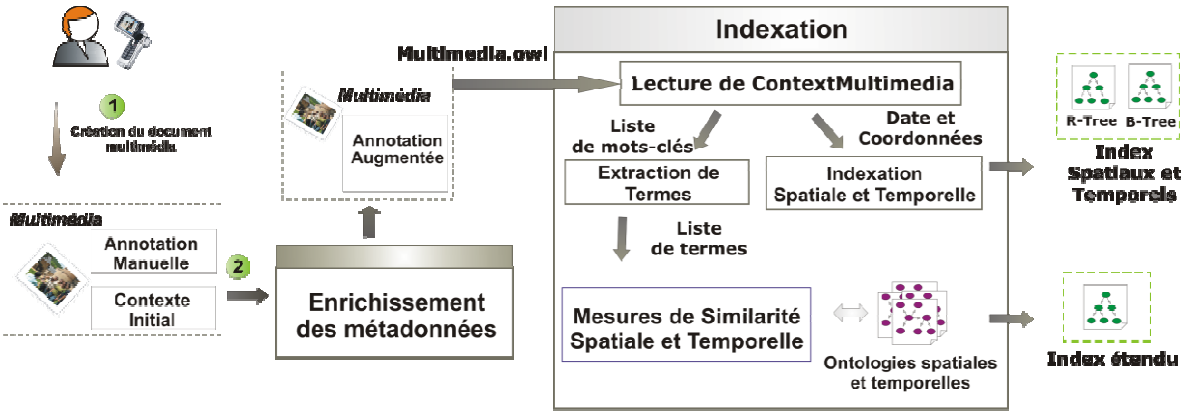


Figure 97- Flux de données lors de l'indexation sémantique proposée.

Les concepts principaux de l'approche sont :

I) L'indexation multi-modèle

Nous explorons les métadonnées des documents multimédias produites à l'aide de ContextAnnotator afin de créer trois types d'index : spatiaux, temporels, et textuels. Les index temporels et spatiaux sont construits, respectivement, à partir de la date et des coordonnées

géographiques associées à chaque document. Ils sont à la base du fonctionnement des interfaces classiques d'organisation et de visualisation de multimédias (telles que une visualisation à l'aide d'une carte de la Terre). Les index textuels sont exploités pour la recherche sémantique à base de propriétés et pour la génération de nuages de mots-clés qui servent également comme interfaces de navigation au sein d'une collection de documents.

Nous extrayons, à partir de l'instance de l'ontologie ContextMultimedia associée à chaque document, un ensemble de mots-clés afin de composer les termes d'indexation. Nous employons l'annotation manuelle fournie par les utilisateurs augmentée de mots-clés choisis parmi les métadonnées contextuelles. Dans une première phase, cet ensemble de mots-clés est exploité pour produire des nuages de mots-clés. Par exemple, la Figure 98 illustre le nuage de mots-clés généré pour une collection de documents créés à Vizille, une ville à proximité de Grenoble. Nous pouvons distinguer l'annotation manuelle concernant le contenu (« Castle ») et les mots-clés contextuels dérivées automatiquement (par exemple, les mots-clés « Vizille », « Château de Vizille », « Spring », et «Karol »).

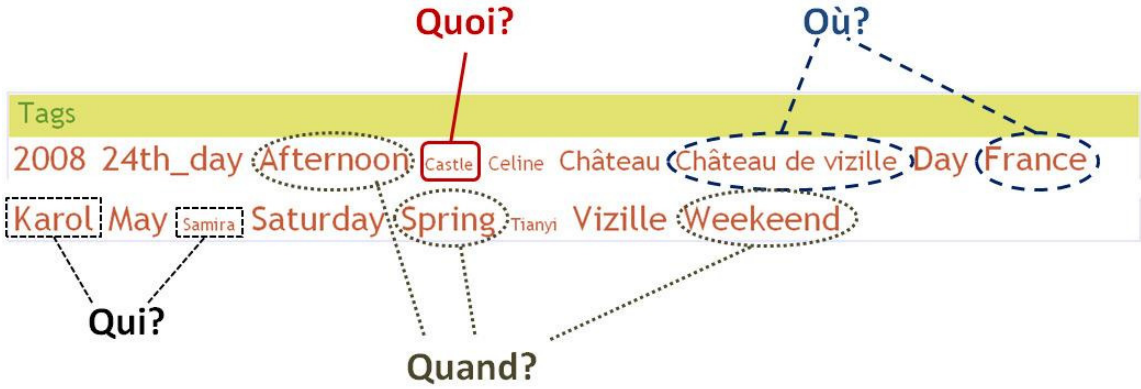


Figure 98 - Nuage de tags généré automatiquement pour une collection créée à Vizille (une ville à proximité de Grenoble).

Dans une deuxième phase du processus d'indexation, nous ajoutons un suffixe pour chaque mot-clé afin de composer un terme d'indexation. Ce suffixe agit en tant que « timbre » sémantique et indique la relation sémantique entre le terme et le document multimédia (par exemple, France.Where). Dans la section 8.2.3, nous détaillons ce mécanisme de timbre sémantique et son incorporation au modèle vectoriel (VSM) d'indexation.

II) Le modèle de requête basé sur la segmentation des métadonnées

En employant notre processus d'indexation, un outil de gestion de documents multimédias peut offrir à un utilisateur l'opportunité d'indiquer, lors d'une requête, la relation entre les documents désirés et les mots-clés qui composent sa recherche. Notre modèle de requête a été prévu pour donner support à une interface simplifiée de requêtes basées sur des textes. Un exemple d'exploitation de notre modèle est une interface de type formulaire à travers laquelle l'outil guide l'utilisateur dans l'expression de la sémantique de chaque mot-clé de sa recherche. La Figure 99 montre un exemple de ce type d'interface. Nous offrons les champs de saisie de texte où les étiquettes indiquent le rapport entre les documents désirées et chaque mot-clé. Les relations disponibles sont inspirées de la définition des diverses dimensions du contexte : « qui », « quand », « où », et « quoi ». Une autre manière d'exprimer cette requête est l'usage d'une seule barre de recherche dans laquelle les utilisateurs peuvent informer la sémantique des mots-clés en utilisant des préfixes tel que sur Google[107]. Par exemple, la requête de la Figure 99 se transformerait en « when:Winter where:Paris where:France who:Paris ».

[107] http://www.google.com/support/websearch/bin/answer.py?hl=en&answer=136861

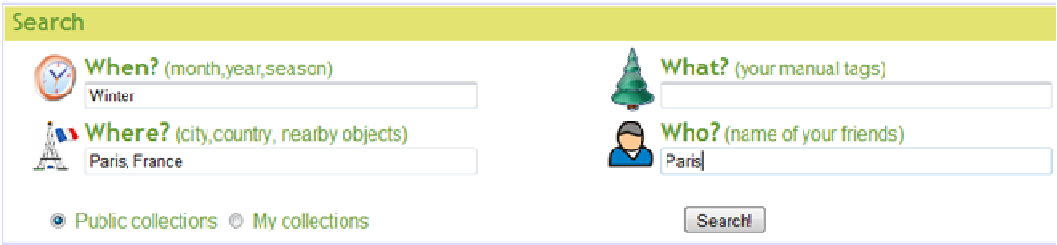


Figure 99 - Un exemple d'interface segmentée pour la recherche contextuelle de documents multimédias.

III) L'extension spatiale des index

Les moteurs de recherche peuvent donner peu de résultats lorsque la requête d'un utilisateur est partiellement incorrecte ou composée de termes proches sémantiquement, mais différents de ceux employés pour indexer le document recherché. Dans ce cas-ci, la majeure partie des systèmes de recherche tentent de proposer des requêtes alternatives. Par exemple, la suggestion d'autres termes ou l'expansion de la requête initiale. Dans ces propositions, le moteur de requête exploite des ressources sémantiques telles que des ontologies et des thesaurus (par exemple, WordNet), afin de réécrire une requête en plusieurs autres, en utilisant des concepts similaires aux termes trouvés dans la requête de l'utilisateur (comme des synonymes, des hyponymes).

La segmentation de l'interface de recherche et la préservation des relations sémantiques entre les termes d'indexation et les documents multimédias indexés fournissent d'autres voies d'expansion de requêtes, car une partie de la sémantique des termes de la requête et des index est connue par le système. Ainsi, au delà d'une expansion par synonyme, des mécanismes d'expansion liés à la dimension contextuelle des termes de la requête peuvent être utilisés telles que l'expansion spatiale, temporelle et thématique (en référence au contenu du document).

La réécriture de requête est l'approche la plus utilisée d'expansion de requêtes. L'inconvénient de cette approche est l'augmentation du temps de réponse liée au processus de réécriture et à l'exécution de plusieurs requêtes [Hamiche *et al.*, 2008]. Pour éviter ces problèmes, au lieu d'une expansion de requête, nous proposons une expansion sémantique des index basée sur les mots-clés. Nous augmentons l'ensemble des termes d'indexation d'un document en employant des ressources sémantiques spatiales. Nous combinons les métadonnées originales du contexte de création d'un document, et le raisonnement spatial afin de dériver une liste de termes potentiellement liés au document multimédia. Nous décrivons cette approche d'expansion spatiale de termes d'indexation dans la section 8.2.6.

8.2.3 Génération des termes d'indexation

Dans notre approche d'indexation, nous employons des annotations manuelles et automatiques pour la recherche et pour le calcul du classement des résultats des requêtes. Notre algorithme de classement est basé sur une adaptation du modèle vectoriel (VSM) [Salton *et al.*, 1975]. Dans le VSM, chaque document répertorié possède un vecteur de poids d'une taille équivalente au nombre de termes du corpus. Ce vecteur contient, pour chacun de ces termes, un poids de pertinence du terme par rapport au document. Le poids d'un terme a une influence directe sur sa puissance de discrimination : plus le poids d'un terme dans le vecteur d'un document est grand, plus haut est le classement du document dans une recherche par ce terme. Dans notre approche d'indexation, un terme est la combinaison d'un mot et d'un « timbre » sémantique (un suffixe). Des mots sont extraits à partir des métadonnées manuelles

et automatiques du document multimédia. Un timbre sémantique représente le type d'annotation (contenu ou contexte) et la connaissance sur la relation sémantique entre le mot et le document (par exemple, une métadonnée temporelle).

La photo de Bob décrite dans la section 8.2.1 aura comme termes : Stadium.what, Saint_Denis.where, France.where, Europe.where, Stade_de_France.where, Sunset.when, Spring.when, 2008.when, May.when, April.who, Saint_Denis.locatedin, France.locatedin, Europe.locatedin, Stade_de_France.veryclose, Stade_de_France.outside, et April.nearby.

Nous utilisons l'annotation textuelle, ajoutée par l'utilisateur (« stadium » dans l'exemple) et les métadonnées inférées par la plate-forme d'enrichissement pour composer les index « étiquetés ». L'idée principale est de toujours employer le VSM pour calculer la similarité entre les requêtes textuelles et les index, sans perdre la sémantique de chaque mot-clé, génératrice du terme d'indexation.

Les suffixes utilisés dans notre proposition sont présentés dans le Tableau 14. La segmentation des métadonnées de ContextMultimedia guide l'attribution des suffixes. Une métadonnée peut générer plusieurs termes étiquetés. Dans certains cas, le suffixe indique simplement la dimension du contexte d'où provient le mot (par exemple, « where » pour les métadonnées spatiales, « who » pour les sociales, etc.). Nous ajoutons également des suffixes apportant plus d'information sémantique tels que « locatedIn », « outside », « presentIn ». Ils sont utiles lorsque la requête d'un utilisateur possède plus de détails concernant la sémantique de chaque terme de la requête. Par exemple, l'utilisateur peut simplement rechercher de photos où April appartient contexte social, ainsi, le moteur de requête doit utiliser « April.who ». Si l'utilisateur recherche les photos sur lesquelles April apparait, alors le moteur de requête doit produire « April.presentIn » comme requête. Le Tableau 14 illustre également les concepts et les propriétés de l'ontologie ContextMultimedia qui sont exploitées pour la génération des termes d'indexation. Les concepts et les propriétés sont liés aux suffixes qu'ils peuvent porter lors de la génération du terme d'indexation. Par exemple, un objet géoréférencé (Georeferenced_Object) peut avoir pour suffixes « where », « outside », « inside » et « veryclose ».

Les annotations textuelles subissent un processus additionnel de lexémisation. L'algorithme de lexémisation de Porter (en anglais, *Porter's stemming*) est utilisé avant la génération du terme final afin de faciliter les recherches futures des utilisateurs. Comme résultat, des mots-clés tels que « flowers » et « flower » génèrent le même terme (« flower ».what).

D'autres relations de ContextMultimedia peuvent être utilisées pour générer des suffixes, néanmoins, le Tableau 14 illustre les concepts et les propriétés que nous utilisons à présent. Ces propriétés peuvent être capturées en combinant une application mobile d'acquisition de contexte à la plate-forme ContextAnnotator.

Nous détaillons dans la prochaine section la formalisation mathématique du processus d'indexation, et les méthodes utilisées pour le calcul de poids de discrimination d'un terme dans le vecteur d'un document multimédia.

Tableau 14 - Suffixes utilisés sur notre approche d'indexation.

Segmentation	Suffixes	Concepts et propriétés	Exemples
S P A T I A L	Where	Georeferenced_Objet et les propriétés du concept Address	France.Where
	LocatedIn	Les propriétés du concept Address	France.Locatedin
	Outside	Georeferenced_Objet	Stade de France. Outside
	Inside	Georeferenced_Objet	Mon_Travail.Inside
	VeryCloseTo	Georeferenced_Objet	Stade de France.VeryCloseTo
T E M P O R E L	When	Les propriétées temporelles et spatiotemporelles de ConextMultimedia: Saison, Month, Year, Day, LighStatus	Sunset.When Winter.When
S O C I A L	Who	Les prénoms des personnes (instances du concept Person) appartenant au contexte social du document	April.Who
	PresentIn	Personne présent sur le document (propriété isPresentedIn est satisfaite)	Karol. presentIn
	Nearby	Personnes appartennant au context social qui ne sont pas sur le document	Ledi.nearby
Mot-clé Textuel	What	Instances du concept *Tag*	Stadium.What

8.2.4 Poids des termes d'indexation

Nous définissons un corpus *P* comme étant un ensemble de documents multimédias, et *T* l'ensemble des termes utilisés pour les indexer. Chaque document multimédia md_i possède douze ensembles de termes dérivés de leurs métadonnées qui sont essentiels dans notre processus d'indexation :

- $What(md_i)=\{t_1, t_2,\ldots, t_n\}$. Cet ensemble possède les termes ($t_i \in T$) générés à partir des métadonnées textuelles (les instances du concept *Tag*).
- $When(md_i)=\{t_1, t_2,\ldots, t_n\}$. Les concepts temporels et certains attributs du concept *PhysicalEnvironment* (tels que la période la journée par rapport au coucher/lever du soleil) génèrent des termes qui sont groupés dans l'ensemble $When(md_i)$.
- $Who(md_i)=\{t_1, t_2,\ldots, t_n\}$. Cet ensemble contient les termes générés à partir des identifiants des personnes (le prénom, un surnom, etc.) qui appartiennent à la dimension sociale du contexte de création du document md_i.

- $Where(md_i)=\{<t_1,d(t_1,md_i)>, \ldots, <t_n,d(t_n,md_i)>\}$ est un ensemble de paires $<t_i,d(t_i,md_i)>$ où le t_i est un terme dérivé des concepts spatiaux qui sont à une distance du lieu de création d'un document inférieure à un seuil donné. La valeur $d(t_i,md_i)$ représente cette distance. Les termes générés à partir des propriétés du concept *Address* possèdent $d(t_i,md_i)=0$, car ces termes doivent indiquer le lieu de création et avoir plus de poids lors de l'indexation.
- $LocatedIn(md_i)= \{t_1, t_2,\ldots, t_n\}$. Cet ensemble possède les termes générés à partir de la hiérarchie des concepts géographiques des propriétés du concept *Address* (tels que le nom de la ville et du pays du lieu de création du document).
- $Outside(md_i)= \{<t_1,d(t_1,md_i)>, \ldots, <t_n,d(t_n,md_i)>\}$ englobe les termes générés à partir du nom des objets géoréférencés et des lieux personnels décrits par des géométries non ponctuelles (par exemple, un polygone) dont le lieu de création de md_i n'est pas à l'intérieur. La valeur $d(t_i,md_i)$ représente la distance entre le lieu de création et le contour ces objets.
- $Inside(md_i)=\{t_1, t_2,\ldots, t_n\}$ englobe les termes générés à partir du nom des objets géoréférencés et des lieux personnels décrits par des géométries non ponctuelles (par exemple, un polygone) dont le lieu de création de md_i est à l'intérieur.
- $VerycloseTo(md_i)= \{<t_1,d(t_1,md_i)>, \ldots, <t_n,d(t_n,md_i)>\}$. Cet ensemble contient les termes générés à partir du nom des objets géoréférencés et des lieux personnels décrits par des points (i.e., des coordonnées géographiques) et qui sont à proximité du lieu de création de md_i. Les termes dérivent de la relation *veryCloseTo* de l'ontologie ContextMultimedia.
- $PresentIn(md_i)=\{t_1, t_2,\ldots, t_n\}$. Cet ensemble contient les termes générés à partir des identifiants des personnes, appartenant à la dimension sociale du contexte de création, qui sont également présentes sur le document md_i (par exemple, leurs visages apparaissant sur une photo).
- $Nearby(md_i)=\{t_1, t_2,\ldots, t_n\}$ groupe les termes générés à partir des identifiants des personnes qui appartiennent à la dimension sociale du contexte de création du document md_i, mais que ne sont pas présentes sur le document (c'est-à-dire, personnes non associées à un segment du document multimédia).
- $ST(md_i) =\{t_1, t_2,\ldots, t_n\}$. Cet ensemble est l'union des termes des ensembles $When(md_i)$,$What(md_i)$,$LocatedIn(md_i)$, $Nearby(md_i)$, $Inside(md_i)$,et $PresentIn (md_i)$.
- $LI_{ext}(md_i) =\{t_1, t_2,\ldots, t_n\}$ définit un ensemble de termes dérivés par le processus d'expansion sémantique.

Une fois la création de l'ensemble $Locatedin(md_i)$ réalisée, nous augmentons l'ensemble des termes qui indexent chaque document. Pour accomplir ce processus, nous employons des ressources sémantiques (c'est-à-dire des ontologies spatiales) pour trouver des concepts sémantiquement proches du contexte de création d'un document multimédia. Nous proposons une expansion spatiale des termes de l'ensemble $Locatedin(md_i)$. D'abord, nous recherchons dans notre base de connaissances le concept c qui correspond au concept le plus profond dans la hiérarchie $Locatedin(md_i)$ (par exemple, pour la photo de Bob, nous recherchons Saint Denis). Les autres concepts dans la hiérarchie seront employés éventuellement pour éliminer des ambiguïtés (par exemple, deux villes avec le même nom). Après cela, nous appliquons une mesure de similarité spatiale $Sim_{SPATIAL}(c,c_{ext},md_i)$ afin de trouver dans cette même ontologie d'autres termes c_{ext} qui sont pertinents pour indexer le document md_i. Avec chaque instance c_{ext} qui satisfait la condition $Sim_{SPATIAL}(c,c_{ext},md_i)>\lambda$, nous produisons deux nouveaux

termes à partir de la combinaison du nom du concept c_{ext} et des timbres *locatedin* et *where*. Nous pouvons alors définir $LI_{ext}(md_i)=\{t_1, t_2,\ldots, t_k\}$ comme étant l'ensemble de termes dérivés de ce processus. Le processus d'expansion spatiale est détaillé dans la section 8.2.6.

Ces ensembles sont utilisés pour la construction du vecteur d'indexation du document. Par exemple, pour la photo de Bob (section 8.2.1), md_b, nous avons :

- $What(md_b)$={stadium.what}
- $Where(md_b)$={<Europe.where,0> , <France.where,0> ,<Ile de France.where,0> , <Agglomeration of Paris.where,0> , <Saint Denis.where,0> , <Stade de France.where,200> }
- $When(md_b)$={ Sunset.when, Spring.when, 2008.when, May.when, Saturday.when}
- $Who(md_b)$={April.who}
- $LocatedIn(md_b)$={Europe.locatedIn, France.locatedIn, Ile de France.locatedIn, Agglomeration of Paris.locatedIn , Saint Denis.locatedIn}
- $VeryCloseTo(md_b)$= {< Stade de France.verycloseto, 200 >}
- $Outside(md_b)=\varnothing$
- $Inside(md_b)=\varnothing$
- $Nearby(md_b)$={April.nearby}
- $PresentIn(md_b)=\varnothing$
- $LI_{ext}(md_b)$={Paris.locatedIn, Courneuve.locatedIn, Paris.where, Courneuve.where }

Après la construction des ensembles de termes, nous démarrons le processus de création des index en utilisant l'approche vectorielle.

Nous définissons le vecteur $|T|$-dimensionnel du document md_i comme:

$$\vec{V}\left(md_i\right) = (w_{t_1 md_i}, w_{t_2 md_i}, \ldots, w_{t_{|T|} md_i} \qquad \text{(I)}$$

où w_{tjmdi} correspond au poids de pertinence du terme t_j par rapport le document md_i. Les poids des termes sont calculés par la formule suivante:

$$w_{t_j md_i} = \begin{cases} idf(t_j),\ t_j \in ST(md_i) \\ idf(t_j) \times InvDist(t_j, md_i),\ t_j \in Where(md_i) \vee t_j \in VeryCloseTo(md_i) \vee t_j \in Outside(md \\ idf(t_j) \times Sim_{SPATIAL}(c, c_{ext}, md_i),\ \ t_j \in LI_{ext}(md_i) \\ idf(t_j) \times DF(t_j, md_i),\ \ t_j \in Who(md_i) \\ 0, otherwise \end{cases} \qquad \text{(II)}$$

La focntion $idf(t_j)$ représente la fréquence inverse du terme t_j qui est calculé comme l'IDF des approches classiques en recherche de documents :

$$idf(t_j) = \log(|P| / ft_j) \qquad \text{(III)}$$

où |P| est la taille du corpus de documents multimédias indexés et ft_j compte combien de fois le terme t_j indexe des documents.

Le $idf(t_j)$ est multiplié par un facteur de pénalisation $InvDist(t_i,md_i)$ lorsque le terme t_j est généré à partir d'un concept spatial qui possède une distance Euclidienne en relation au lieu de création du document (i.e., $t_j \in Where(md_i) \vee t_j \in Outside(md_i) \vee t_j \in VerycloseTo(m_{di})$). La

valeur de ce facteur est inversement proportionnelle à la distance Euclidienne, en kilomètres, entre les coordonnées géographiques du lieu de création du document et la position géographique de l'objet ou du lieu géoréférencé. La formule suivante montre le calcul du facteur :

$$InvDist(t_j, md_i) = \left\{ 1/1+dist(t_j, md_i) \right\} \quad \text{(IV)}$$

Cette méthode de pondération repose sur la supposition que, plus un objet géoréférencé est proche du lieu de création du document, plus il sera pertinent pour l'indexer. Pour les termes dérivés à la suite de l'expansion spatiale, leurs poids dépendent de la similarité avec l'adresse originale du document ($Sim_{SPATIAL}(c, c_{ext}, md_i)$).

Les termes contenant les suffixes « who » subissent également une pénalité : $DF(t_i, md_i)$. L'objectif est de distinguer lorsqu'une personne apparaissait sur le document du cas où elle est simplement présente à proximité. $DF(t_i, md_i)$ est calculé par la formule suivante :

$$DF(t_j) = \begin{cases} 1, & concept(t_j).presentIn \in presentIn(md_i) \\ 0.75, & concept(t_j).nearby \in nearby(md_i) \wedge \\ & concept(t_j).presentIn \notin presentIn(md_i) \end{cases} \quad \text{(V)}$$

$DF(t_i, md_i)$ possède la valeur 1 lorsque la personne est présent sur le document et 0,75 lorsqu'elle appartient au contexte social. Ainsi, dans une recherche en utilisant le suffixe « who », les documents sur lesquels la personne apparait sont mieux classés que lorsqu'elle appartient juste au contexte social.

Dans le cas où un terme n'est présent dans aucun des ensembles de termes, le poids du terme possède la valeur zéro.

Nous illustrons dans le Tableau 15 le calcul du poids pour certains des index de la photo de Bob. Par exemple, si nous utilisons le terme « Brazil.locatedIn » pour chercher cette photo, elle n'apparaitra pas sur le résultat, car le poids de ce terme est 0 sur le vecteur de termes du document de Bob.

Tableau 15 - Exemple de calcule des poids de termes qui indexent la photo de Bob.

Origine du terme	Terme	Valeur du poids
Where(md_b)	t_1=Stade de france.where	idf(t_1).1/(1+0.2)
Where(md_b)	t_2=Saint Denis.where	idf(t_2).1/(1+0)
LocatedIn(md_b)	t_3=Saint Denis.locatedIn	idf(t_3)
LI_{ext}(md_b)	t_4=Paris.locatedIn	idf(t_4). $Sim_{SPATIAL}$(*Saint Denis, Paris)*
Who(md_b)	t_5=April.who	idf(t_5).0,75
Nearby(md_b)	t_6=April.nearby	idf(t_6)
When(md_b)	t_7=2008.when	idf(t_7)
aucun	t_8=Brazil.locatedIn	0

8.2.5 Requête et mise en correspondance

Dans notre modèle de requête, les utilisateurs spécifient les relations sémantiques entre le document multimédia désiré (par exemple, les vidéos ou les photos personnelles qu'ils recherchent) et les mots-clés de la requête. L'utilisateur peut simplement indiquer la dimension des métadonnées à laquelle le mot-clé fait référence (quoi, où, qui, quand), ou il peut utiliser des options avancées pour renseigner des relations plus précises telles que «le mot-clé est le prénom d'une personne sur la photo » ou «le mot-clé est le nom de l'objet près du lieu de création des vidéos». Une requête est ainsi un ensemble de paires

$Q(q_i)=\{<k_1,sr_1>,\ldots,<k_n,sr_n>\}$ où n est le nombre de mots-clés k_J, et sr_j est la relation sémantique entre le mot-clé k_J et les documents recherchés. Nous transformons $Q(q_i)$ en un vecteur $\vec{V}(q_i)=(w_{t_1q_i},\ldots,w_{t_{|T|}q_i})$ où $w_{t_jq_i}$ représente le poids de discrimination d'un terme sur la requête pour discerner les documents appropriés de ceux non pertinents. Afin de construire ce vecteur, la première étape est de créer l'ensemble de termes $ST(q_i)=\{t_1, t_2,\ldots, t_n\}$ à partir des paires $Q(q_i)$.

Puisque nous indexons nos documents multimédias avec des termes produits de la combinaison d'un mot et d'un timbre sémantique, nous devons employer un processus similaire pour les mots-clés de la requête. La relation sr_j définit, en fait, le timbre sémantique à utiliser. Une fois la génération des termes de la requête, nous calculons les poids des termes de $\vec{V}(q_i)$ comme :

$$w_{t_jq_i}=\begin{cases}1\times Desc(t_j), & t_j\in ST(q_i)\\ 0, otherwise & \end{cases} \qquad \text{(VI)}$$

$Desc(t_j)$ est un nombre réel dans l'intervalle] 0.1] qui exprime un facteur distinctif d'un terme parmi les autres sur une requête. L'objectif est d'établir des priorités parmi les termes d'une requête. Des études du comportement de l'utilisateur lors de recherche des documents multimédias ont indiqué que les annotations les plus importantes pour le rappel d'un document sont « qui », « où », et « quand » dans cet ordre [Naaman *et al.*, 2004]. Par conséquent, le facteur distinctif peut être exploité pour exprimer cet ordre de priorité.

Une fois le calcul des poids de termes d'une requête effectué, nous calculons la similarité requête-document pour chaque document de corpus en utilisant la similarité de cosinus :

$$Scores(md_i,q_i)=\frac{\vec{V}(md_i)\bullet\vec{V}(q_i)}{\left|\vec{V}(md_i)\right|\times\left|\vec{V}(q_i)\right|} \qquad \text{(VII)}$$

Les documents qui sont indexés pour au moins un des termes de la requête obtiennent un score positif. Plus de termes un document possède et plus grand est le poids de ces termes dans le vecteur du document, plus grande est la similarité. Le résultat produit par cette fonction de mise en correspondance est utilisé pour le classement des résultats. Un facteur de qualité de chaque document (par exemple, un vote réalisé par les utilisateurs eux-mêmes) peut être appliqué pour changer cet ordre du classement.

Nous tenons à souligner qu'un document multimédia peut être transformé également dans une requête. L'objectif est alors de récupérer des documents créés dans un contexte similaires au « document requête ». Les métadonnées du document requête produisent le vecteur des termes, et le cosinus est appliqué pour trouver d'autres documents dans le corpus dotés de métadonnées similaires.

Nous détaillons dans la prochaine section le processus d'expansion spatiale.

8.2.6 Processus d'expansion spatiale

Comme indiqué précédemment, chaque document multimédia est représenté par un vecteur qui contient les poids de pertinence de chaque terme du corpus par rapport à ce document. Si nous considérons la photo exemple de Bob, la position de chacun des termes dérivés de l'annotation de la photo contient un poids calculé selon la formule (II). Dans les approches classiques, on affecte la valeur 0 à tous les poids des autres termes, ce qui signifie que ces termes ne sont pas appropriés pour indexer le document. Dans notre proposition, nous considérons qu'une partie des autres termes peut être également pertinente pour un document

multimédia, s'ils ont un degré de similarité avec le contexte de création du document. Nous procédons de cette façon afin de diminuer les effets de l'imprécision dans les requêtes de l'utilisateur comme décrit dans la section 8.2.1.

Nous décrivons une approche pour l'expansion de termes spatiaux qui peut être étendue à d'autres dimensions du contexte de création. Les termes spatiaux que nous considérons pour l'expansion de termes sont ceux ayant le suffixe « locatedIn ». Afin d'augmenter l'ensemble de termes, nous proposons de calculer une mesure de pertinence de termes potentiellement relatifs au contexte spatial du document multimédia. De fait, nous appliquons une mesure de similarité entre les concepts qui génèrent les termes et des concepts d'une ontologie spatiale. Nous utilisons deux critères spatiaux pour la mesure de similarité : la sémantique et la distance. La similarité globale est calculée comme :

$$Sim_{SPATIAL}(c, c_{ext}, md_i) = \theta \times SemSim(c, c_{ext}) + (1-\theta) \times \frac{1}{DistSim(c_{ext}, md_i)+1} \qquad \text{(VIII)}$$

où θ et $1-\theta$ représentent, respectivement, les poids de pertinence de la similarité sémantique et de la distance.

Cette formule de similarité globale est inspirée du travail de similarité des concepts spatiaux proposé par Jones *et al.* [Jones *et al*, 2001]. La dimension sémantique de la similarité globale mesure la proximité sémantique des termes spatiaux tels que : équivalence, inclusion, voisinage et appartenance, tandis que la distance est employée pour calculer la distance physique entre deux endroits Afin de considérer les relations sémantiques, nous devons employer une ontologie spatiale qui représente les concepts géographiques d'une région, leurs propriétés et leurs relations sémantiques. La construction manuelle d'une telle ontologie demande du temps et se révèle être un processus consommateur d'énergie. Cependant, de récents travaux ont obtenu de bons résultats dans la génération automatique de ce type d'ontologie spatiale [Buscaldi *et al*, 2006]. Il est raisonnable de considérer que de plus en plus d'ontologies géographiques seront disponibles à l'avenir.

Dans notre approche, l'ontologie géographique est basée sur le modèle GeoNames[108]. Nous définissons un modèle générique, que nous instancions selon les données géographiques disponibles. Ce modèle (Figure 100) représente le concept géographique « Place », ses propriétés (nom, nom équivalent, la géométrie), et les relations qui peuvent exister parmi les concepts (par exemple, « voisin », « partie de » et « capital de »).

Afin de calculer la similarité sémantique spatiale, nous commençons par rechercher dans notre ontologie spatiale le concept *c* qui produit le terme « c.locatedIn » d'un document multimédia (par exemple, « Saint Denis » pour « Saint Denis.locatedin »). Si nous trouvons *c* dans notre ontologie, nous calculons la similarité spatiale entre *c* et les autres concepts du même type (par exemple, région, ville). Les étapes du processus sont :

- Nous employons l'ontologie spatiale afin de rechercher les noms équivalents connus pour un même lieu. Ces noms produiront des termes dont la similarité spatiale avec le concept *c* est égale à 1 (i.e., $Sim_{SPATIAL}(c, c_{ext}, md_i)=1$. Un exemple de cette expansion est le concept « Paris » qui possède le nom alternatif « Capital of France». Ainsi, un document créé à Paris possède les termes « Paris.locatedin » et « Capital of France.locatedin » dont le dernier terme est dérivé du processus d'expansion.
- Nous cherchons l'ensemble des lieux potentiellement connexes à le lieu *c*. Afin de réduire l'espace de recherche, nous considérons seulement les voisins de *c* ou les concepts qui appartiennent à la même unité territoriale de *c*. Les concepts désignés

[108] www.geonames.org

composent l'ensemble *Rc*. Ainsi, de l'exemple de la Figure 100, nous avons Rc (Saint Denis)= {Paris, Vincennes, Courneuve,Versailles}

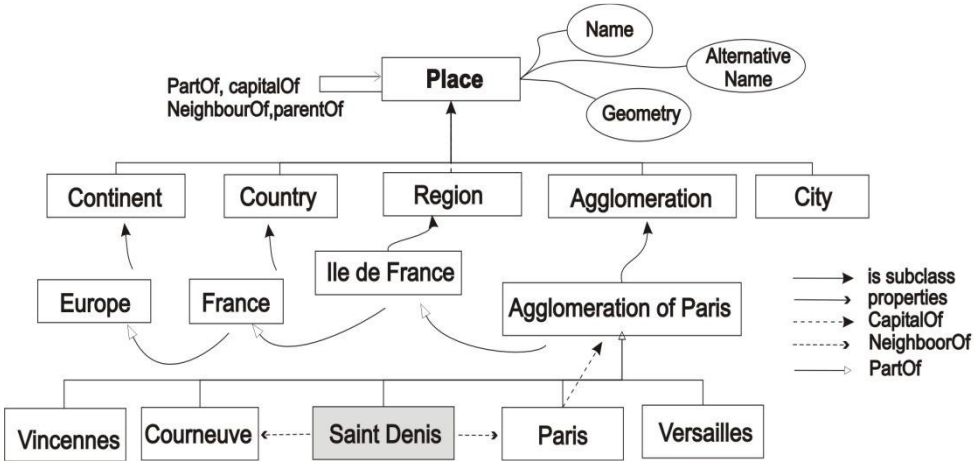


Figure 100 - Une partie de l'ontologie spatiale utilisée pour l'expansion spatiale.

Dans les étapes suivantes, notre but est de réduire l'ensemble initial de concepts de *Rc* et de garder seulement ceux les plus proches spatialement de *c*. Nous calculons ainsi la similarité sémantique entre *c* et les concepts $c_{ext} \in R_c$. La similarité est calculée par la formule :

$$SemSim(c, c_{ext}) = \frac{2 \times depth(lsb)}{depth_{lsb}(c) + depth_{lsb}(c_{ext}) + \alpha + \beta} \qquad \text{(IX)}$$

Pour chaque $c_{ext} \in R_c$, nous calculons *SemSim(c,c$_{ext}$)* en employant une adaptation de la mesure de similarité sémantique proposée par Wu & Palmer connue pour être la mesure de similarité sémantique la plus performante (voir [Liu *et al.*, 2006] pour une comparaison entre les mesures de similarité sémantique). Dans la formule (IX), *lsb* est le premier ancêtre commun aux deux concepts *c* et c_{ext} dans l'ontologie. Par exemple, pour les concepts *Saint-Denis* et *Paris*, le *lsb* est *Agglomeration de Paris*. *Depth(lsb)* représente la distance qui sépare le concept *lsb* et la racine de l'ontologie (i.e., le concept *Place*). $Depth_{lsb}(c)$ et $Depth_{lsb}(c_{ext})$ représentent les distances entre les concepts et la racine de la hiérarchie en passant par *lsb*.

Le facteur α différencie la similarité entre deux concepts spatiaux voisins. En fait, la similarité entre deux concepts voisins (α=0) doit être plus haute que la similarité entre des villes non voisines ($\alpha > 0$). Le facteur β est introduit afin de donner plus d'importance à une unité géographique (β=0), puisqu'en général la préfecture d'une région est davantage connue par les utilisateurs. L'introduction de ces deux facteurs transforme la mesure de similarité en une mesure asymétrique. Par exemple, si un des concepts spatiaux est la capitale d'une unité géographique, nous avons :

$$SemSim(c, capital) \neq SemSim(capital, c) \qquad \text{(X)}$$

Après le calcul de la similarité sémantique, nous mesurons la distance physique *DistSim* (md_i, c_{ext}) entre le lieu de création de chaque document et les concepts $c_{ext} \in R_c$. Afin d'être plus realiste, cette mesure doit considérer le temps de déplacement entre les deux endroits. Cependant, cette approche dépend de la disponibilité de ces informations. Dans notre travail, nous calculons la distance euclidienne entre les coordonnées géographiques du lieu de création et le contour du concept c_{ext}.

Finalement, nous intégrons les deux mesures *DistSim et SemSim* en utilisant la formule VIII. Après le calcul de la similarité globale $Sim_{SPATIAL}(c,c_{ext},md_i)$, nous gardons seulement les concepts c_{ext} qui ont une similarité plus grande qu'un seuil λ fixé.

8.2.7 Exemple d'expansion spatiale

Nous présentons dans cette section un exemple de calcul de similarité spatiale pour la photo de Bob. Afin de calculer la similarité sémantique spatiale (*SemSim*), nous employons une partie de l'ontologie géographique européenne présentée dans la Figure 100. Premièrement, nous avons produit l'ensemble de concepts relatifs à Saint-Denis : Rc= {Courneuve, Vincennes, Paris, Versailles}. Ensuite, nous calculons la similarité sémantique (*SemSim*) entre Saint-Denis et chaque concept de l'ensemble Rc. Nous donnons un exemple du calcul dans le cas du concept Paris. Les valeurs utilisées sont $\alpha = 1, \beta = 2 \text{ and } \theta = 0{,}65$: (voir les formules VII et IX).

$$SemSim(Saint\ Denis, Paris) = \frac{2 \times 4}{5+5+0+0} = 0{,}8$$

Cette valeur indique que le concept spatial Paris présente un haut degré de similarité par rapport à Saint Denis. Ainsi, les documents créés à Saint Denis seront probablement indexés également par le concept Paris. Nous tenons à signaler que, en raison de l'asymétrie de la similarité, *SemSim(Paris, Saint Denis)* possède une valeur inférieure (0,667) et, en conséquence, les documents créés à Paris ont moins de chance d'être indexés par Saint Denis.

Après avoir calculé *SemSim*, nous calculons la composante de la similarité globale, relative à la distance entre le lieu de création du document et le concept de Rc. La photo de Bob a été prise à 200 mètres du Stade de France. Ce lieu est situé à 2.5km du contour de la ville de Paris. Ainsi, $SimDist(md_b,Paris)$ est :

$$Distance(md_b, Paris) = 2{,}5\,km \Rightarrow SimDist(md_b, Paris) = \frac{1}{2{,}5+1} = 0{,}28$$

La similarité globale est calculée par :

$$Sim_{spatial}(Saint\ Denis, Paris, md_b) = 0{,}65 \times 0{,}8 + 0{,}35 \times 0{,}285 = 0{,}61$$

D'autres valeurs de similarité globale:

$Sim_{spatial}(Saint\text{-}Denis, Vincennes, md_b)=0.428$
$Sim_{spatial}(Saint\text{-}Denis, Courneuve, md_b)=0.545$

L'étape finale est le choix des concepts spatiaux dont la similarité est plus haute qu'un seuil donné que nous plaçons à la valeur 0.5. Par conséquent, la ville Vincennes ne sera pas incluse dans l'index de la photo de Bob. Paris et Courneuve sont eux des concepts jugées pertinents pour indexés cette photo et les termes « Paris.where », « Paris.locatedin », « Courneuve.where », et « Courneuve.locatedin » indexent également la photo de Bob. Ainsi, si Bob cherche des documents avec le terme Paris, les photos prises à Saint Denis pourront apparaitre dans le résultat. Grâce au calcul du poids du terme Paris (voir formule (II)), les documents effectivement créés à Paris sont mieux classés dans le résultat. L'ajout d'autres informations dans la requête (telles que « April.who » et « Stadium.what ») vont placer la photo de Bob pour en tête de classement.

La Figure 101 illustre un exemple d'expansion de requêtes. La carte à gauche affiche le résultat de la requête « stadium.what », « Paris.where». La carte à droite illustre le résultat lorsque les termes dérivés de l'expansion sont inclus dans le processus de mise en correspondance.

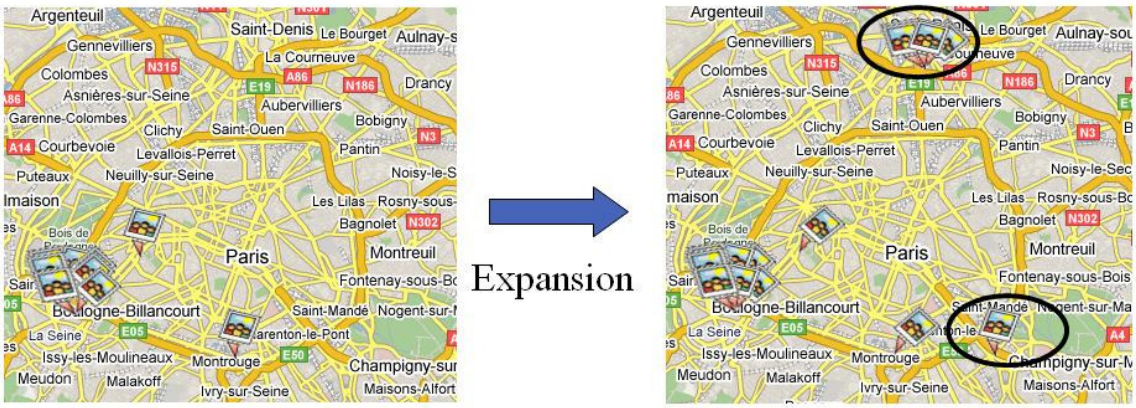


Figure 101 - Exemple d'expansion des termes

Nous détaillons des expérimentations et une évaluation de notre mécanisme de recherche dans le Chapitre 9.

8.3 Partage sensible au contexte de documents multimédias personnels

Le partage de documents multimédias sur les dispositifs mobiles est aujourd'hui une des fonctionnalités les plus prisées sur les applications mobiles et multimédias. Nous avons vu dans l'état de l'art que l'information concernant le contexte de création des documents peut être utile pour améliorer ces services de partage, et que les informations contextuelles peuvent devenir elles-mêmes les contenus partagés.

La richesse des métadonnées d'un document multimédia fournie par ContextMultimedia peut être également exploitée dans ces processus de partage sensible au contexte de documents multimédias personnels. Les métadonnées représentées par ContextMultimedia offrent des informations contextuelles d'un niveau sémantique plus élevé et moins symbolique, au contraire des approches décrites dans l'état de l'art, qui sont, principalement, basées sur les coordonnées GPS ou l'identifiant du réseau cellulaire. De plus, ContextAnnotator fournit un ensemble de services pour générer automatiquement ces métadonnées et évite ainsi que l'utilisateur ait à renseigner manuellement les informations contextuelles à partager.

Dans cette section, nous présentons comment les métadonnées décrivant le contexte de création d'un document multimédia peuvent être exploitées par des services de partage sensible au contexte.

Il s'agit d'abord d'un modèle de notification de publications d'un document multimédia qui exploite les métadonnées contextuelles. Ce modèle offre la représentation de règles de notification qui, une fois activées, permettent à un système de notifier automatiquement d'autres utilisateurs qu'un document multimédia a été publié. Notre approche de partage repose sur les travaux d'adaptation d'Angela Carrillo réalisé dans notre équipe de recherche [Carrillo Ramos, 2007] qui sont également décrits dans l'état de l'art (chapitre 5). Dans ces travaux, des règles d'inférence, une fois satisfaites, déclenchent mécanismes d'adaptation du système multi-agent. Nous utilisons la même approche de règles d'inférence pour déclencher une notification de publication d'un document multimédia.

Notre deuxième proposition de partage sensible au contexte concerne l'assistance à la rédaction de messages qui accompagnent les documents partagés. Cette proposition d'un MMS sensible au contexte consiste en une extension des travaux de génération de messages automatiques [Cemerlang *et al.*, 2006][Koolwaaij *et al.*, 2006]. Nous proposons, en plus d'une simple génération de messages contextuels, un vocabulaire de mots réservés qui fonctionnent comme des raccourcis dont la valeur est un attribut contextuel décrit dans les

métadonnées associées au document partagé. Lors de l'envoi du message, le service de partage sensible au contexte remplace ces mots raccourcis par des attributs contextuels générés en utilisant ContextAnnotator.

8.3.1 Notification sensible au contexte de publication de documents multimédias

Les systèmes sensibles au contexte utilisent fréquemment des règles contextuelles pour définir leur comportement lors d'un changement du *contexte d'utilisation*. Une règle contextuelle définit une mise en correspondance selon le modèle de règles ECA (Event-Condition-Action). Dans une règle contextuelle, *l'événement* représente l'acquisition de nouveaux attributs contextuels, *les conditions* établissent un ensemble d'états contextuels valides, et *l'action* indique le comportement que le système doit suivre si le contexte actuel est un contexte valide, selon les *conditions* de la règle.

Nous nous inspirons de ces approches pour définir un modèle de notification/distribution de documents multimédias reposant sur des règles contextuelles. L'idée principale est d'offrir à un utilisateur **la description préalable de situations qui permettent à un système de partager automatiquement un document multimédia avec des membres du réseau social de l'utilisateur.**

La Figure 102 illustre le schéma ECA de notre modèle de notification. Lors de la publication d'un document multimédia (par exemple, la mise en ligne d'une photo sur l'espace personnel d'un utilisateur), le système de gestion multimédia évalue le contexte de création de ce document afin de démarrer un processus de notification automatique. L'action décrite par la règle contextuelle définit également les personnes à notifier et la manière dont le processus de notification/envoi doit être réalisé (par email, par MMS, …)

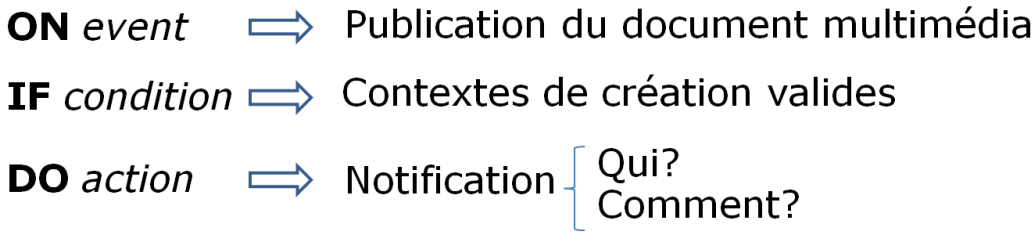


Figure 102- Application du modèle ECA pour le partage sensible au contexte.

Plusieurs manières existent pour représenter les règles ECA. Nous avons opté pour une représentation à l'aide de règles d'inférence. Nous souhaitons ainsi pouvoir exploiter la sémantique des métadonnées de ContextMultimedia, comme nous avons fait pour les services d'inférence contextuelle de ContextAnnotator. De plus, le langage SWRL offre des fonctions mathématiques pour exprimer la comparaison (« = ») et les inégalités (« > » et « < ») entre les propriétés de données qui sont des expressions très utiles pour la description de conditions de partage.

Nous adaptons l'approche des règles contextuelles proposée par Angela Carrillo [Carrillo Ramos, 2007]. Dans cette approche, l'*antécédent* d'une règle d'inférence contient les « faits » (les attributs du contexte capturé) et les conditions d'activation d'une action qui font référence à « ces faits ». Le *conséquent* décrit l'action d'adaptation que le système doit réaliser.

Dans notre approche,

- Les « faits » sont représentés par les métadonnées contextuelles du document à partager ;
- Les *conditions contextuelles* décrivent les contextes de création valides ;

- Le *conséquent* ajoute des relations de notification aux instances des personnes qui doivent être notifiées.

La Figure 103 illustre une partie[109] de grammaire EBNF (*Extended Backus–Naur Form*)[110] des règles contextuelles pour le partage. La règle doit posséder une variable indiquant toute personne du réseau social (*Person(?person)*), et les moyens de notification disponibles dans le système (*Notifier(MMS)*,…). Ensuite, les attributs des métadonnées du document doivent être listés. Ceci peut être optionnel dans la règle, car les moteurs d'inférence offrent d'autres mécanismes d'injection de faits (« **fact** ») tels que la simple indication de la localisation d'un fichier OWL. Les conditions contextuelles peuvent faire référence aux métadonnées spatiales, temporelles et sociales de ContextMultimedia en utilisant le langage SWRL. Après l'exécution des règles contextuelles, les personnes devant être notifiées sont celles qui possèdent une relation de notification valide (c'est-à-dire, une relation inférée par l'exécution des règles contextuelles).

context rule := '(' **facts** ') **conditions** '→' **notification**

facts := 'Person(?person)' {∧ Notifier(**notifier**)} **{[fact]}**

fact := {∧ ('*DataProperty* | *ObjectProperty* | *Individuals* ')'} /* métadonnées sur ContextMultimedia*/

conditions := {∧ ('**atom**')'} /* **atom** comprend les expressions de SWRL[109] */
notifier := 'MMS'| 'Email' | 'EmailLink'
notification := ' hasToBeNotified (?person, true)'

{'∧ hasToBeNotifiedBy (?person, **notifier**)'}

Figure 103 – Une partie de la grammaire EBNF d'une règle contextuelle de partage (*context rule*).

La Figure 104 illustre un exemple de règle contextuelle de partage. Elle établit que chaque fois qu'un document multimédia est publié, et la femme du créateur du document est présente sur son lieu de création, un email avec l'URI de ce document doit lui être envoyé.

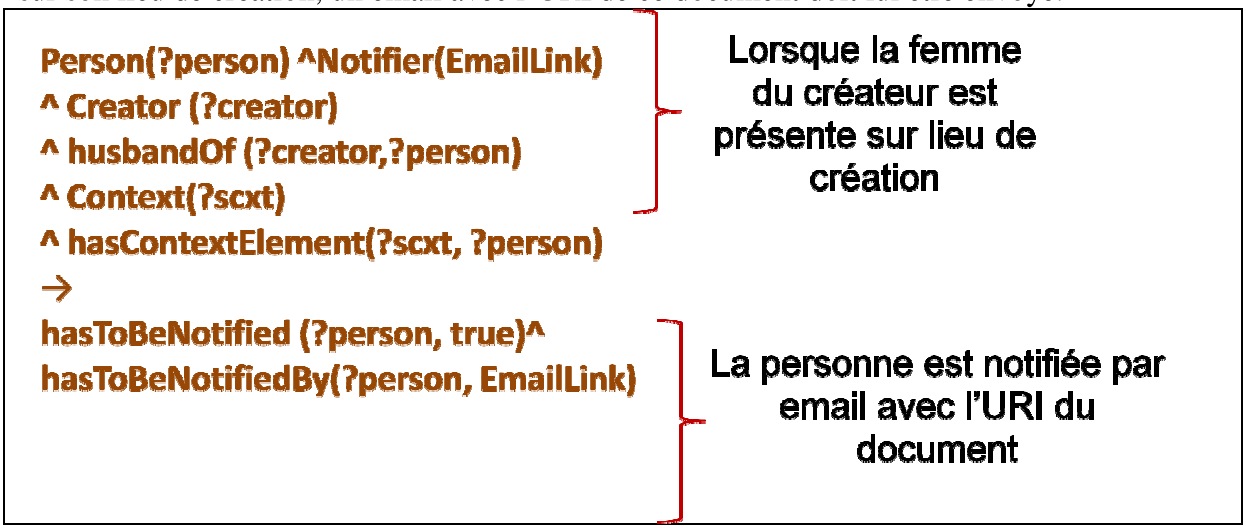


Figure 104 – Exemple de règle de partage (*context rule*).

[109] La grammaire proposée fait référence à des expressions en SWRL et OWL. Pour être complète, la grammaire doit inclure également la définition de **Atom** du langage SWRL(http://www.w3.org/Submission/SWRL/) et les définitions des instances et des propriétés de OWL .

[110] http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=26153

La Figure 105 contient une règle plus élaborée qui établit le partage du document créé au domicile du créateur du document avec toute personne de son réseau social qui était présente lors du jour de création.

```
Person(?person) ^Notifier(EmailLink)
^ Creator (?creator)
^ foaf:knows (?creator,?person)
^ Context(?scxt)
^ hasContextElement(?scxt, ?person)
^ Place(?place) ^ hasContextElement(?scxt, ?place)
^ hasUserDescription(?place, "home")
→
hasToBeNotified (?person, true)^
hasToBeNotifiedBy(?person, EmailLink)
```

Toutes Personnes Présentes

À la maison

Ces personnes sont notifiées par email avec l'URI du document

Figure 105 – Exemple de règle de partage (*context rule*).

Le processus de partage est démarré lorsqu'un utilisateur publie un document sur son espace personnel et signale qu'il peut être partagé. La Figure 108 illustre ce processus qui suit le patron de conception ECA *architectural* d'exécution de règles contextuelles [Dockhorn *et al.*, 2005]. L'entrepôt de multimédias doit récupérer le profil de l'utilisateur et les règles contextuelles. Ensuite, il exécute le processus d'inférence. Le *notificateur* lit l'ontologie résultat et cherche les personnes possédant la relation *hasToBeNotified(?person, true)*. Ce module utilise les profils des personnes pour trouver les informations de notification (telles que leur adresse email).

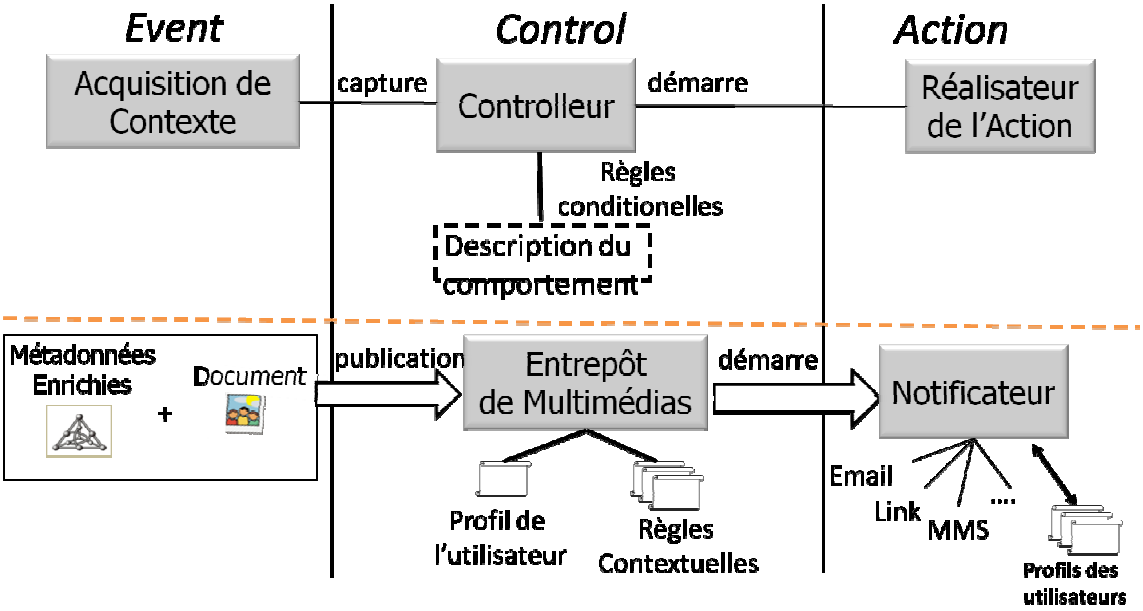


Figure 106 – Le haut de la figure illustre le patron de conception ECAA. Le bas de la figure présente notre architecture de notification.

D'autres règles peuvent été créées selon la grammaire proposée dans la Figure 103 et peuvent, ainsi, profiter de la sémantique offerte par ContextMultimedia. Un système de

gestion de documents multimédias doit offrir des interfaces de création et de modification de ces règles afin de faciliter, aux utilisateurs, la configuration du partage automatique de documents.

8.3.2 Génération automatique de messages contextuels

La création automatique de messages de partage est l'approche de partage sensible au contexte la plus utilisée (Chapitre 3, Section 3.4). Ces approches transforment toutes les informations contextuelles susceptibles d'être découvertes par le système de gestion de multimédia dans un message qui accompagne un document multimédia (tel qu'un MMS). Ces informations sont également exploitées pour générer automatiquement des rubriques sur des blogs personnels [Cemerlang *et al.*, 2006][Koolwaaij *et al.*, 2006]. Ces deux types d'approches peuvent être également implémentées en utilisant une application mobile et notre plate-forme ContextAnnotator, car les informations exploitées par ces messages sont les mêmes que celles utilisées pour la génération des métadonnées d'un document multimédia.

Afin d'illustrer également le potentiel de ContextAnnotator pour la génération de messages contextuels, nous avons mis en œuvre un service de génération de MMS sensible au contexte. La Figure 107 présente une séquence d'exécution de cet exemple d'utilisation de ContextAnnotator : une application mobile qui capture les coordonnées géographiques d'un dispositif, les adresses des dispositifs Bluetooth à proximité, la date et l'heure lors de la prise d'une photo. Cette application se connecte, ensuite, à ContextAnnotator pour dériver des annotations (étape III). Cette application active les services d'inférence spatiale et sociale de ContextAnnotator. Les annotations dérivées sont présentées à l'utilisateur et sont également utilisées pour la génération automatique d'un message d'un MMS qui contient l'adresse, les objets géoréférencés proches, et un bulletin météo courant du lieu où se trouve l'utilisateur (étape V).

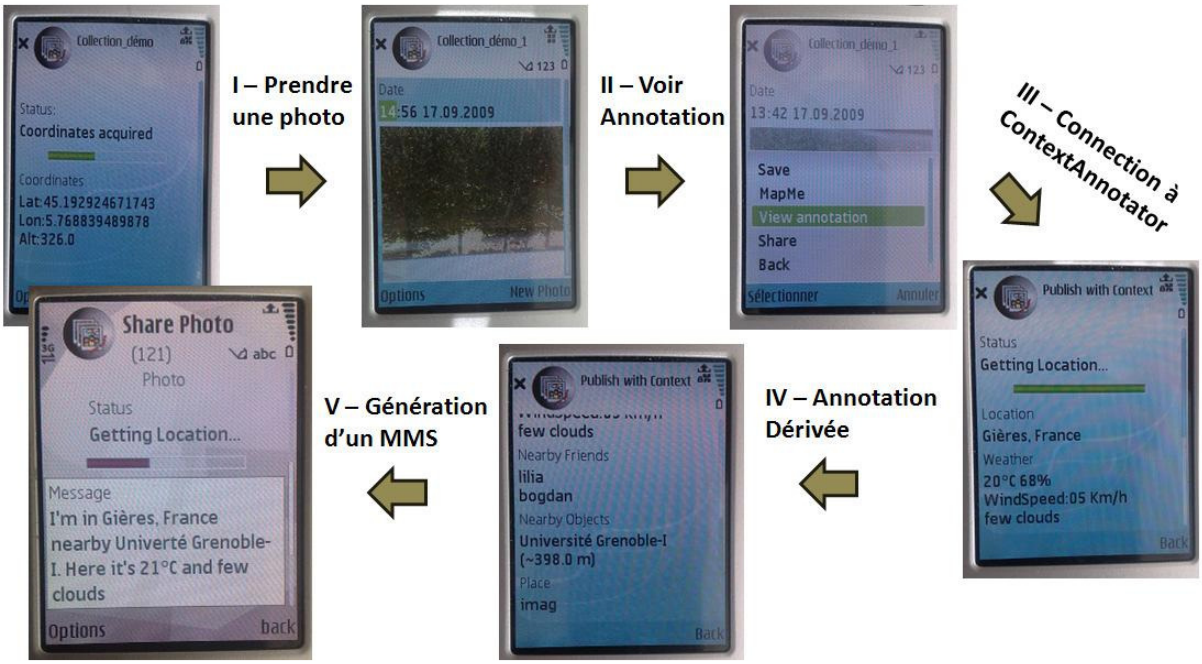


Figure 107 - Application de génération d'un MMS

En vue d'un usage plus générique de ce service de partage et en raison de la faible utilité d'un message de partage prédéfini, nous proposons également une évolution de ce service. L'objectif de cette extension du service de partage sensible au contexte est de simplifier la tâche de l'utilisateur lors de l'écriture du message et de lui offrir une manière de le modifier.

De plus, nous prenons en compte les absences d'informations contextuelles dérivées qui peuvent être récurrentes.

Nous proposons un service de MMS sensible au contexte qui contient un vocabulaire de mots raccourcis qui sont remplacés par les valeurs des attributs du contexte actuel de l'utilisateur. Celui-ci peut écrire un message en incluant les mots raccourcis. Le processeur d'écriture de MMS remplace les mots réservés par l'information obtenue à partir des données dérivées par ContextAnnotator. Tableau 16 montre certains de ces mots réservés et leurs éléments relatifs de contexte. Par exemple, l'utilisateur peut écrire « Bjr ça va ? chuis à **.lc, .no** avec **.np** .Tu viens ? » Le système va générer : «« Bjr ça va ? chuis à **Vizille**, **proche du Château de Vizille** avec **mon ami Elton** et **ma femme Karol**. Tu viens ? »

Mots réservés	Information Contextuelle	Exemple
.ad	Adresse	1, Avenue 1, Paris, France
.lc	Nom du lieu personnel	home, job, place du Tertre
.tm	Date/time	03/04
.wt	Météo	warm (32° C, sunny)
.dr	Durée dans le même lieu	3 heures
.np	Personnes proches et ses relations	Bob, Alice (Bob's friend)
.no	L'objet géoréférencé le plus proche	Tour Eiffel (200m)
.nos	Les objets géoréférencés très proche	Musée du Louvre (inside), Pyramide du Louvre (200m)

Tableau 16 - Mots raccourcis du service de MMS sensible au contexte.

Pendant l'écriture du message, l'utilisateur peut accéder au vocabulaire disponible et aux valeurs actuelles des éléments/des attributs du contexte d'utilisation. Lorsqu'il n'est pas possible que le système dérive une information (par exemple, il n'y a aucun nom de lieu personnel lié à aux coordonnées géographiques), le générateur de MMS propose de remplacer le mot raccourci par une valeur de la même dimension contextuelle, mais avec un niveau sémantique plus bas. Par exemple, si le nom d'un lieu personnel ne peut pas être inféré, le système propose l'adresse comme valeur alternative.

La Figure 108 illustre l'architecture du service de génération de MMS. Le client mobile invoque le DevAC qui capture les informations initiales concernant le contexte de l'utilisateur. Le service de partage envoie à son serveur homologue le contexte acquis sur forme d'une ontologie (instance de ContextMultimedia). Le serveur utilise *ContextAnnotator* pour inférer/générer toutes les informations possibles sur le contexte (la requête explicite l'élément *ContextElement*). Une fois calculées ces informations, le module serveur transforme le résultat de l'ontologie en un tableau contenant les valeurs des mots raccourcis. Ces valeurs sont extraites des relations et des propriétés de données de l'ontologie enrichie.

Le vocabulaire de mots raccourcis peut évoluer en fonction des services d'interprétation et d'inférence disponibles sur ContextAnnotator. Il suffit d'ajouter, sur le serveur de partage, de nouveau mots raccourcis et de les relier aux propriétés et aux relations de ContextMultimedia.

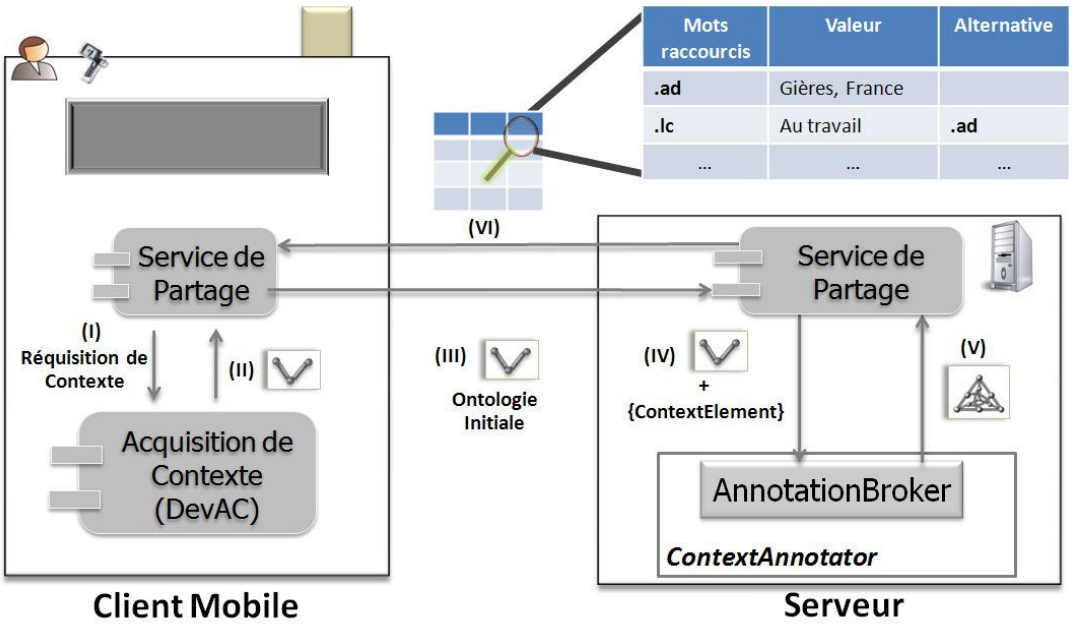


Figure 108 - Architecture client-serveur du service de partage.

8.4 Conclusion

L'apport des métadonnées fournit par la combinaison de ContextMultimedia et de ContextAnnotator ouvre de nouvelles perspectives d'indexation et de partage de documents multimédias personnels. Dans ce chapitre, nous avons présenté des approches d'exploitation de ces métadonnées enrichies, principalement, celles relatives au contexte de création des documents multimédias.

Le processus d'indexation sémantique emploie une adaptation du modèle vectoriel pour l'indexation de documents multimédias. Cinq dimensions – spatiale, temporelle, spatio-temporelle, textuelle et sociale – sont employées pour classifier les métadonnées et pour établir les index par mots-clés des documents. Ce processus maintient une partie de la sémantique de la relation entre les documents et chaque terme en gardant un timbre sémantique. Par conséquent, les utilisateurs peuvent formuler des requêtes qui combinent des mots-clés de différentes dimensions. Dans le chapitre 9, nous présentons des expérimentations réalisées pour mesurer l'utilité d'usage de cette approche d'indexation sémantique et du mécanisme d'expansion spatiale de termes.

Dans ce chapitre, nous avons également montré comment le mécanisme de règles contextuelles, autrefois appliqué pour l'adaptation de systèmes sensible au contexte, peut être utilisé pour la notification automatique de publications de documents multimédias. Les utilisateurs expriment des règles de partage automatique qui font référence au contexte de création des documents. Le système de gestion de multimédia utilise ces règles pour définir quelles personnes seront notifiées. Notre proposition de notification est une évolution des approches de notification basée sur l'annotation manuelle ou sur des données symboliques, car le système profite de la richesse des métadonnées contextuelles formalisées par ContextMultimedia.

9 IMPLEMENTATION ET EXPERIMENTATIONS

À ce jour, nous avons développé des prototypes qui valident les principes fondateurs de nos modèles conceptuels et de nos plates-formes : placer la sémantique et la sensibilité au contexte au cœur des processus de gestion de documents multimédias et des processus d'adaptation d'une application mobile et multimédia.

Afin d'illustrer les apports de nos propositions, nous avons créé un système, mobile et Web, de gestion de photos baptisé PhotoMap. Nous souhaitons illustrer à l'aide de PhotoMap les avantages de l'exploitation de métadonnées contextuelles pour l'organisation, l'annotation, la recherche, et le partage de documents multimédias personnels, sans exiger, en contrepartie, un effort démesuré de configuration ou d'annotation manuelle de la part des utilisateurs finaux. Nous détaillons PhotoMap et ses sous-systèmes dans la section 9.1.

Dans ce chapitre, nous rapportons également deux expérimentations réalisées en utilisant nos prototypes. La première décrit des tests de mesure de l'efficacité de notre mécanisme d'indexation sémantique dans un corpus d'images construit à partir des photos géoréférencées par PhotoMap et par Flickr. La deuxième expérimentation illustre l'impact de notre processus d'adaptation sur le temps d'installation d'une application mobile particulière.

9.1 PhotoMap

Nous avons développé un système, mobile et Web, de gestion de photos personnelles pour valider les divers formalismes introduits par notre approche. Les principaux objectifs de PhotoMap sont de :

- fournir une application mobile pour permettre aux utilisateurs de créer des photos et de les regrouper en des collections (albums) associées à des événements spatio-temporels (par exemple, une collection de photos prises sur un même trajet).
- proposer un système Web qui organise les collections de photos en utilisant leurs métadonnées associées de façon à explorer les annotations spatiales, temporelles et sociales.

- faciliter l'interrogation, le partage, et la visualisation des photos par les utilisateurs sur la base des métadonnées générées automatiquement par les plates-formes d'acquisition de contexte (*DevAC*) et d'enrichissement de métadonnées (*ContextAnnotator*).

L'implémentation de PhotoMap repose sur une architecture distribuée qui est composée de trois sous-systèmes implémentés en Java : *i*) une application mobile reposant sur notre infrastructure de composants, *ii*) une application *desktop* d'aide à l'annotation et, *iii*) un système Web d'organisation de documents multimédias. Cette implementation a été objet d'une démonstration lors de la conférence UBIMOB 2008 [Viana *et al.*, 2008e].

La Figure 109 présente le système de gestion de multimédias une fois mis en place. La Figure 109 illustre également les interactions entre les sous-systèmes et les plates-formes de CoMMeDiA. Nous décrivons à présent les sous-systèmes de PhotoMap.

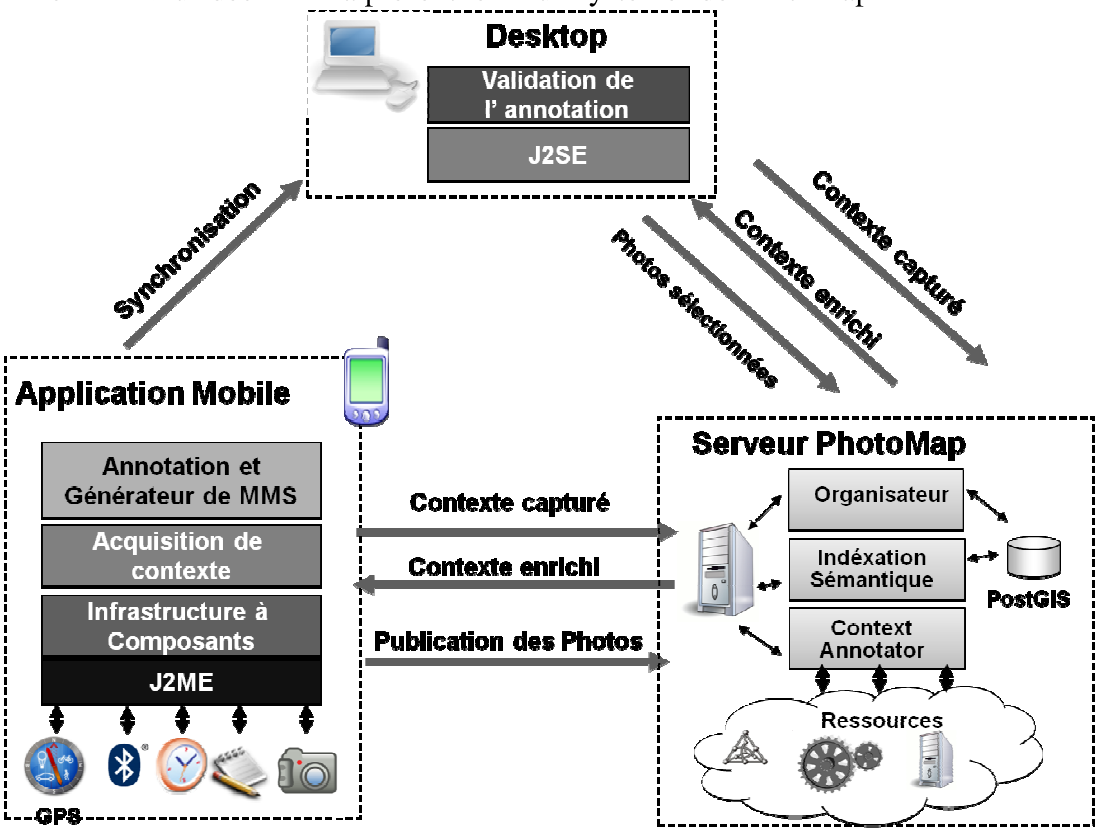


Figure 109 - Architecture du système PhotoMap.

9.1.1 Application mobile

L'application mobile, développée en J2ME au-dessus de notre infrastructure à composants, offre la création, l'annotation et le partage de documents multimédias sur un dispositif mobile. Le principe de cette application mobile est de substituer le logiciel de création de documents multimédias sur des dispositifs mobiles par une version reposant sur la sensibilité au contexte. Les fonctions de prise de photos et d'envoi de ces multimédias restent quasiment inchangées du point de vue de l'IHM (Interface Homme-Machine). Cependant, en amont, l'application mobile utilise nos plates-formes pour produire des métadonnées qui seront essentielles pour automatiser l'organisation, l'indexation et le partage *a posteriori* des photos créées.

L'application mobile permet aux utilisateurs de créer des collections de photos qui sont le plus souvent associées à des événements (par exemple, un voyage touristique, une fête, une promenade en montagne). Ainsi, l'utilisateur définit au préalable, sur le dispositif mobile, une

organisation préliminaire de ses documents. Lors de la création de la collection, l'utilisateur peut l'indexer par un simple nom ou par une description textuelle. Ces annotations sont propagées à chaque document multimédia de la collection lors de son indexation.

Une fois la collection relative à l'événement initialisée, l'application mobile capture la position géographique du dispositif mobile en utilisant les services de DevAC. L'information capturée dépendra des caractéristiques matérielles du dispositif mobile de l'utilisateur, car DevAC a été adapté lors du déploiement de l'application (par exemple, sur le téléphone N95, le composant d'accès au GPS a la priorité). Lorsque le GPS est disponible, les coordonnées capturées (i.e. latitude, longitude, et altitude) sont stockées à chaque changement de position afin de reconstruire le trajet parcouru par l'utilisateur. Ces informations sont représentées en utilisant les concepts *Event_Collection* et *TrackList* de l'ontologie *ContextMultimedia* (chapitre 6).

En l'absence du GPS ou en cas de mauvais fonctionnement, l'application opte pour le non stockage du trajet, en raison de la faible précision des autres composants de capture de localisation (par exemple, l'identificateur du réseau cellulaire), ou en raison de l'impossibilité inconvénient de le capturée en boucle (par exemple, la transformation d'une adresse en coordonnées géographiques a besoin que l'utilisateur informe une nouvelle adresse à chaque déplacement). L'utilisateur peut, néanmoins, ajouter un point de départ à la collection en utilisant les services alternatifs de localisation. La Figure 110 illustre un exemple de cette situation. L'utilisateur indique que la collection est créée à l'intérieur d'un bâtiment (*type of Collection : Indoor*). Le système propose alors à l'utilisateur d'indiquer une adresse (étape I) qui est ensuite transformée en coordonnées géographiques grâce au service *MyAddressToCoords* de la plate-forme *ContextAnnotator*. Les coordonnées géographiques découvertes pourront être utilisées par d'autres services de l'application mobile comme celui illustré par Figure 110 qui affiche sur une carte la position de l'utilisateur (étape IV).

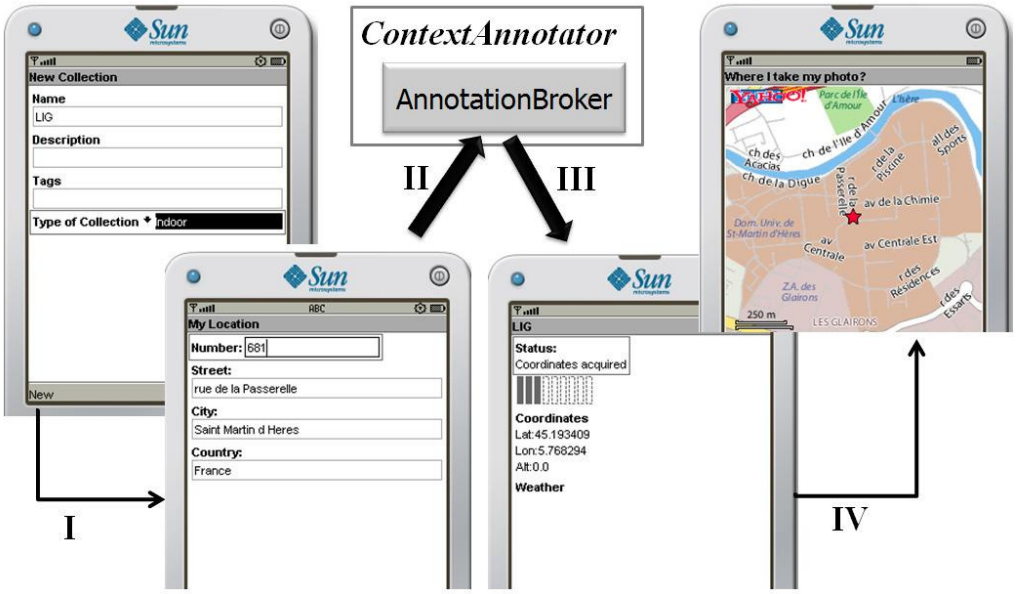


Figure 110 - Association de coordonnés géographiques à une collection en utilisant une adresse postale.

L'application mobile de PhotoMap contrôle également les informations contextuelles qui caractérisent la situation dans laquelle une prise de vue a été réalisée. Lorsque l'utilisateur prend une photo, le client mobile capture à l'aide de la plate-forme DevAC :

- la localisation géographique du dispositif selon le composant de capture de localisation activé ;

- les adresses Bluetooth des dispositifs situés à proximité, si cette fonctionnalité est présente sur le dispositif mobile ;
- les propriétés de l'appareil photo numérique intégré dans le dispositif ;
- la date et l'heure de la prise de vue.

L'utilisateur peut également ajouter des annotations textuelles à chaque photo. Toutes ces métadonnées (contexte capturé et annotations manuelles) sont stockées dans une instance de l'ontologie *ContextMultimedia* lors de la sauvegarde de la photo. L'application mobile s'exécute en *mode déconnecté* la plupart du temps afin d'éviter une consommation batterie excessive causée par le transfert de données. Ainsi, les métadonnées ne sont pas enrichies automatiquement par l'application à chaque prise de vue.

En plus de sauvegarder localement la photo et les métadonnées associées, l'utilisateur peut opter pour la publication directe de cette photo sur le Web, ou pour le partage avec des membres de son réseau social (via e-mail, MMS, etc.). Dans ce deux cas, le système passe en *mode connecté*. Les métadonnées initiales sont envoyées à la plate-forme *ContextAnnotator* qui démarre les processus d'enrichissement d'informations contextuelles. Les métadonnées enrichies sont envoyées à l'application mobile qui les utilise pour l'assistance à la rédaction d'un MMS, ou, simplement, affiche l'annotation sur l'écran du dispositif mobile. La Figure 111 présente une séquence d'exécution de l'application mobile, depuis la création d'une photo jusqu'à l'affichage des annotations dérivées.

Après de recevoir les annotations, l'utilisateur peut les valider et publier le document sur le site de Web de PhotoMap. Nous tenons à souligner que les métadonnées enrichies représentées sur forme d'ontologie (en utilisant ContextMultimedia) sont également envoyées au site Web du système.

Au moment de la publication, le serveur Web démarre également le processus de notification sensible au contexte en exécutant les règles contextuelles de partage décrites par l'utilisateur. Par exemple, **si** l'utilisateur de l'exemple illustré par la Figure 111 a défini par la règle : « *toutes les personnes présentes lors de la création d'un document multimédia et qui appartiennent à mon réseau social, reçoivent une notification »*[111], **alors** Karol et Céline (les personnes détectées à proximité) reçoivent un lien Web de l'URI de la photo publiée.

[111] Person(?person) ^ Notifier(EmailLink) ^ Creator (?creator) ^ foaf:knows (?creator,?person) ^ Context(?scxt) ^ hasContextElement(?scxt, ?person)

→ hasToBeNotified (?person, true)^ hasToBeNotifiedBy(?person, EmailLink)

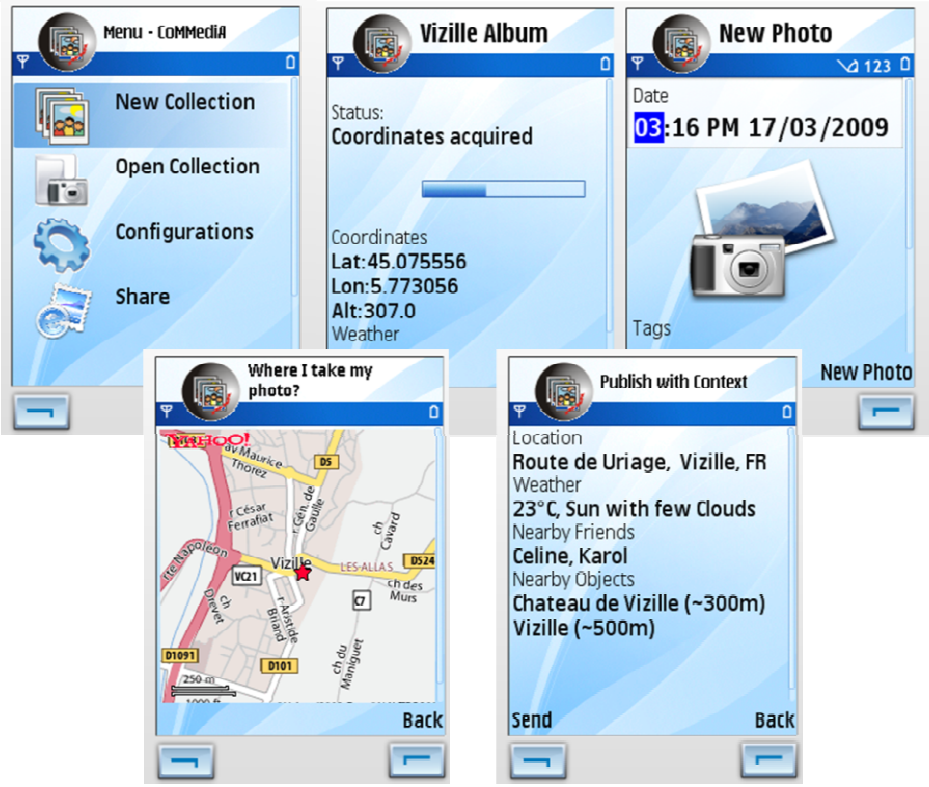


Figure 111 - Capture d'écran du service d'annotation de l'application mobile de PhotoMap.

L'application mobile a été déployée sur les dispositifs mobiles Sony Ericsson K750i (avec un GPS connecté par Bluetooth), et Nokia N95 (disposant d'un GPS intégré). Plusieurs collections des photos ont été créées par des membres de notre équipe de recherche sur diverses villes (Grenoble, Saint Malo, Paris, Rome, Linz, San Francisco …). Ces différentes validations ont permis l'évolution du code de l'application et la correction d'erreurs d'implémentation. Une vidéo de démonstration de l'usage de l'application mobile est disponible sur le site Web http://pyro.imag.fr/PhotoMap/.

9.1.2 Application de bureau d'aide à l'annotation

Le principe de l'application de bureau de PhotoMap est l'assistance à l'annotation de photos. L'application exploite les métadonnées enrichies d'un document multimédia afin d'offrir des interfaces d'annotation dont l'objectif est de transformer la tâche répétitive d'annotation manuelle en une activité de validation d'annotations.

Les utilisateurs qui ne souhaitent pas publier les documents multimédias directement à partir de leurs téléphones mobiles peuvent utiliser l'application de bureau pour cette finalité. La Figure 112 présente le rôle d'intermédiaire que joue cette application. Lorsqu'une collection de documents multimédias sur le dispositif mobile est achevée, le parcours, les photos et les métadonnées associées sont stockés sur le dispositif mobile. Ces fichiers (les images et les ontologies) doivent être transférés sur l'ordinateur de l'utilisateur (par exemple, par un câble USB). L'application de bureau utilise ces fichiers pour générer de nouvelles métadonnées. Les utilisateurs peuvent également sélectionner les photos à publier, et valider les annotations dérivées. Les photos choisies et les métadonnées enrichies sont envoyées au serveur Web du système et seront disponibles sur l'espace personnel de l'utilisateur.

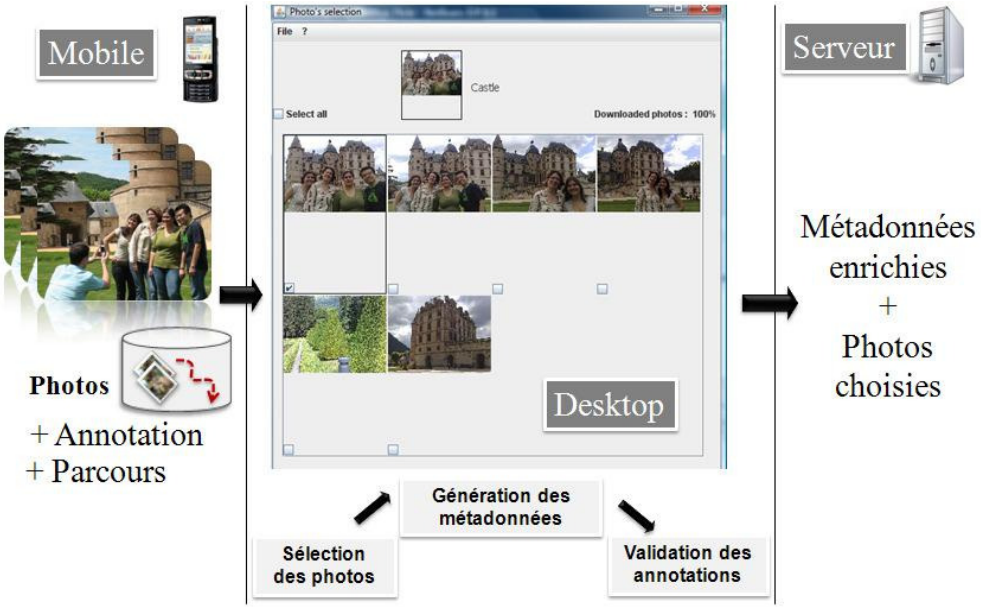


Figure 112 - Le flux d'information dans les sous-systèmes de PhotoMap.

L'utilisateur démarre l'application de bureau en indiquant le fichier *.owl* de l'ontologie qui décrit la collection. Ce fichier possède le nom de la collection attribué par l'utilisateur lors de l'usage de l'application mobile. La Figure 113 présente un exemple de représentation en OWL d'une collection de documents associé à un événement. Des informations décrivant le début de la collection, sa fin, le trajet parcouru, les documents appartenant à la collection, et des annotations textuelles, sont présentes sur ce fichier. Pour le traitement de ce fichier, nous employons la bibliothèque *Jena Semantic Web Framework*[112] disposant de modules d'interrogation d'ontologies.

```
<?xml version="1.0"?>
...
<rdf:RDF  xml:base="http://www-lsr.imag.fr/users/Windson.Viana/ontologies/Castle.owl">
 <ctxtmm:Event_Collection rdf:ID="Castle">
    <time:hasBeginning>
   <time:Instant rdf:ID="collectionInstantBeg">
     <time:inXSDDateTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
     >2008-05-24T16:09:24</time:inXSDDateTime>
   </time:Instant>
  </time:hasBeginning>
  <ctxtmm:documents rdf:resource="http://www.owl-ontologies.com/Castledemo_4.owl#Castledemo_4"/>
  <ctxtmm:documents rdf:resource="http://www.owl-ontologies.com/Castledemo_3.owl#Castledemo_3"/>
  <ctxtmm: documents rdf:resource="http://www.owl-ontologies.com/Castledemo_5.owl#Castledemo_5"/>
  <ctxtmm: documents rdf:resource="http://www.owl-ontologies.com/Castledemo_6.owl#Castledemo_6"/>
  <ctxtmm:textual_annotation>
   <ctxtmm:Tag rdf:ID="tag1">
    <ctxtmm:tag rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Castle</ctxtmm:tag>
   </ctxtmm:Tag>
  </ctxtmm:textual_annotation>
  <ctxtmm:hasTrackList>
   <ctxtmm:TrackList rdf:ID="TrackList">
    <ctxtmm:hasTrackPoint>
     <ctxtmm:TrackPoint rdf:ID="TrackPoint10">
      <georss:point rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >45.074927468665 5.773967955393</georss:point>
      <time:inXSDDateTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
      >2008-05-24T16:12:45</time:inXSDDateTime>
```

[112] http://jena.sourceforge.net/

```
      </ctxtmm:TrackPoint>
     </ctxtmm:hasTrackPoint>
     <ctxtmm:hasTrackPoint>
      <ctxtmm:TrackPoint rdf:ID="TrackPoint0">
       <time:inXSDDateTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
       >2008-05-24T16:09:32</time:inXSDDateTime>
       <georss:point rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
       >45.075733975083004 5.774255203106001</georss:point>
      </ctxtmm:TrackPoint>
     </ctxtmm:hasTrackPoint>
....
   </ctxtmm:hasTrackList>
   <time:hasEnd>
    <time:Instant rdf:ID="collectionInstantEnd">
     <time:inXSDDateTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
     >2008-05-24T16:15:07</time:inXSDDateTime>
    </time:Instant>
   </time:hasEnd>
  </ctxtmm:Event_Collection>
</rdf:RDF>
```

Figure 113 - Le fichier OWL décrivant une collection de documents multimédias.

Après la lecture des métadonnées initiales, le système affiche les photos de la collection en permettant à l'utilisateur de choisir celles qui doivent être publiées (la capture d'écran de la Figure 112 illustre l'interface de sélection de photos). Pour chaque photo sélectionnée, l'application de bureau interroge les services d'enrichissement de métadonnées de la plate-forme ContextAnnotator afin de générer de nouvelles informations (par exemple, adresse, objets voisins, saison, moment de la journée, personnes à proximité…). L'application de bureau met en œuvre également le mécanisme de détection de visages.

Les métadonnées générées par ce processus d'enrichissement sont présentées sur une interface de validation d'annotations. La Figure 114 illustre un exemple de validation de métadonnées d'une photo. Le système a dérivé automatiquement des métadonnées spatiales, temporelles et sociales sur le contexte de création de la photo. Ces métadonnées peuvent être suffisamment riches pour organiser et indexer la collection de photos. En utilisant l'interface de l'application de bureau, l'utilisateur peut modifier les annotations dérivées et ajouter de nouvelles métadonnées. De plus, l'interface de validation offre l'association des visages détectés sur l'image avec les profils de personnes du réseau social de l'utilisateur. L'application de bureau suggère les identifiants des personnes détectées à proximité afin de faciliter ce processus d'association.

Les métadonnées enrichies et validées par l'utilisateur sont représentées à nouveau sous forme d'une instance de l'ontologie *ContextMultimedia*. Cette représentation utilise les concepts de toutes les dimensions contextuelles de notre ontologie.

L'usage de ContextMultimedia pour le stockage des métadonnées enrichies assure la mobilité et l'interopérabilité des annotations générées qui peuvent être exploitées par des moteurs de recherche ou d'autres outils de gestion de documents multimédias. À la fin du processus, l'application desktop envoie les images sélectionnées et les nouvelles ontologies au serveur Web de PhotoMap que nous décrivons dans la prochaine section.

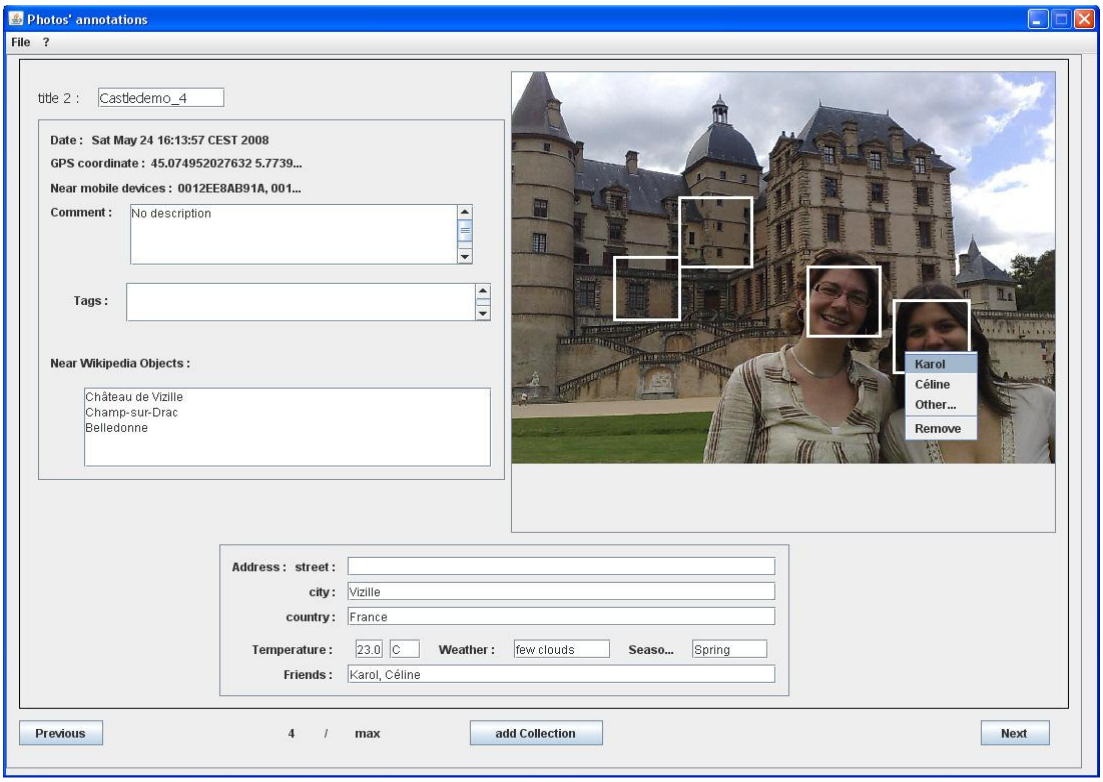


Figure 114 - Interface de validation d'annotations.

9.1.3 Le système Web d'organisation et de publication de photos

Nous avons créé une application Web d'organisation et de visualisation d'images afin d'illustrer l'apport de la sémantique des métadonnées produites et représentées par nos propositions dans la gestion de documents multimédias. Notre objectif est d'explorer des voies d'extension des applications Web existantes pour supporter et exploiter ces métadonnées.

PhotoMap Web a été développé en utilisant J2EE, Struts, Jena, Ajax et Google Maps. Cette application Web offre des interfaces pour l'organisation de collections, la visualisation d'annotations, et la recherche de documents. PhotoMap Web est chargé également des processus d'indexation des documents multimédias et de la notification sensible au contexte.

Au moment de la publication d'une photo ou d'une collection via l'application mobile ou via l'application de bureau, un module du serveur Web de PhotoMap active le processus d'indexation des documents multimédias. La première étape est la génération automatique des index temporels et spatiaux qui sont le support de l'interface d'organisation. Nous optons en transformer les métadonnées, décrits en OWL, en index B-Tree et GiST de la base de données PostGreSQL étendue par son module spatial PostGIS. Nous souhaitons ainsi éviter la lecture séquentielle des fichiers OWL à chaque accès de l'utilisateur, ce que ralentit le temps d'exécution de l'application Web. Nous avons ainsi créé une base de données relationnelle qui stocke les métadonnées de l'ontologie ContextMultimedia et permet de mieux gérer les documents d'une collection. Un module du server lie le ficher OWL en utilisant Jena et insère les informations dans cette base de données. Par exemple, la date et l'heure du début d'une collection décrits par les concepts de OWL:Time deviennent l'index temporel qui classifie en ordre chronologique les « collections événements ». Ces index sont utilisés pour la génération des nuages de mots-clés, et sont aussi le support d'exécution des requêtes spatiales à l'aide de cartes.

La deuxième étape du processus d'indexation est relative à la génération des index timbrés sémantiquement qui sont exploités par l'interface de recherche textuelle de documents. Nous décrivons l'implémentation de ce processus d'indexation dans la section 9.2.1.

La Figure 115 montre la page d'accueil de l'espace personnel de l'utilisateur Bogdan dont le réseau social augmenté est illustré par la Figure 116. L'organisation de cette interface est inspirée des interfaces de visualisation de données des Systèmes d'Information Spatio-temporelle [Moisuc, 2007].

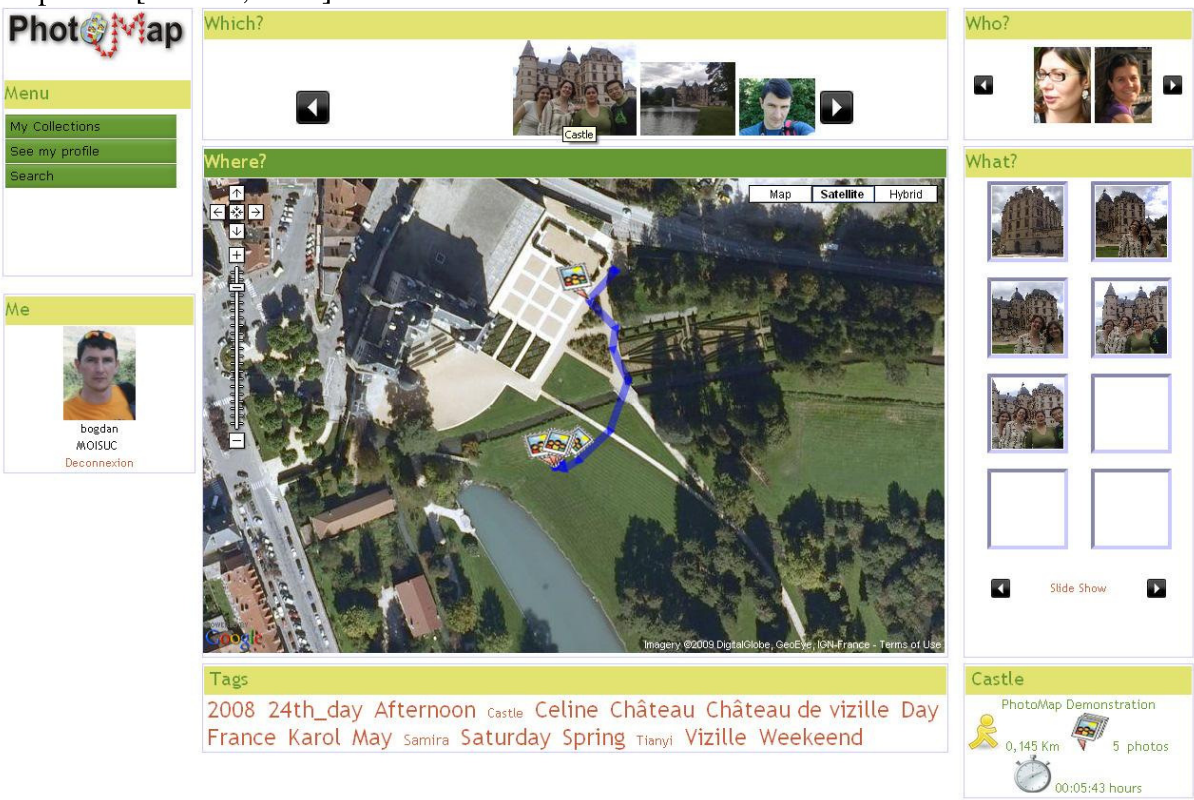


Figure 115 - Page d'entrée de l'application Web.

L'interface est décomposée en régions (des « fenêtres ») qui correspondent chacune à une dimension des données visualisées :

- La fenêtre « *Which ?* » affiche les images symboles de chaque collection événement qui sont présentées en ordre chronologique de la plus récente à la plus lointaine.
- La fenêtre « *Where ?* » présente le trajet parcouru par l'utilisateur tout au long de la création des documents de la collection.
- La fenêtre « *Who ?* » affiche les photos des personnes du réseau social de l'utilisateur détectées à proximité lors de la création de la collection.
- La fenêtre « *What ?* » possède les miniatures des photos de la collection sélectionnée qui peuvent être agrandies pour une visualisation séquentielle (le lien « *slide show* »).
- La fenêtre « Tag » contient le nuage de mots-clés d'indexation générés pour la collection sélectionnée. Un clic sur un mot-clé ouvre une page convenant toutes les images de la collection indexées par ce mot-clé.
- La fenêtre « Info » affiche des informations sur la collection telles que la distance parcourue, la durée, les annotations textuelles. Le titre de cette fenêtre correspond au nom de la collection sélectionnée (par exemple, dans la Figure 115, la fenêtre possède le nom « Castle »).

Cette organisation des collections de photos est réalisée automatiquement en exploitant les métadonnées capturées par l'application mobile de PhotoMap, enrichies par ContextAnnotator, et validées par l'utilisateur à l'aide de l'application de bureau.

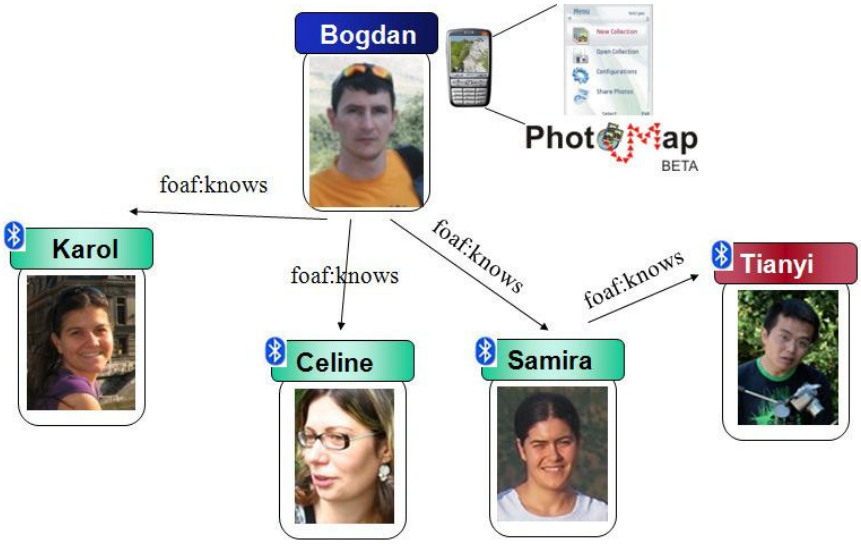


Figure 116 - Réseau social augmenté de l'utilisateur Bogdan.

Une partie des métadonnées d'un document est également utilisée comme fond documentaire de chaque photo dans une fenêtre d'annotation (Figure 117). Notre objectif est d'offrir à l'utilisateur des informations liées au contexte de création des documents multimédias. Nous souhaitons, en utilisant l'annotation, renforcer la mémoire et le souvenir des documents multimédias personnels [Naaman *et al.*, 2004] [Matellanes *et al.*, 2005].

En cliquant sur une des miniatures des photos ou sur l'icône affichant le lieu de prise d'une photo, l'utilisateur peut apercevoir l'annotation associée à chaque document. La Figure 117 illustre un exemple d'annotation d'une photo prise par l'utilisateur Bogdan. Chaque onglet de la fenêtre *popup* d'annotation affiche une dimension des métadonnées. L'onglet « What » présente la photo, les mots-clés ajoutés manuellement, et un lien vers le fichier OWL de métadonnées. L'onglet « Where » contient les informations sur l'adresse (« Vizille, France »), les objets référencées à proximité (« Château de Vizille »), la météo (« 23°, *few clouds* »). L'onglet *Who* affiche les personnes détectées à proximité et leurs relations sociales avec l'utilisateur (par exemple, « Tianyi » est classé dans la catégorie ami d'un ami). L'onglet « When » indique les métadonnées temporelles et spatio-temporelles du contexte de création de la photo (telles que « 2008, *Spring* »).

Ces annotations possèdent un double rôle. Elles accomplissent leur fonction d'activation de souvenirs et fonctionnent en même temps comme un index qui peut être cliqué par l'utilisateur (le système affiche comme résultat toutes les photos indexées par ce mot-clé). Les objets géoréférencés affichés dans l'onglet « Where » incluent également un lien vers la rubrique de Wikipedia qui les décrivent. Ceci permet à l'utilisateur de consulter des informations sur les points d'intérêts qu'il a vu, probablement, lors de son parcours.

D'autres interfaces de configuration de l'application Web sont disponibles sur le système. Ces interfaces offrent la description du réseau social de l'utilisateur et l'ajout de lieux personnels, comme nous avons présenté dans le chapitre 6.

Le site http://pyro.imag.fr/PhotoMap/ fournit l'accès à l'espace personnel de deux utilisateurs. Il est possible de consulter leurs collections de photos et d'interagir avec les interfaces d'organisation, d'annotation, et de recherche.

Figure 117 – Les onglets d'annotation d'une photo.

9.2 Expérimentations

9.2.1 Indexation Sémantique

Afin de valider notre approche, nous avons développé une interface de requête à base de propriétés de documents multimédias personnels. La Figure 118 illustre cette interface avec un exemple de résultat d'une requête. Cette interface a été intégrée dans le système Web de PhotoMap. Pour l'implémentation du mécanisme d'indexation, nous avons employé PostGreSQL et son extension PostGIS. Nous employons Jena, une bibliothèque Java de manipulation d'ontologies OWL et RDF, pour lire les métadonnées des documents multimédias (les instances de ContextMultimedia) lors du processus d'indexation qui génère trois index pour chaque document (temporel, spatial, et textuel). Chaque fois qu'un document est ajouté dans la base, l'*idf* (Section 7.1.4, formule (2)) de chaque terme qui l'indexe est recalculé. La mise en correspondance document-requête (calcul du cosinus) est faite à l'aide

de procédures stockées de la base de données, en s'inspirant de l'implémentation du modèle vectoriel (VSM) sur une base de données relationnelle, proposée par [Becker *et al.*, 2003].

Lors de l'implémentation du processus d'expansion spatiale, nous avons employé une base de données géographique contenant les unités territoriales de la France pour la construction de notre ontologie spatiale. Cette base de données contient le nom, les contours géographiques, et le statut administratif (par exemple, préfecture de département) de 36656 villes françaises. À partir de cette base de données, nous avons pu générer notre ontologie spatiale et calculer les similarités sémantiques *SemSim* entre les concepts de l'ontologie. La génération de l'ontologie a pris 1h 30 sur un ordinateur de bureau standard (2.4 gigahertz et 1G de RAM) et peut être appliquée sur d'autres bases de données contenant les contours des villes d'un pays. Nous avons exporté les résultats dans des fichiers d'OWL-DL qui instancient l'ontologie de GeoNames.

Nous avons employé les photos existant dans PhotoMap (un ensemble de 400 photos automatiquement géoréférencées) afin de calibrer le processus de l'expansion spatiale. Nous avons obtenu comme meilleures valeurs de paramètres pour le calcul de la similarité :

$$\alpha = 1,\ \beta = 2\ and\ \ \theta = 0{,}65$$

Cependant, le nombre de documents multimédias disponibles sur PhotoMap n'est pas, pour l'instant, suffisamment représentatif pour valider notre approche d'indexation sémantique de documents multimédias. Nous avons ainsi décidé d'augmenter notre corpus avec des photos importées de Flickr. Nous avons développé une application *desktop* en Java qui télécharge des photos géoréférencées en employant l'API Flickr[113] et génère des instances de ContextMultimedia contenant les métadonnées : la date et l'heure de création, les coordonnées géographiques et les annotations manuelles de chaque photo. Cette application fournit, en conséquence, les mêmes informations que nous obtenons à partir de l'application mobile de PhotoMap. Nous avons construit un corpus de 8007 photos avec 10630 termes et 138 termes générés par l'extension spatiale.

La première expérimentation que nous avons réalisée mesure les avantages de l'utilisation des métadonnées contextuelles pour la recherche de documents multimédias personnels. Nous avons choisi dix photos de l'ensemble de PhotoMap et nous avons tenté de les retrouver dans le nouveau corpus en employant différentes combinaisons de termes contextuelles.

Nous avons calculé le classement moyen (en anglais, *mean rank - MR*) et le classement moyen réciproque (en anglais, *Mean Reciprocal Rank-MRR*) [Voorhees *et al.*, 1999] afin de découvrir, parmi les différents sous-ensembles de métadonnées d'une photo, quels sont les meilleurs pour rechercher une photo. Le classement moyen est une mesure plus simple à comprendre, il indique la position moyenne dans le classement de la première bonne réponse du résultat, dans notre cas, la seule bonne réponse. Par exemple, *6,2* de classement moyen indique que la image recherchée apparait, en moyenne, entre la sixième ou septième position du classement. Le classement moyen réciproque est la moyenne de l'inverse du rang de la première bonne réponse. Cette mesure apporte une valeur entre 0 et 1 et peut être plus facilement utilisée pour comparer les différentes requêtes.

Le Tableau 17 montre les résultats des expérimentations. En utilisant uniquement l'information temporelle ou le nom d'une ville dans une requête, les résultats présentent les plus mauvaises MRR (0.054 et 0.12) avec un meilleur résultat pour l'information spatiale. Le problème d'une requête avec un nom de ville est que la majeure partie du corpus se concentre sur quelques villes (Paris, Grenoble, Saint Denis, Vizille, Marseille, Saint-Malo). Par conséquent, un nom de ville n'est ici pas suffisant pour distinguer une photo. Cependant, le corpus reflète le comportement d'un utilisateur qui prend habituellement des photos dans les mêmes régions. Un comportement qui pourrait être récurrent. L'utilisation d'un nom d'une

[113] http://www.flickr.com/services/api/

ville et de l'information temporelle (par exemple, Paris, hiver, 2007) fournit, comme attendu, des meilleurs résultats que la seule utilisation d'un nom de ville ou un nom d'objet géoréférencé.

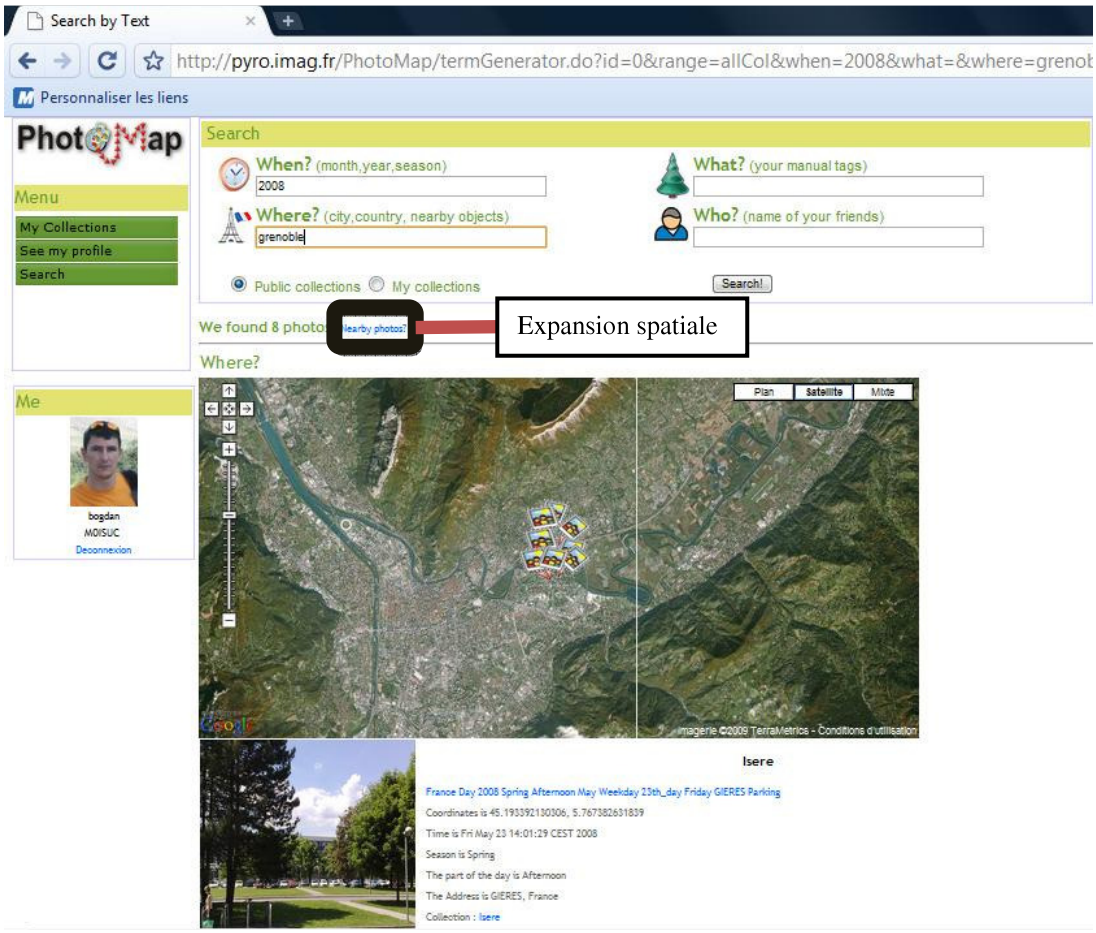


Figure 118 - Interface de requête à base de propriété intégrée à PhotoMap.

L'utilisation de noms d'objets géoréférencés (dans ces tests, les rubriques géoréférencés de Wikipedia) avec des métadonnées temporelles a donné les meilleurs résultats en termes de MRR. Ceci montre l'importance des objets à proximité pour la recherche d'un multimédia personnel si l'utilisateur se souvient du lieu où le document a été créé, y compris lorsque l'objet n'est pas présent sur le document.

La deuxième expérimentation réalisée compare l'utilisation de mots-clés manuellement ajoutés par les utilisateurs, avec les mots-clés générés automatiquement par ContextAnnotator, dans la découverte de photos liées à un monument touristique. Nous faisons des requêtes avec les mots-clés : « Louvre », « Tour Eiffel », « Notre Dame de Paris » et « Stade de France ». Dans premier temps, nous avons écrit des requêtes avec ces mots-clés en tant qu'étiquettes textuelles (par exemple, « Louvre.what »). Dans ce cas-ci, le système répond avec les documents multimédias annotés manuellement par ces mots-clés. Ensuite, nous avons reformulé ces requêtes comme étiquettes spatiales (par exemple, « Louvre.where »). Le système renvoie, ainsi, des documents multimédias créés à proximité du lieu géoréférencé qui possède le mot-clé comme nom. Le classement du résultat de cette requête est directement influencé par la distance entre le lieu de création des documents multimédias et la localisation de l'objet géoréférencé (plus, le lieu de création d'un document est proche de l'objet géoréférencé, plus le document sera haut dans le classement du résultat d'une requête qui constituée du mot-clé identifiant l'objet géoréférencé).

Type de terme de la requête	MR	MRR
Ville	142,85	0,12
Ville, Moment de la journée, Saison, Mois, Année	11,85	0,21
Moment de la journée, Saison, Mois, Année	34,14	0,054
Objet géoréférencé	43,6	0,162
Objet géoréférencé, Moment de la journée, Saison, Mois, Année	6,2	0,475

Tableau 17 - Résultats de tests de recherche à l'aide de timbres sémantiques.

Nous avons mesuré la précision moyenne[114] des résultats d'un requête en considérant des groupes de résultats de taille différente (par exemple, le *Top 3* –les trois premiers documents du classement, le *Top* 50- les cinquante premières images du classement, …). Nous considérons une réponse comme correcte, uniquement si l'image du résultat contient l'objet. Les résultats de notre expérimentation sont affichés sur la Figure 119.

La ligne indiquée par la légende *Manual* représente la précision moyenne des requêtes dont les mots-clés sont mis en correspondance avec les annotations textuelles, et la ligne avec la légende *Wiki* illustre la précision moyenne lorsque les mots-clés de la requête sont comparés aux annotations spatiales. Nous pouvons conclure que lorsqu'une photo est prise près de l'objet géoréférencé (le *Top 3* et le *Top 5*), la précision moyenne n'est pas aussi bonne que sur l'autre type de requête. Ce résultat est dû au fait qu'une photo prise très proche d'un monument peut être, en réalité, une photo d'un autre objet vu de ce lieu de création. La Figure 120 illustre un exemple de ce problème. Elle montre le résultat *Top 3* de la requête avec le mot-clé « Tour Eiffel » et un des photos du résultat montre l'Arc de Triomphe vue de la Tour Eiffel, … Afin d'éviter ce problème, les relations spatiales qualitatives, proposées dans le chapitre 6, peuvent être utilisées. Par exemple, les photos prises à l'extérieur ont plus chances d'afficher le monument. Cependant, le calcul de ces relations spatiales qualitatives exige la description du contour des objets géoréférencées, information qui n'est pas encore disponible sur Wikipedia.

Concernant la ligne de requêtes de mots-clés textuelles (identifié par « manual » dans la figure), nous observons la nette diminution de la précision moyenne de ces requêtes lorsque le nombre de photos considérées dans le résultat augmente. Ce comportement est dû à plusieurs erreurs sur les étiquettes manuelles ajoutées par les utilisateurs de Flickr.

Nous pouvons conclure, à partir de cette expérimentation, que, sur les grandes collections d'images personnelles géoréférencées, les photos des monuments touristiques peuvent être trouvées grâce à l'intégration des rubriques géoréférencées de Wikipedia. La précision du résultat est quasiment aussi bonne que l'usage de mots-clés personnels.

[114] La précision moyenne est la moyenne d'un ratio calculé pour chaque requête. Ce ratio est la division du nombre de résultats corrects par les nombres de documents du résultat. Par exemple, un Top 3 d'une requête (les trois premiers résultats) contenant deux documents corrects aura une précision de 2/3.

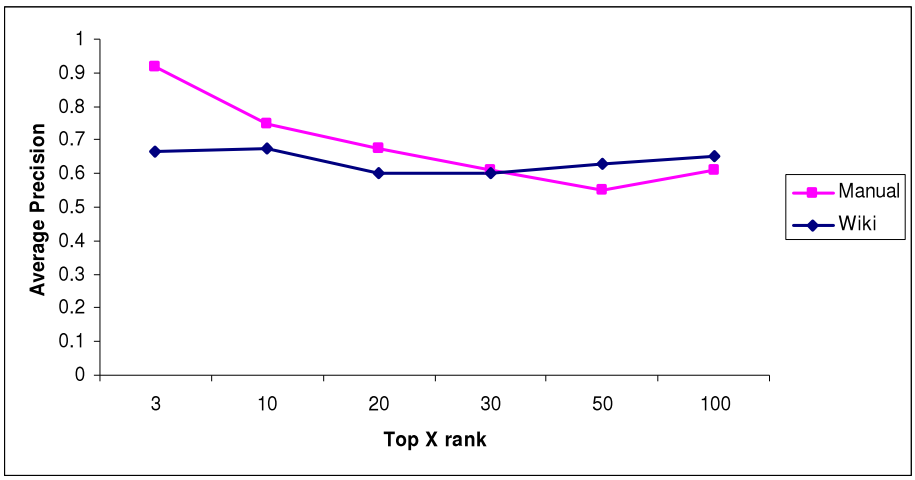


Figure 119- Comparaison entre des requêtes avec des mots-clés manuels avec des requêtes formulées avec le nom d'un objet géoréférencé.

Dans la troisième expérimentation, nous avons comparé l'utilisation :

a) de l'indexation de documents multimédias sans timbres sémantiques (par exemple, sans «.where ») ;

b) l'indexation sémantique de documents multimédias sans expansion spatiale (en considérant exclusivement les timbres sémantiques) ;

c) l'indexation sémantique de documents multimédias combinée à l'expansion spatiale.

L'approche (a) est équivalente à celle proposée par MediaAssist. Nous avons fait une recherche dans le corpus avec les mots-clés « Paris » et « football ». Pour les requêtes sans timbres sémantiques (a), nous avons obtenu 72 photos dans différents endroits de la France où l'équipe de football Paris Saint-Germain (PSG) a joué, mélangées aux photos des stades de Paris (par exemple, le Parc de Prince) et des photos du Stade de France. Le Stade de France est situé dans la banlieue de Paris, néanmoins, comme nous l'avons supposé, les utilisateurs de Flickr ont ajouté Paris comme annotation manuelle.

Pour notre proposition sans expansion spatiale (b), nous avons formulé la requête « Paris.where » et « football.what ». Le système a répondu avec peu d'images (26). Le résultat contient exclusivement les photos vraiment prises dans des limites de ville de Paris, qui ont été annotées manuellement avec le mot-clé « football ».

Lorsque nous avons employé l'expansion spatiale, nous avons obtenu 98 photos qui ont été prises ou à Paris, ou près du Stade de France. La Figure 91 du Chapitre 7 illustre cette comparaison *sans* et *avec* l'expansion spatiale.

Figure 120 - Les trois images mieux classées de la requête « Tour Eiffel .where ».

Il est difficile de comparer ces approches de trois requêtes mesurant uniquement le rappel et la précision (Quelles sont les photos correctes pour ces types de requêtes ?). Cependant, il est clair que l'indexation que nous proposons apporte un modèle de requête sémantique qui est plus précis, car permettant à un utilisateur de mieux exprimer ses besoins. De plus, le processus d'expansion spatiale peut être employé pour augmenter le rappel lorsque le système renvoie peu de résultats.

Par conséquent, dans notre interface de requêtes nous n'activons pas au départ l'expansion spatiale. Elle est une option d'expansion des résultats (voir Figure 118). Une étude d'utilisabilité de cette interface de requête doit être réalisée pour mieux clarifier l'apport de notre approche d'indexation sémantique et d'expansion spatiale.

9.2.2 Évaluation du déploiement adaptatif

Hormis leur utilisation pour l'application mobile de PhotoMap, nos solutions de déploiement adaptatif, de capture de contexte, et l'infrastructure à composants, ont été exploitées pour le développement et déploiement d'un ensemble de services mobiles dédiés à l'administration de l'enseignement lors du projet de collaboration franco-brésilien (FAPESP-CNRS). Un de ces services, appelé CA Messenger (C*ontext-Aware Messenger),* permet d'envoyer des messages sensibles au contexte liés à des événements prédéfinis. Par exemple, le coordinateur pédagogique du cours peut envoyer un message pour diffuser une information sur le début d'un séminaire qui sera transmis aux étudiants d'une formation ou à ceux situés près du lieu du séminaire. Ce service utilise l'architecture pour les opérations sensibles au contexte suivantes : adaptation du module de communication (lors du déploiement et de l'exécution), adaptation de contenu (filtrage des messages), adaptation de la présentation (les messages peuvent être formatés et envoyés via l'application, ou sous la forme d'un *e-mail* envoyé à l'utilisateur, s'il n'est pas connecté). Le service est composé de deux modules : le *CA Messenger Server* (serveur) et *CA Message Viewer* (client mobile).

Le *CA Messenger Server* est composé par les sous-modules suivants : l'interface de gestion des messages (fonction d'enregistrement, de modification, de visualisation et d'exclusion), le module d'adaptation des messages et des Web Services pour l'accès à la base de données du système SW3A (un système Web d'évaluation et de suivi académique de l'université UNESP). Lors du projet, un service Web permettant de localiser l'utilisateur en utilisant le Bluetooth/Wifi a été créé. Les divers emplacements dans l'Université pilote (salles de cours, salles de séminaires, bibliothèques, réunion, bâtiment, etc.) ont été géométriquement représentés dans la base de données géoréférencées de SW3A. Un composant de localisation reposant sur cette infrastructure Bluetooth a été développé et intégré à l'intergiciel DevAC pour localiser l'utilisateur (BTLocationSensor). L'application CA Messenger exploite cette information dans le filtrage des messages à envoyer.

La flexibilité de l'application à travers les services adaptatifs est visible au niveau des interfaces, des méthodes de communication, et de capture de localisation. Les dépendances d'exécution décrites dans la méta-application sont liées à la version de la plate-forme, aux bibliothèques optionnelles de J2ME, et à la quantité de mémoire d'exécution disponible. La Figure 122 illustre deux versions d'interface, chacune choisie en fonction des caractéristiques du dispositif mobile. Les détails sur cette application et ses fonctionnalités ont été rapportés dans [Viana *et al.*, 2009c] [Viana *et al.*, 2008f][Bergeret *et al.*, 2009].

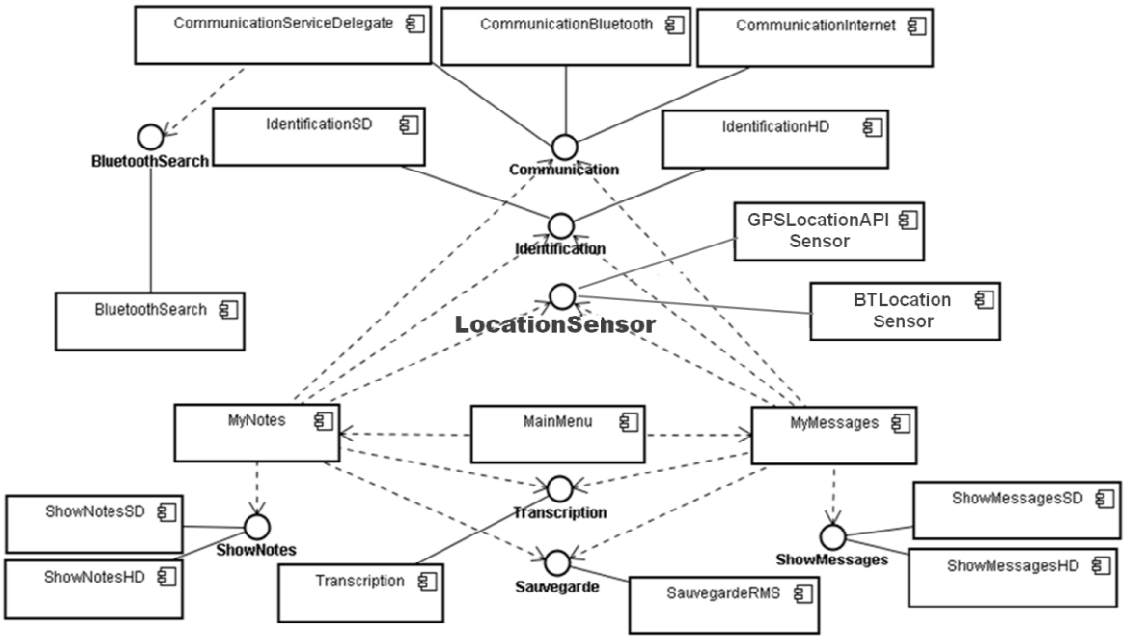


Figure 121 - Méta-application du CA Messenger, en évidence les composants de localisation du DevAC.

Nous avons développé, en utilisant J2EE et WURFL, un portail d'applications contenant nos approches de déploiement adaptatif et de découverte automatique des caractéristiques de dispositifs mobiles. La génération d'une application peut engendrer des temps d'exécution plus ou moins longs sur ce serveur d'installation en fonction du nombre de composants ou de services décrits sur la méta-application. Nous avons souhaité mesurer ce temps lorsque l'utilisateur décide d'installer une application sur son dispositif mobile. Cette mesure concerne uniquement le temps nécessaire pour générer et compiler l'application à partir des composants décrits dans la version générée. Elle ne comprend pas son temps de téléchargement et d'installation.

Figure 122 - Les deux versions d'interfaces qui sont sélectionnés en fonction des caractéristique du dispositif.

Nous avons utilisé la méta-application CAMessenger pour ces expérimentations. Des parties du fichier XML décrivant la méta-application ont été recopiées ou éliminées pour simuler des méta-applications de diverses tailles. Cette expérimentation a été réalisée sur un le serveur Marvelig[115]. Le Tableau 18 montre que le temps d'exécution est quasiment transparent jusqu'à 20 composants (380 millisecondes) et reste raisonnable jusqu'à 100 (légèrement supérieur à une seconde). En revanche, si la demande d'installation est de 200 composants, le temps de génération se rapproche des quatre secondes et sera donc perceptible.

Nombre de services	1	1	1	2	7	14	21	42	56	70	140
Nombre de composants	1	2	3	5	10	20	30	60	80	100	200
1ere mesure	40	46	52	182	270	380	510	853	1119	1370	3994
2eme mesure	40	44	53	178	255	376	497	893	1196	1359	3891
3eme mesure	37	43	51	179	281	403	518	855	1086	1348	2757
4eme mesure	40	41	49	176	273	368	489	825	1091	1398	4037
5eme mesure	38	42	56	182	253	370	490	832	1090	1359	4067
Moyenne (ms)	**39**	**43**	**52**	**179**	**266**	**379**	**501**	**852**	**1116**	**1367**	**3749**

Tableau 18 - Temps d'exécution pour générer l'application CA Messenger par le serveur web d'installation.

La Figure 123 montre l'évolution du temps d'exécution en fonction du nombre de composants. Ce test montre que le temps d'installation d'applications de taille raisonnable est négligeable comparé au temps de téléchargement. L'augmentation du nombre de dépendances à valider peut également affecter ce temps d'installation. Néanmoins notre solution de déploiement reste exploitable dans un environnement réel.

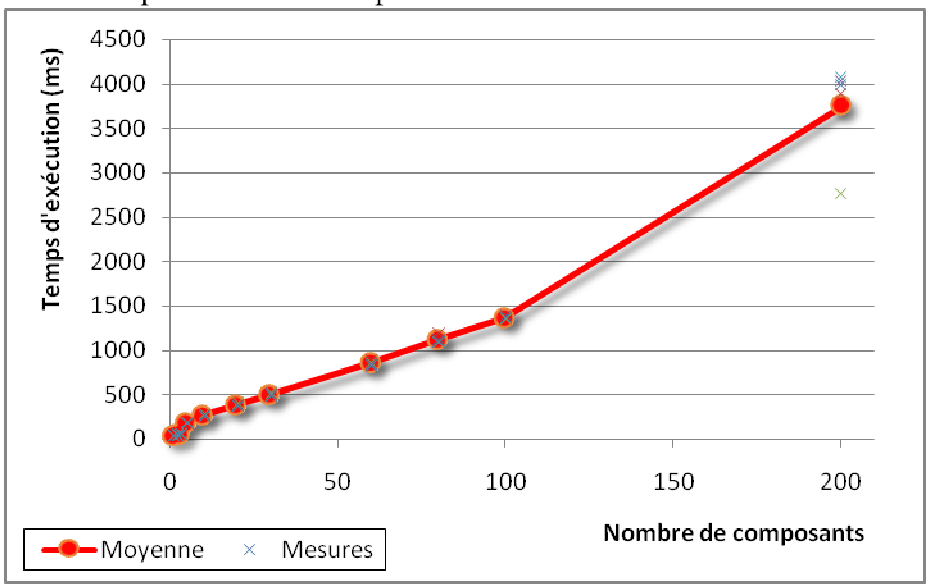


Figure 123 - Graphe d'évolution du temps d'exécution pour générer une application mobile par le gestionnaire de déploiement.

[115] https://marvelig.liglab.fr/doku.php

10 CONCLUSION

Ce travail de thèse a abordé principalement deux thématiques : la sensibilité au contexte appliquée à la gestion de documents multimédias personnels, et le déploiement d'applications mobiles et sensibles au contexte.

- La gestion sensible au contexte vise à augmenter la quantité et le niveau sémantique des métadonnées associées à des documents multimédias personnels.
- Le mécanisme de déploiement adaptatif vise à rendre plus aisé le développement et le déploiement d'applications mobiles, multimédias et multi-capteurs.

Nous avons présenté des solutions qui permettent la construction d'outils de gestion de documents multimédias plus évoluées, et des propositions qui aident les concepteurs dans le développement d'applications mobiles susceptibles d'être activées sur une grande diversité de dispositifs.

Notre travail de thèse part du constat que les dispositifs mobiles multifonctionnels, multi-capteurs et connectables sur le Web deviennent les principaux équipements de création et de partage de documents multimédias personnels. Ce changement d'usage des dispositifs mobiles, déjà en marche, ouvre de nouvelles perspectives pour la gestion de documents multimédias.

Les mécanismes de gestion de documents multimédias reposent fondamentalement sur l'association de métadonnées à ces documents. Ces mécanismes soufrent d'un coté du fossé sémantique des méthodes d'extraction de métadonnées à partir du contenu, et de l'autre, des inconvénients de l'association manuelle de métadonnées par les utilisateurs eux-mêmes. La génération automatique des métadonnées contextuelles capturées à l'aide de dispositifs mobiles peut aider à transformer l'activité répétitive de l'annotation en une activité de validation de métadonnées.

L'étude des modèles et des solutions dédiés à l'acquisition automatique, la représentation, et l'exploitation des métadonnées associées aux documents multimédias, nous a permis de dresser un panorama présentant les atouts et les limites des approches de gestion de documents multimédias. Nous avons :

- Les ontologies semblent mieux répondre aux problèmes d'expressivité et d'interopérabilité des approches d'annotations par mots-clés.
- La caractérisation du contexte de création des documents peut automatiser l'acquisition de métadonnées. Cependant, il n'existe pas de vocabulaire dans le domaine de la gestion de documents multimédias proposant une représentation sémantique et extensive des métadonnées contextuelles.
- Les approches de gestion sensible au contexte de documents multimédias sont peu formalisées et reposent sur des informations contextuelles quasiment symboliques. Dans ces approches, l'exploitation des métadonnées contextuelles est segmentée (soit elles sont utilisées pour l'annotation, soit pour l'organisation, etc.), et les relations sémantiques entre les métadonnées et le document multimédia sont perdues lors de la phase d'indexation des documents.

Du point vu conceptuel, les solutions de gestion sensible au contexte de documents multimédias entremêlent les mécanismes d'acquisition et d'enrichissement de contexte directement dans le code des applications en rendant plus difficile leur extension et leur flexibilité.

Nous avons également étudié des approches du domaine d'adaptation au contexte des Systèmes d'Information. Notre objectif était d'identifier les principes de modélisation du contexte et de structuration de la gestion d'informations contextuelles (acquisition, enrichissement, exploitation) en vue d'une utilisation pour le développement d'une architecture destinée à la mise en œuvre de services sensibles au contexte de gestion de multimédias. Cependant, nous avons identifié certaines limitations qui empêchent l'application directe de ces propositions pour la gestion de documents multimédias :

- La notion de contexte de ces approches est trop centrée sur l'adaptation. Par conséquent, les modèles de contexte correspondants sont liés à la représentation des éléments pertinents pour l'adaptation d'un système.
- L'approche dite du « tout-ou-rien » des architectures sensibles au contexte qui ne prennent pas en compte l'hétérogénéité des dispositifs mobiles et de leurs plates-formes de programmation.

Bilan du travail réalisé

Nos contributions peuvent être divisées en deux parties : i) propositions dédiées à une gestion sensible au contexte de documents multimédias, et, ii) propositions destinées à la mise en œuvre d'un mécanisme de déploiement adaptatif d'applications mobiles et sensibles au contexte.

Les propositions dédiées à une gestion sensible au contexte de multimédias englobent :

I - Des modèles pour la représentation d'informations contextuelles

Nous avons proposé trois ontologies : un modèle générique nommée *Context Top*, l'ontologie *ContextMultimedia* pour l'annotation d'un document multimédia, et une ontologie de description de dispositifs mobiles. Ces modèles permettent la représentation des informations contextuelles, à la fois pour la description du contexte de création d'un document multimédia, et pour l'exploitation lors d'un processus d'adaptation d'applications mobiles.

ContextTop offre une segmentation du contexte en cinq dimensions : spatiale, temporelle, spatiotemporelle, sociale et informationnelle. Ce modèle sert de base aux autres ontologies.

ContextMultimedia répond au besoin d'un vocabulaire pour représenter les métadonnées d'un document multimédia personnel, notamment, celles associés au contexte de création. Ce modèle fournit des descripteurs textuels, contextuels, et de contenu qui permettent une description formelle et extensive d'un document multimédia, et englobent la majorité des métadonnées utilisées par les approches de gestion de ces documents.

II – La plate-forme d'enrichissement des métadonnées *ContextAnnotator*

Nous proposons une plate-forme pour augmenter automatiquement les métadonnées associées à un document multimédia en exploitant les connaissances renseignées dans le modèle *ContextMultimedia*. *ContextAnnotator* offre, aux services mobiles de partage de contenu et aux outils de gestion de multimédias, une liste de services d'enrichissement de métadonnées qui reposent sur trois mécanismes : i) l'accès aux sources de données du Web, ii) l'accès aux données stockées et gérées par la plate-forme elle-même, et iii) l'exécution de processus d'inférence et d'analyse du contenu du document multimédia.

La modélisation de ce processus d'enrichissement est inspirée des architectures de gestion de contexte pour l'adaptation de Systèmes d'Information. Ce processus est flexible, car le concepteur peut indiquer les métadonnées ou les dimensions de métadonnées qu'il souhaite dériver. De plus, le processus est modélisé de telle sorte que son extension soit facilité par l'ajout de nouveaux services Web.

III – Des mécanismes d'exploitation des métadonnées contextuelles pour le partage sensible au contexte et pour l'indexation sémantique de documents multimédias

Une extension du modèle vectoriel est proposée afin de garder, lors de la phase d'indexation et d'interrogation, les relations sémantiques entres les termes d'indexation et les documents multimédias. Cette extension assure la prise en compte de ces relations sémantiques lors de l'interrogation du corpus de documents et peut réduire les problèmes d'ambigüité des approches reposant uniquement sur une comparaison syntaxique de termes. Cinq dimensions – spatiale, temporelle, spatio-temporelle, textuelle et sociale – sont employées pour classer les métadonnées et pour établir les index par mots-clés des documents. Ce processus maintient une partie de la sémantique de la relation entre les documents et chaque terme, à travers un timbre sémantique. Un processus d'expansion sémantique est également ajouté à l'indexation afin d'éviter les problèmes des requêtes imprécises. Les expérimentations entreprises montrent les avantages de l'utilisation de ces métadonnées contextuelles dans le processus d'indexation des photos personnelles.

Les connaissances exprimées à l'aide de l'ontologie ContextMultimedia sont également exploitées sur la plate-forme CoMMediA, à la fois pour enrichir les messages partagés et pour définir un mécanisme de notification de multimédias. Notre modèle de notification, par exemple, permet aux utilisateurs d'exprimer des règles de partage automatique qui font référence au contexte de création des documents. Le système de gestion de multimédia utilise ces règles pour définir quelles personnes seront notifiées. Notre proposition de notification profite de la richesse des métadonnées représentées par ContextMultimedia pour offrir ainsi un modèle de notification plus sémantique.

Le Tableau 19 présente une vue synthétique de nos contributions pour la gestion de documents multimédias qui peut être comparée aux propositions résumées par le Tableau 2 (chapitre 3). Nos modèles fournissent des vocabulaires extensibles pour la description de métadonnées, et nous avons proposé des mécanismes qui exploitent ces informations dans toutes les phases de la gestion de documents multimédias.

Tableau 19 - Vue synthétique des propositions de CoMMeDiA pour la gestion sensible au contexte.

Approche	Métadonnées	Stockage de Métadonnées	Acquisition de Métadonnées	Exploitation
CoMMeDiA	Contexte social, spatial, temporel, spatio-temporel et informationnel	L'ontologie ContextMultimedia	• Capture : Application mobile • Enrichissement : Web Services, réglés d'inférence et méthodes de détection de visages	• Organisation • Indexation Sémantique • Partage sensible au contexte • Annotation

Le succès des solutions de gestion de documents multimédias est étroitement lié à l'acquisition de contenu et de métadonnées créés sur les dispositifs mobiles des utilisateurs. Cependant, l'hétérogénéité inhérente à ces terminaux exige quasiment la réécriture du code de ce genre d'application pour chaque modèle de dispositif. Un des objectifs de CoMMediA est précisément d'aider les concepteurs à développer et à déployer des applications mobiles adaptées au type de dispositif des utilisateurs, sans que ces concepteurs aient besoin de réécrire plusieurs fois le code de l'application. L'ensemble des mécanismes de description des applications et de déploiement adaptatif de CoMMediA comprend :

IV – L'infrastructure à composants iLet

Afin d'offrir un support à l'adaptation d'applications mobiles sur les plates-formes mobiles Java les plus contraignantes (Doja, J2ME MIDP, etc.), nous avons proposé un modèle de composants, un modèle de connexion, et un conteneur d'exécution inspirés de la plate-forme OSGi. L'infrastructure possède 11KB et a été testé avec succès sur plusieurs modèles de dispositifs mobiles. Par ailleurs, il est important d'observer que, même si originellement cette infrastructure a été développée pour notre proposition d'adaptation, elle peut être utilisée pour le développement d'applications mobiles indépendamment de nos solutions.

V – La description et la découverte automatique des caractéristiques d'un dispositif mobile

Grâce à une extension du registre de profils WURFL, nous pouvons profiter à la fois d'un mécanisme de découverte automatique d'un dispositif mobile et d'une de description détaillée des plates-formes préinstallées et des ressources matérielles intégrées (en incluant les capteurs) aux dispositifs mobiles. Une extension de l'ontologie *Context Top* et une ontologie de description d'une plate-forme mobile assurent la description minutieuse d'un dispositif mobile.

VI – L'approche de déploiement adaptatif

Nous proposons un modèle de description des méta-applications qui décrivent un assemblage de composants et de services adaptatif qui contiennent plusieurs alternatives d'implémentation. La méta-application décrit le comportement fonctionnel idéal de l'application, et ses alternatives proposent une dégradation de ses fonctionnalités afin de permettre son exécution sur une plus grande quantité de dispositifs mobiles.

Nous avons proposé un gestionnaire de déploiement qui génère des applications à partir de la mise en correspondance des méta-applications et de l'identification automatique d'un dispositif mobile. Notre approche de déploiement adaptatif fournit un mécanisme d'adaptation qui n'exige pas l'installation *a priori* d'un intergiciel et peut être utilisé pour adapter des applications mobiles en général.

VII - L'intergiciel DevAC (Device-Aware Context toolkit)

Ce module de capture de contexte est un intergiciel orienté composants s'exécutant au dessus du canevas iLet. DevAC est un canevas hiérarchique de capteurs synchrones et asynchrones pour la caractérisation du contexte de l'utilisateur.

Nous avons introduit des mécanismes d'adaptation pour modifier la capture du contexte en fonction du modèle du dispositif mobile et du contexte d'exécution de l'intergiciel. En effet, DevAC est lui-même adapté lors de l'installation de l'application mobile par le gestionnaire de déploiement de CoMMeDiA.

DevAC est plus flexible que les architectures d'acquisition de contexte reposant sur une approche de tout-au-rien, car ses composants sont déployés en fonction des besoins des applications sensibles au contexte. Nous assurons ainsi une plus grande portabilité des applications mobiles et sensibles au contexte, et une économie de ressources exploitées par l'intergiciel de capture du contexte.

Le Tableau 20 présente une vue synthétique de nos contributions pour le déploiement adaptatif et peut être comparée aux propositions résumées par le Tableau 11 (chapitre 5).

Tableau 20 - Vue synthétique des propositions de CoMMeDiA pour le déploiement adaptatif.

Critères	CoMMeDiA
Identification automatique des DM	OUI
Plate-forme	J2ME MIDP et DOJA
Type d'application	Applications orientées composants
Exigence d'installation d'un intergiciel	Partiel. L'intergiciel est installé avec l'application.
Métadonnées de l'application	Décrites en XML
Restrictions	Décrites en XML
Adaptation après l'installation	OUI du DevAC
Mise à jour	-
Mise en correspondance entre les restrictions et le contexte d'exécution	Programmé en Java et enrichi par des méthodes d'inférences des ontologies OWL
Détection de conflits de règles d'adaptation	Une priorité est établie entre les alternatives de composants
Technologie de Composants	Ilets

Perspectives

Notre travail de thèse ouvre des perspectives scientifiques à court et à plus long termes. Nous soulignons à présent les perspectives qui nous semblent pertinentes pour l'évolution des propositions réalisées.

I- Évolution et évaluation approfondie de PhotoMap

À ce jour, le système créé pour valider nos propositions gère uniquement des photos et une partie des concepts de ContexMultimedia n'est pas pris en compte lors d'indexation (par exemple, les relations sociales, à l'exception de *friend* et *friend of friend*). À cout terme, nous souhaitons intégrer dans PhotoMap d'autres types de multimédias et tous les concepts et relations de ContextMultimedia. Nous allons étendre l'application mobile pour produire des vidéos et des fichiers audio, et aussi de l'annotation vocale d'un multimédia.

Une fois finies ces améliorations, nous souhaitons réaliser des tests d'utilisabilité. En fait, PhotoMap a été uniquement utilisé par des membres de notre équipe de recherche lors de simples essais. Nous envisageons, en revanche, d'étudier le comportement de plusieurs utilisateurs afin de juger de l'exactitude et de l'utilité des annotations générées automatiquement. Nous souhaitons effectuer des expérimentations similaires à celles réaliser par Naaman [Naaman *et al.*, 2004] en utilisant nos outils, et pouvoir ainsi mieux mesurer l'apport de nouvelles métadonnées intégrées. De plus, nous espérons introduire un mécanisme de *relevance feedback* dans notre processus d'interrogation afin d'adapter les résultats d'une requête au comportement de requête d'un utilisateur.

II- Recommandation sensible au contexte de documents multimédias

Le partage massif de documents multimédias sur le Web, nous permet d'envisager, dans un avenir proche, l'existence des grands entrepôts de documents multimédias enrichis de métadonnées contextuelles. Ces documents peuvent être exploités dans un processus de suggestion sensible au contexte de documents multimédias. Le principe est de recommander à un utilisateur des multimédias créés dans un contexte similaire à son contexte actuel. Par exemple, un touriste souhaite voir des photos prises à la même époque de l'année et dans la même région que celle où il se trouve actuellement.

Notre modélisation mathématique du processus d'indexation peut être étendue pour offrir la base de calcul de similarité entre le contexte actuel de l'utilisateur et le contexte annoté pour chaque multimédia répertorié. Le résultat peut donner plus d'importance aux documents créés par des personnes du réseau social élargi de l'utilisateur.

III- Extension du modèle de description d'applications mobiles dans une perspective de réutilisation

Notre modèle de description des méta-applications permet la description des variabilités d'une application en fonction des caractéristiques matérielles et logicielles d'un dispositif mobile. Cette approche destinée au déploiement adaptatif partage certains principes avec l'ingénierie de *lignes de produits logiciels* (*Software Product Lines*-SPL). Une SPL propose le développement systématique des logiciels partageant des points communs (*core assets*), mais présentant aussi des différences (appelées variabilités), qui sont développées à partir de composants communs dans un domaine déterminé. Le principe est de minimiser les coûts de construction de logiciels dans un domaine d'application particulier en ne développant plus chaque logiciel séparément, mais plutôt en le concevant à partir d'éléments réutilisables. Contrairement à notre approche, la variabilité dans une SPL comporte des aspects fonctionnels et non-fonctionnels qui vont varier selon l'application produite à partir de la *famille de produits*. L'objectif d'une SPL n'est pas d'adapter une application précise, mais plutôt de réutiliser le maximum des composants et des ressources entre plusieurs applications d'une même famille.

Nous souhaitons étendre notre proposition de description de méta-applications dans une perspective de développement d'une ligne de produit d'applications mobiles. Notre modèle de description d'un composant et des services peuvent être exploités pour la construction d'un registre de composants qui décrit sémantiquement les propriétés fonctionnelles (par exemple, les services décrit en OWL-S) et les propriétés non-fonctionnelles (notamment, les dépendances d'exécution liés à notre extension de WURFL). Ce registre doit être intégré à la SPL MobiLine, développé en collaboration par les universités brésiliennes UFC et UFRJ dont l'objectif principal est de permettre une réutilisation systématique dans toutes les phases de développement d'un logiciel mobile et sensible au contexte.

ANNEXES

ANNEXE 1 : LES PLATES-FORMES LOGICIELLES MOBILES ET LEURS KITS DE DEVELOPPEMENT

Nous avons choisi d'étudier plus en détail les cinq principales plates-formes mobiles : J2ME MIDP/CLDC, Symbian OS, iPhone OS, Windows Mobile et Android. Nous examinons ces plates-formes mobiles en mettant en évidence trois aspects : i) la portabilité des applications développées; ii) l'environnement d'exécution des applications mobiles, iii) l'accès aux fonctions multimédias et aux données des capteurs intégrés dans les terminaux. À la fin de cette annexe, nous offrons une vue synthétique du point de vue du développement dans ces cinq plates-formes étudiées[116] (Tableau 21, Tableau 22, et Tableau 23), et également une vue synthétique des diverses étapes du déploiement sur ces plates-formes (Tableau 24, Tableau 25, et Tableau 26).

10.1.1.1 Symbian OS

Symbian OS[117] est un système d'exploitation développé par Symbian Ltd. appartenant à Nokia depuis 2008. Ce système est une évolution du système Epoc32 qui équipait les dispositifs mobiles de la marque Psion (voir Section 2.1). La première version de Symbian a été lancée en 1999 et est désormais disponible sur plus de 250 modèles de terminaux mobiles de divers fabricants (tels que Nokia, Sony Ericsson, Motorola, Samsung).

Un des premiers systèmes d'exploitation destinés aux téléphones mobiles à accepter l'installation d'applications tierces, Symbian est aujourd'hui multitâche et de code source partiellement ouvert. La moulinette de développement Carbide C++118 permet de développer des applications natives en langage C++ non standard. Cet environnement de programmation

[116] Nous avons choisi une notation par étoile sur chaque caractéristique retenue des plates-formes sur les tableaux de comparaison.

[117] http://www.symbian.org/

[118] http://www.forum.nokia.com/Resources_and_Information/Tools/IDEs/Carbide.c++/

repose sur l'outil Eclipse IDE119 et est disponible pour Windows XP et Linux. Carbide C++ peut également être intégré à Microsoft Visual Studio par le biais d'un *plugin*. En utilisant le langage natif, une application possède l'accès quasiment illimité à toutes fonctionnalités disponibles sur les terminaux mobiles Symbian (réseaux, multimédia, fonctions téléphoniques, etc.). Cependant, chaque constructeur de dispositif peut développer son propre SDK en ajoutant ou en enlevant des fonctions et en personnalisant l'interface utilisateur. Ces différences entre les SDK de Symbian OS entrainement la diminution de la portabilité d'une application dès qu'elle utilise une fonction absente sur un des SDK. Le SDK Series 60 (S60) fournit par Nokia et le SDK UIQ fournit par Sony Ericsson sont les kits de développement les plus répandus. La Figure 124 illustre des captures d'écran du système Symbian qui s'exécute sur le dispositif Nokia N95 et dans l'émulateur Windows appartenant au SDK S60 FP2.

Figure 124 - Le système d'exploitation Symbian OS s'exécutant sur le dispositif N95 et sur l'émulateur du SDK Series 60.

Concernant les API multimédias, le SDK Series 60 offre des documents pour reproduire des sons, des vidéos et pour visionner des images. L'enregistrement de multimédia dépend du modèle du dispositif. Malgré cet inconvénient, les bibliothèques multimédias existantes dans le SDK offrent le contrôle quasi-total du matériel (tel que le réglage du flash par une application de prise de photo). D'autres bibliothèques du SDK proposent un accès aux données provenant du GPS, de l'accéléromètre et des boussoles dans le cas de la présence de ces capteurs sur le dispositif. Les API natives de Symbian OS permettent d'acquérir le niveau des signaux perçus par la carte de réseau Wi-Fi. Cette information peut être utilisée pour la construction d'un système de localisation indoor si l'information de localisation des points d'accès détectés est disponible [Madigan *et al.*, 2005].

L'environnement d'exécution des applications est limité par les caractéristiques matérielles du dispositif. L'approche multitâche permet la communication entre deux applications qui s'exécutent sur le même dispositif en s'appuyant sur une connexion par *sockets*. Cette communication a même permis le développement des services Web qui sont mis en œuvre localement sur le dispositif mobile. Au delà de la programmation C/C++, les plates-formes J2ME, Python et Ruby peuvent être utilisées pour la création d'applications Symbian.

[119] http://www.eclipse.org/

10.1.1.2 iPhone OS

Dérivé du système d'exploitation Mac OS X, l'iPhone OS[120] équipe les dispositifs mobiles du constructeur Apple : l'iPod Touch et les trois versions de l'iPhone (iPhone, iPhone 3G, iPhone 3G-S). Le premier dispositif est un baladeur numérique avec accès Wi-Fi et l'iPhone est le *smartphone* le plus vendu au monde. Ces dispositifs possèdent tous un large écran tactile, ce qui explique, en partie, le grand succès de vente de ces terminaux. Initialement, l'iPhone OS était un système d'exploitation fermé aux applications tierces. Cependant, depuis le début de 2008, un SDK est désormais disponible et des applications peuvent être installées via le portail App Store. Ces applications sont développées en langage Objective-C à l'aide de l'outil Xcode qui fonctionne exclusivement sur Mac OS. La Figure 125 présente des captures d'écran de l'iPhone OS s'exécutant sur l'iPhone 3G.

Apple impose une série de restrictions pour le développement d'applications tierces (de la structuration du code jusqu'au contenu présenté aux utilisateurs). Ces limitations sont minutieusement inspectées par le constructeur avant la mise à disposition des applications. Néanmoins, ces applications bénéficient d'une large gamme de bibliothèques pour l'accès aux capteurs et aux fonctions multimédias des dispositifs. Par exemple, la bibliothèque Core Location Framework offre le positionnement géographique du dispositif même en absence d'un GPS intégré et à l'intérieur de bâtiments. Ceci est possible grâce au service de positionnement hybride fourni par l'entreprise Skyhook[121]. Cette entreprise possède une énorme base de données avec les adresses MAC (*Media Access Control*) des points d'accès Wi-Fi sur les zones urbaines denses dans le monde (Figure 125). Tous les dispositifs reposant sur l'iPhone OS possèdent une carte réseau Wi-Fi. Ainsi, le service de positionnement capture les adresses MAC détectées par le dispositif et l'intensité du signal de chaque point d'accès. Ces informations sont envoyées au serveur de Skyhook qui calcule une localisation approximative (erreur de 10 à 30 mètres) en utilisant une méthode de recherche par similarité sur leur base de données [Hui *et al.*, 2007]. L'entreprise garantit une mise à jour fréquente de cette base de données afin d'assurer la qualité des informations de localisation. Cependant, en zones moins dense (en termes de présence de point d'accès Wi-Fi), ce service ne fonctionne pas correctement.

Les applications sur l'iPhone OS fonctionnent selon le modèle d'exécution *sandbox* sur lequel une application n'accède pas aux données et à l'espace de mémoire appartenant à d'autres applications. Malgré la possibilité d'exécuter simultanément plusieurs applications produites par Apple, l'iPhone OS permet uniquement l'exécution d'une application tierce à chaque fois. Ceci empêche, par exemple, à une application tierce de tourner en tâche de fond.

[120] http://developer.apple.com/iphone/program/

[121] http://www.skyhookwireless.com/howitworks/

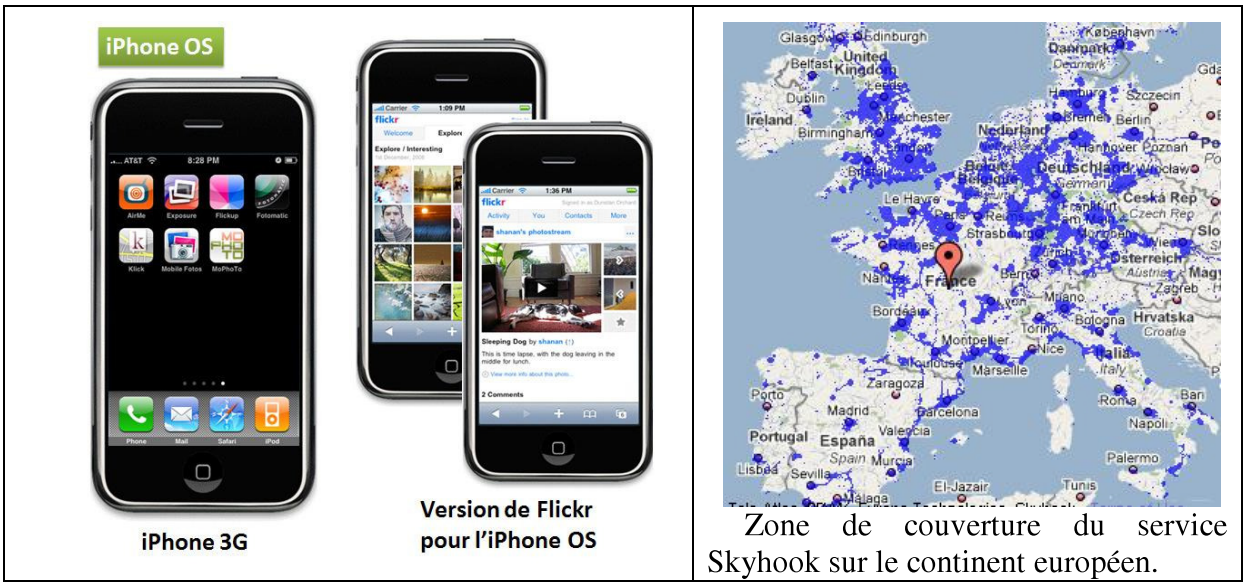


Figure 125 – *À gauche*) capture d'écran de l'iPhone et des émulateurs exécutant l'application de visionnement de photos en ligne : Flickr. *À droite*) Un exemple de zone de couverture du service de localisation par Wi-Fi fourni par Skyhook.

10.1.1.3 Windows Mobile

Conçu initialement en 1996 pour fonctionner sur des assistants personnels, Windows Mobile[122], autrefois nommé Windows Pocket PC, est la solution Microsoft de système d'exploitation mobile. Aujourd'hui, sur sa version 7.0, ce système d'exploitation équipe plus d'une dizaine de modèles de smartphone et il est la plate-forme mobile la plus adoptée derrière Symbian Os. Son avantage majeur est la forte intégration avec les logiciels Windows Desktop (Office, Outlook, Messenger, etc.), et la possibilité de recevoir des courriels en temps réel. La Figure 126 présente ce système d'exploitation s'exécutant sur le HTC Diamond et sur l'émulateur du SDK pour Windows XP.

Les applications natives de Windows Mobile sont développées en Visual C++ à l'aide de l'environnement de développement Microsoft Visual Studio. Ces applications peuvent tirer profit de l'API Win32 afin d'accéder à toutes les fonctionnalités du dispositif. Par exemple, Windows Mobile possède l'API Windows Media Player qui assure la reproduction des documents multimédias (musiques et films en divers formats). La prise de photos et l'enregistrement de vidéos et de fichiers audio est également disponible selon les capacités matérielles du terminal mobile. Certains dispositifs Windows Mobile disposent d'un GPS (tels que le HTC Touch Diamond de la Figure 126). La bibliothèque GPS Intermediate Driver rassemble des fonctions de contrôle de ce capteur de localisation. La dernière version du Windows Mobile inclut également une bibliothèque pour la manipulation des données provenant d'un accéléromètre. Néanmoins, il n'existe pas encore de dispositifs dotés de Windows Mobile qui possèdent ce capteur.

[122] http://www.microsoft.com/france/windowsmobile/default.mspx

Figure 126 - Windows Mobile s'exécutant sur le HTC Touch Diamond et sur son émulateur pour Windows XP.

Windows Mobile est un système d'exploitation multitâche. Sur certains modèles de *smartphones*, deux applications peuvent même communiquer par le biais du protocole COM (Component Object Model). Par ailleurs, le système d'exploitation mobile de Microsoft présente d'autres similitudes avec ces homologues *desktops* en possédant également un registre des clés des logiciels, ainsi que la technologie DLL pour le chargement des bibliothèques. Au-delà de la programmation en Visual C++, Windows Mobile accepte des applications mobiles développées avec les plates-formes logicielles J2ME et Compact .Net Framework. Parmi les systèmes d'exploitations mobile, Windows Mobile est celui qui possède le plus grand nombre d'outils de programmation de très bonne qualité et une large quantité de documentation fournie par Microsoft.

10.1.1.4 Android

Android[123] est un système d'exploitation mobile libre et de code source ouvert. Ce système, basé sur le noyau Linux (version 2.6), a été conçu par Android, une startup rachetée par Google. Il a été annoncé à la fin de l'année 2007. Néanmoins, le premier dispositif commercial l'intégrant a été mis en vente par l'opérateur T-Mobile seulement fin 2008. Il s'agit du dispositif G1 doté d'un GPS intégré (voir Figure 127). D'autres terminaux mobiles ont été annoncés au long de l'année 2009. Android intègre une base de données relationnelle (le SQLlite) et plusieurs services Google (tels que Google Maps, Google Search, Gmail) qui peuvent être exploités par les applications tierces.

Google offre un SDK pour le développement de ces applications en Java. À l'exception de la syntaxe du langage lui-même, le langage Java interprété par Android est complètement différent des versions Java *desktop* et mobile standardisées par Sun Microsystems. Des bibliothèques pour la création d'interfaces graphiques à la structuration d'une application elle-même, tout est différent dans l'univers Java d'Android. Les applications, par exemple, ne possèdent pas un seul point d'entrée (i.e., un main()). Elles sont structurées selon un modèle de programmation par composants où chaque composant peut être invoqué par le système d'exploitation ou par d'autres applications. Les composants essentiels d'Android sont de quatre types :

- *Activity*. Ce composant représente une fonctionnalité fournie par l'application qui possède une interface graphique. Une application peut contenir un seul composant ou plusieurs composants qui héritent du composant Activity.

[123] http://developer.android.com

- *Services*. Ce composant assure l'exécution de tâches qui n'ont pas besoin d'une interface graphique et peuvent être exécutées en mode asynchrone. Par exemple, pour une application de lecture de musique, un composant Activity peut afficher l'interface graphique de la *playlist* et un composant Service peut entamer la lecture et la mise en mémoire tampon du fichier audio.
- *Broadcast Receivers*. Android possède un mécanisme de notification d'événements (par exemple, la prise d'une photo, le changement de fuseau horaire). Les composants du type *Broadcast Providers* permettent aux applications d'utiliser ces mécanismes. Les applications peuvent elles aussi générer des événements et notifier d'autres applications.
- *Content Providers*. Ce composant offre l'accès aux contenus et aux données disponibles sur le dispositif (la liste de contacts, l'agenda, la base de données, etc.)

Une application sur Android est ainsi un arrangement de ces composants. Chaque application est mise en œuvre sur sa propre instance de la machine virtuelle fournie par l'entreprise Dalvik et peu communiquer par RPC (*Remote Procedure Calls*). Android est probablement le système d'exploitation mobile qui offre aux applications le plus de contrôle sur le matériel des terminaux mobiles. Par exemple, une application peut activer ou désactiver les cartes réseaux Wifi et Bluetooth, accéder au niveau de batterie et actionner l'appareil photo. De plus, Android met l'accent sur les services de géolocalisation en fournissant la bibliothèque Google Sensor Api. Les programmes développés en code natif Linux ne sont pas encore supportés. Un autre désavantage d'Android est la faible quantité de documentation et de développeurs en comparaison aux autres plates-formes mobiles. La Figure 127 présente l'exécution de deux applications sur l'émulateur d'Android : Flickr et Google Maps.

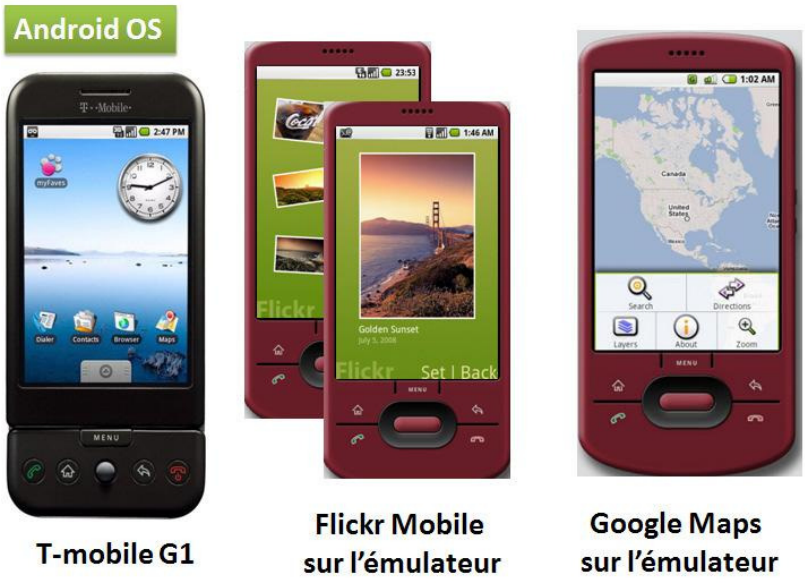


Figure 127 - Dispositif et émulateurs exécutant le système d'exploitation Android.

10.1.1.5 J2ME MIDP

Java 2 Mobile Edition (J2ME)[124] est une collection de technologies et spécifications standardisées et distribuées par Sun Microsystems[125] (entreprise ayant aujourd'hui fusionné

[124] http://java.sun.com/j2me/

avec Oracle Systems126). J2ME n'est pas un système d'exploitation mobile. Il est, en revanche, une plate-forme logicielle d'exécution d'applications qui peut être portée sur divers dispositifs dotés de capacités de calcul (des capteurs, des cartes de mémoire, des téléviseurs, des téléphones mobiles, des PDA, etc.). Les spécifications J2ME sont divisées en configurations et en profils. Le couple profil-configuration indique les fonctionnalités (*i.e.* les bibliothèques) disponibles pour une application et la forme selon laquelle son code doit être structuré. L'ensemble des technologies Java est présenté sur la Figure 128 (partie gauche) dans laquelle nous distinguons deux colonnes de la plate-forme J2ME : le couple MIDP/CLDC et le couple Personal Profile/ CDC.

Une configuration détaille les fonctions de base de la machine virtuelle Java embarquée (chargement de classes, protocoles de communication, types de variables supportées, etc.). La configuration CDC (Connected Device Configuration) a été conçue pour des dispositifs en possession de plus d'un mégaoctet de mémoire et d'une connexion réseau stable et de haute vitesse (ADSL, Ethernet, etc.). Cette configuration s'appuie sur une variante de la machine virtuelle JVM (Java Virtual Machine) de la spécification J2SE (Java desktop). Elle est destinée à s'exécuter sur certains modèles d'assistants personnels et sur des boîtiers de TV et d'accès à l'internet (tel que le boîtier de FreeBox). La configuration CLDC (Connected Limited Device Configuration) a été initialement tournée vers des dispositifs de moins d'un mégaoctet de mémoire et avec des connexions réseaux de basse vitesse (tels que GSM et GPRS). Elle a été conçue pour combler les restrictions des premiers téléphones mobiles. CLDC repose sur une version allégée de la machine virtuelle Java : la KVM (Kilobytes Virtual Machine). CLDC possède deux variantes : i) la version 1.0 qui n'offre pas de support aux nombres à virgule flottante et ; ii) la version 2.0 qui incorpore ce type de variable et d'autres fonctions de calcul.

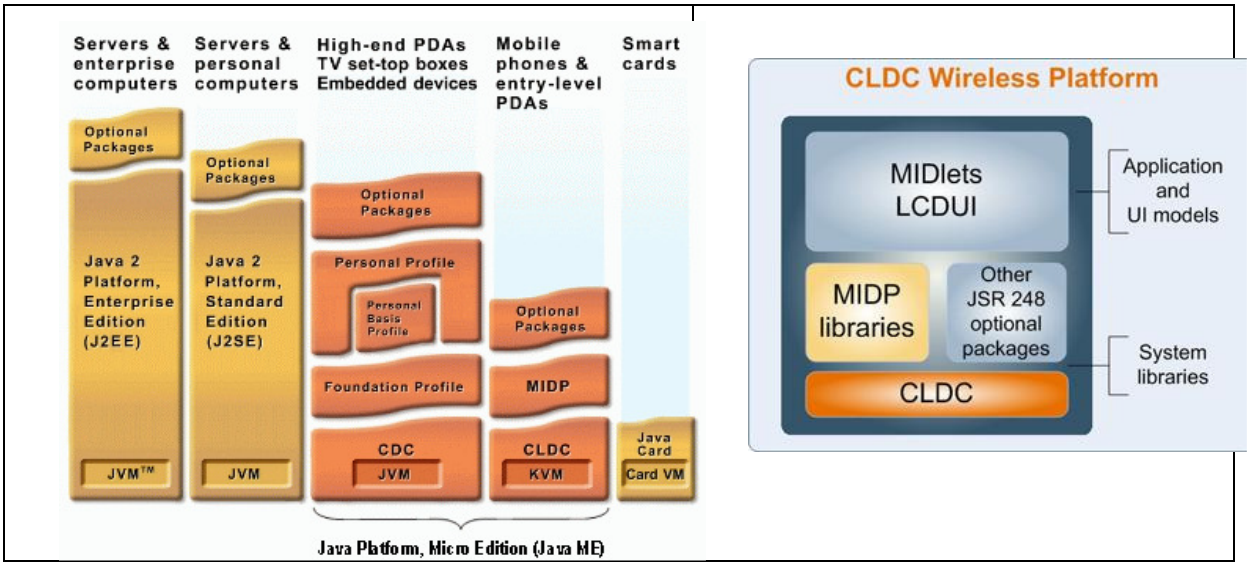


Figure 128 - *À gauche)* L'ensemble des technologies Java. *À droite*) la plate-forme J2ME MIDP/CLDC en détail.

Les profils J2ME déterminent les bibliothèques de haut niveau exploitables par les applications (interface graphique, accès aux fonctionnalités multimédias des dispositifs, etc.). La configuration CDC possède une famille de profils : Personal Profile, Foundation Profile, et Personal Basis Profile. Chaque profil reposant sur CDC contient un ensemble de bibliothèques graphiques différent (AWT, support à des Applets, etc.). Le profil MIDP

[125] http://www.sun.com/

[126] http://www.sun.com/third-party/global/oracle/index.jsp

(Mobile Information Device Profile) est celui utilisé en compagnie de la configuration CLDC. Le MIDP ajoute au CLDC les API d'affichage graphique (2D et 3D), d'accès aux systèmes de fichiers du dispositif, et les fonctions multimédias (accès à l'appareil photo, envoi de SMS et de MMS, etc.).

Contrairement à la configuration CDC et d'autres plates-formes indépendantes d'un système d'exploitation, les machines virtuelles J2ME MIDP/CLDC sont préinstallées en usine sur la majorité des dispositifs mobiles (plus de 8000 modèles de terminaux). J2ME MIDP/CLDC est ainsi la plate-forme mobile la plus universelle [Wesson *et al.*, 2005]. La Figure 128 (partie droite) présente les principaux composants de cette plate-forme que nous nommons simplement J2ME MIDP dans la suite de ce manuscrit.

Au-delà des bibliothèques obligatoires du profil MIDP, chaque constructeur peut décider d'ajouter des bibliothèques optionnelles à la machine virtuelle de ses dispositifs. La Figure 129 présente les API optionnelles à l'intérieur des polygones blancs. Ces bibliothèques offrent, entre autres, l'utilisation des fonctions multimédias (la prise de photo, l'envoi de SMS et MMS, lecture de vidéo, etc.), la gestion de l'interface réseau Bluetooth (JSR -Java Specification Request 82), l'établissement de la position géographique du terminal mobile (la bibliothèque Location API), et l'accès à d'autres capteurs du dispositif (accéléromètre, boussole, thermomètre, etc.). Ce problème de fragmentation des bibliothèques optionnelles n'allège pas la tâche de porter des applications J2ME MIDP sur plusieurs modèles de dispositifs. Une application exploitant une de ces bibliothèques ne fonctionnera pas (elle ne sera même pas activée) sur un dispositif sur lequel la bibliothèque est absente.

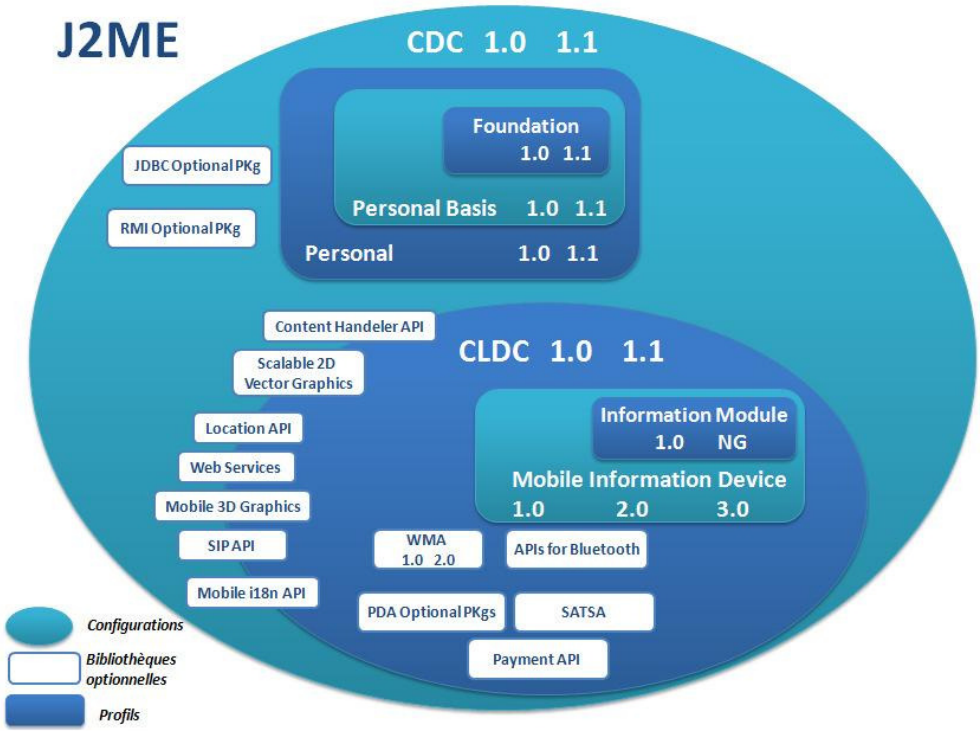


Figure 129 - Les bibliothèques optionnelles de la plate-forme J2ME MIDP.

Le point d'entrée d'une application J2ME MIDP est la classe MIDlet qui possède trois méthodes permettant à la machine virtuelle d'activer le MIDlet, de le suspendre et de le détruire. Une application J2ME MIDP peut être composée de plusieurs MIDlets et de nombreuses ressources (classes, images, etc.). Ces fichiers sont confinés à l'unité de distribution MIDletSuite. Lors de la mise en œuvre de l'application, le MIDletSuite devient l'environnement englobant d'exécution de l'application. Il limite l'accès des MIDlets aux données et à l'espace mémoire d'autres MIDletSuites déjà installés sur le dispositif.

L'application a le droit d'accéder uniquement aux méthodes fournies par les API de la machine virtuelle embarquée et aux ressources retrouvées à l'intérieur du MIDletSuite (données, classes, images, etc.).

Le développement d'applications J2ME MIDP peut se faire sur des ordinateurs équipés de Linux ou Windows. Il est nécessaire au développeur de disposer de Java 2 SDK (celui utilisé pour le développement d'applications Java *desktop*) et de la boîte à outil Wireless Toolkit pour la compilation, le paquetage (de l'anglais *packaging*) et les tests des applications. Les environnements de programmation Eclipse et Netbeans possèdent des *plugins* pour l'intégration du Wireless Toolkit. Les constructeurs de dispositifs fournissent eux-aussi des émulateurs et des outils de paquetages spécifiques pour leurs dispositifs qui sont également intégrables à Eclipse et à NetBeans. La Figure 130 illustre les captures d'écrans de trois applications J2ME MIDP s'exécutant sur des émulateurs de dispositifs mobiles. Ces applications (Google Maps, Jeu de golf en 3D, Opera Mini) offrent un aperçu du potentiel de développement de cette plate-forme malgré son grand inconvénient causé par la fragmentation des bibliothèques.

Figure 130 - Captures d'écrans de trois applications développées à l'aide la plate-forme J2ME MIDP.

Tableau 21 - Comparaison entre les plates-formes mobiles.

	Première Version	Dernière Version du SDK	Langage de programmation		Acquisition d'informations sur le dispositif		Dispositifs Mobiles	
J2ME MIDP	J2ME MIDP 1.0/ CLDC 1.0 en 1998	MIDP 2.1/ CLDC 1.1	• Java avec les API CLDC et MIDP.	★ ★	• Des fonctions de découverte des caractéristiques du dispositif (taille de l'écran, la mémoire). • Des fonctions de détection de la présence des bibliothèques optionnelles (p.ex., Location API).	★ ★ ★	• Plus de 8000 modèles de terminaux mobiles.	★ ★ ★ ★
Android	2007	Android SDK 1.5	• Java avec les API Android.	★ ★	• Plusieurs fonctions de découverte des caractéristiques du dispositif (la taille de l'écran, la mémoire, le niveau de batterie).	★ ★ ★	• Moins d'une dizaine de modèles de dispositif.	★
Windows Mobile	Windows Pocket PC en 1996	Windows Mobile 6.0.	• Le langage natif est le Visual C/C++ . • Possibilité d'utiliser C#, .NET framework et J2ME MIDP.	★ ★ ★	• Bibliothèque Device Configuration API.	★ ★	• Des dizaines de modèles de dispositifs mobiles.	★ ★
iPhone OS	2007 (Ouvert aux applications tierces en 2008)	iPhone OS 3.0	• Objective-C. Des applications J2ME peuvent être installées si le téléphone est débloqué[127].	★ ★	• Quelques fonctions pour découvrir si un appareil photo ou un GPS sont présents.	★	• Seuls quatre modèles : iPod Touch, iPhone, iPhone 3G et iPhone 3G S. Néanmoins, ces modèles sont les plus vendus au monde.	★ ★
Symbian OS	Symbian OS v5 (1999)	Symbian OS v9.4	• Le langage natif est le Symbian C++. • Possibilité d'utiliser J2ME, Flash Lite, Python et Ruby.	★ ★ ★	• Plusieurs fonctions de découverte des caractéristiques du dispositif (telles que la taille de l'écran, la mémoire, le niveau de batterie).	★ ★ ★	• Plus de 250 modèles.	★ ★ ★

[127] http://blog.taragana.com/index.php/archive/how-to-install-compile-run-java-on-iphone/

Tableau 22 - Comparaison entre les plates-formes mobiles.

	Environnement de développement		Fonctionnalités Multimédias		Accès aux capteurs	
J2ME MIDP	• . Développement sur Linux et Windows avec le Sun Wireless Toolkit. • Intégration possible dans les outils Netbeans et Eclipse. • Émulateurs fournis séparément pour chaque constructeur de dispositif.	★ ★	• Reproduction de multimédia en plusieurs formats (MP3, MPEG, etc.). • L'enregistrement des photos, des vidéos et des fichiers audio est disponible selon le modèle du dispositif.	★ ★ ★	• Location API pour l'accès au GPS et Sensor API pour d'autres capteurs (bibliothèques optionnelles). • La localisation basée sur le Wi-Fi n'est pas disponible.	★ ★
Android	• Développement sur Mac OS, Windows et Linux. • Plugins disponibles pour Eclipse.	★ ★	• Bibliothèques pour la prise de photo et pour l'enregistrement des fichiers audio avec contrôle quasi total du matériel. • L'enregistrement de vidéos est désormais disponible sur la version 1.5.	★ ★	• Bibliothèque pour l'accès au GPS, boussole, accéléromètre. • Fonctions disponibles pour le calcule de l'orientation et du niveau de luminosité.	★ ★ ★
Windows Mobile	• Développement sur Windows avec l'outil Microsoft Visual Studio et le Windows Mobile SDK.	★ ★ ★	• L'exécution des documents multimédias est assurée par l'API Windows Media Player. • L'enregistrement des photos, des vidéos et des fichiers audio est disponible selon le modèle du dispositif.	★ ★	• La bibliothèque GPS Intermediate Driver permet d'accèder au GPS intégré si présent. • Le positionnement à partir du niveau de signal Wi-Fi peut être codifié.	★ ★
iPhone OS	• Développement seulement sur Mac OS avec l'outil Xcode. Des émulateurs sont fournis avec cet outil.	★	• Lecture audio de qualité. Visionnements de photos et vidéos sont aussi disponibles. • Le support à la prise de photos et à l'enregistrement des vidéos varie selon le modèle de terminal. Néanmoins la qualité des photos et des vidéos est inférieure à celles d'autres plates-formes.	★ ★	• La bibliothèque Core Location Framework offre l'accès au positionnement par A-GPS, GPS et cellule GSM. Le positionnement *indoor* est assuré par la détection de signaux Wi-Fi et des requêtes sur la base de données Skyhook Wireless.	★ ★ ★
Symbian OS	• Développement sur Windows XP et Linux avec Carbide C++ v1.3 (extension d'Eclipse). Plugin disponible pour l'outil Microsoft Visual Studio. Émulateurs disponibles sur d'autres SDK qui varient selon le constructeur de dispositif	★ ★ ★	• Bibliothèques pour la prise de photo et pour l'enregistrement d'audio avec contrôle quasi total du matériel. L'enregistrement de vidéos dépend du modèle du dispositif.	★ ★ ★	• Accès à la localisation par GPS, A-GPS, cell-ID et Wi-Fi (niveau du signal). • Des bibliothèques pour l'accès à un accéléromètre et une boussole.	★ ★ ★

Tableau 23 - Comparaison entre les plates-formes mobiles.

	Environnement d'exécution		Particularité		Fragmentation	
J2ME MIDP	• Modèle Sand Box (similaire aux Applets sur le Web). Une application ne peut pas accéder aux données d'autres applications J2ME. • Seule la communication par sockets est disponible sur certains modèles de terminaux. • Bibliothèques optionnelles pour l'accès aux données de l'utilisateur sur le dispositif (contacts, agenda, etc.). • Mémoire d'exécution limitée par application et indépendante de la taille de mémoire du dispositif.	★ ★	• Interface visuelle moins aboutie. • Absence de chargement dynamique de code. • Le contrôle de fonctionnalités bas-niveau est limité par les bibliothèques Java. • L'accès à l'appareil photo est disponible, cependant le réglage du niveau de flash est indisponible.	★ ★	• Une application qui utilise seulement les APIs du noyau possède de fortes chances de fonctionner sur plusieurs modèles. • Néanmoins, plusieurs fonctions (telles que l'accès au multimédia et au positionnement) sont optionnelles et varient selon les dispositifs.	★
Android	• Chaque application s'exécute sur sa propre machine virtuelle. Elles sont structurées selon le modèle orienté composant. Les applications peuvent communiquer par RPC (Remote Procedure Call).	★ ★ ★	• Des bibliothèques pour accéder et contrôler quasiment toutes les fonctionnalités du dispositif. Par exemple, activer et désactiver le Wi-Fi ou la fonction téléphone.	★ ★ ★	• La fragmentation n'est pas encore un problème compte tenu de la faible quantité de dispositifs.	★ ★
Windows Mobile	• Les applications développées avec C/C++ s'exécutent directement sur le système d'exploitation et accèdent aux fonctions du dispositif en tirant profit de l'API Win32. • Communication intra-processus sur certains smartphones en utilisant le modèle COM (Component Object Model).	★ ★ ★	• Le système d'exploitation utilise les DLL et des clés de registre semblablement aux versions Windows desktop.	★ ★	• La fragmentation est moins importante que celle de J2ME. Cependant, le support de certaines fonctionnalités dépend de la version du système d'exploitation et des caractéristiques matérielles du terminal mobile.	★ ★
iPhone OS	• Modèle Sand Box. Une application ne peut pas accéder aux données d'autres applications sur le système d'exploitation. • Une application tierce est exécutée à chaque fois et elle ne peut pas s'exécuter en tâche de fond.	★ ★	• Une bibliothèque pour le contrôle de l'écran tactile.	★ ★ ★	• La fragmentation est déjà un problème pour les applications multimédias malgré la faible quantité de modèles des dispositifs.	★
Symbian OS	• Le système d'exploitation est multitâche. Les applications peuvent communiquer par sockets. Un nombre important de bibliothèques est disponible pour l'accès aux fonctionnalités des terminaux mobiles (réseaux, multimédia, fonctions téléphoniques, etc.) et pour la lecture/écriture des données (contacts, accès au système de fichiers).	★ ★ ★	• La création d'un serveur Web qui exécute sur un terminal mobile Symbian OS est désormais possible.	★ ★ ★	• La fragmentation est moins importante que celle de J2ME. Cependant, le support de certaines fonctionnalités dépend de la version du système d'exploitation et des caractéristiques matérielles du terminal mobile.	★ ★

Tableau 24 - Comparaison des activités du déploiement sur les plates-formes mobiles.

	Configuration		Packaging		Transfert	
J2ME MIDP	• Création d'un Manifest.MF avec la description des classes, des ressources et de la classe principale. • Description des bibliothèques exigées pour l'exécution du MIDlet.	★ ★	• Génération d'un .jar avec le code et les ressources qui composeront le MIDlet Suite. • Possibilité de certifier l'application.	★ ★	• Transfert via OTA (Over The Air) à l'aide d'un navigateur Web. • Transfert manuel par USB • Échange d'applications certifiées entre dispositifs mobiles par Bluetooth et MMS.	★ ★ ★
Android	• Création d'un manifeste avec la description des composants, ressources, bibliothèques. • Description des droits d'accès.	★	• Génération d'un fichier .apk (Android package). • Toutes les applications doivent être certifiées.	★ ★	• Transfert via OTA à l'aide de l'application Android Market. • Transfert manuel par USB.	★ ★
Windows Mobile	• Description d'un fichier .ini avec le nom de l'application, ses ressources et les .dll à utiliser. • Possibilité de décrire les clés du registre Windows qui seront modifiées lors de l'installation et de la désinstallation.	★ ★	• Génération d'un .cab avec l'exécutable et les .dll exigées pour l'exécution. • Toutes les applications doivent être certifiées.	★ ★	• Transfert par OTA (Over the Air) • Email, Active Sync, etc.	★ ★ ★
iPhone OS	• Création d'un manifeste avec la description des ressources (les fichiers .nib, les icônes) et des informations sur l'application (version, nom à afficher, le nom de la classe principale) • Description des pré-requis pour l'exécution de l'application (la présence d'un GPS, du service SMS, etc.).	★ ★ ★	• Génération d'un *bundle* qui contient les fichiers décrits sur le manifeste. • Outil Xcode pour aider le processus de packaging.	★ ★	• Transfert exclusivement par OTA en utilisant l'application App Store.	★
Symbian OS	• Description d'un Manifest avec les ressources et le code à déployer sur le terminal mobile.	★	• Génération d'un fichier .sis (Symbian OS Installation System) qui contient tous les fichiers de l'application. • Une application Symbian Os peut être certifiée afin d'avoir plus d'accès aux fonctionnalités du dispositif.	★ ★	• Transfert via OTA à l'aide d'un navigateur Web et, plus récemment, par le portail Ovi Store. • Transfert manuel par USB et Bluetooth.	★ ★ ★

Tableau 25 - Comparaison des activités du déploiement sur les plates-formes mobiles.

	Installation et Désinstallation		Activation et Désactivation	
J2ME MIDP	• Installation activée par l'utilisateur lors d'un transfert par USB, Bluetooth, et MMS. • Dans le cas d'un transfert par OTA, le navigateur peut activer l'installation après une confirmation de l'utilisateur. • Désinstallation manuelle.	★★	• Activation et désactivation manuelles • Une application peut être démarrée par le système d'exploitation en enregistrant des alarmes d'activation à l'aide de l'API Push Registry. • Une application J2ME MIDP 2.0 peut activer une autre application, néanmoins elle ne contrôle pas son cycle de vie.	★ ★ ★
Android	• Installation activée par l'utilisateur lors d'un transfert par USB. • Installation activée également à partir de l'application Android Market. • Désinstallation manuelle.	★★ ★	• Activation et désactivation manuelles. • Une application peut déléguer son activation/désactivation à une autre application (en réalité, un autre composant).	★ ★ ★
Windows Mobile	• Installation déclenchée par l'utilisateur sur l'interface graphique après le transfert du fichier d'installation (.cab). • Désinstallation manuelle.	★★	• Activation et désactivation manuelles • Possibilité d'utiliser Wap Push pour activer des applications.	★ ★
iPhone OS	• Installation déclenchée à partir de l'application App Store. • Désinstallation pouvant être effectuée manuellement par l'utilisateur ou à distance par Apple	★★	• Activation et désactivation manuelles. • Des méthodes de démarrage du type Push sont prévues pour la prochaine version du SDK (version 4.0).	★
Symbian OS	• Installation déclenchée par l'utilisateur sur l'interface graphique après le transfert du fichier d'installation (.sis). • Transfert de l'application par OVI Store permettant à ce logiciel de déclencher automatiquement une installation. • Désinstallation manuelle.	★★ ★	• Activation et désactivation manuelles • Une application peut activer une autre application, néanmoins sans pouvoir contrôler son cycle de vie. • Une application peut s'enregistrer pour démarrer chaque fois que le téléphone est allumé.	★ ★ ★

Tableau 26 - Comparaison des activités du déploiement sur les plates-formes mobiles.

	Adaptation/Reconfiguration		Mise à jour	
J2ME MIDP	• Changement manuel des droits d'accès à une application. L'opération exige un arrêt préalable de l'application.	★	• La mise à jour est activée lorsque l'utilisateur essaie d'installer une application avec le même nom qu'une application déjà installée. • Une application peut être codée pour démarrer sa propre mise à jour.	★★
Android	• Possibilité d'écouter des événements de changement de configuration du dispositif. Des exemples d'événements sont : changement des polices, des paramètres de dates/régions et des modifications physiques du terminaux (orientation de l'écran, activation d'un clavier).	★★★	• La mise à jour est activée lorsque l'utilisateur essaie d'installer une application dont la valeur de la version décrite sur le manifeste diffère de celle de l'application disponible sur le téléphone. • Une application peut être codée pour informer l'utilisateur d'une mise à jour. Néanmoins, l'utilisateur doit démarrer manuellement la mise à jour à partir de l'application Android Market.	★★
Windows Mobile	• Changement manuel des droits d'accès à une application. L'opération exige un arrêt préalable de l'application. • Des clés du registre Windows peuvent être modifiées afin de signaler un changement de configuration.	★★	• La mise à jour est activée lorsque l'utilisateur essaie d'installer une application avec le même nom qu'une application déjà installée. • Une application peut être codée pour démarrer sa propre mise à jour.	★★
iPhone OS	• Changement manuel de préférences d'une application sur le logiciel Settings. • Une application peut être programmée pour réagir automatiquement au changement de préférences.	★★	• Mises à jour signalées sur l'application App Store.	★★★
Symbian OS	• Changement manuel des droits d'accès d'une application. L'opération exige un arrêt préalable de l'application.	★	• La mise à jour est activée lorsque l'utilisateur essaie d'installer une application avec le même nom d'une application déjà installée. • Les mises à jour seront aussi signalées sur le portail Ovi Store.	★★

Mobilité et sensibilité au contexte pour la gestion de documments multimédias personnels : CoMMediA

11 BIBLIOGRAPHIE

[Adachi, 2001] Adachi, F. Wireless past and future – evolving mobile communications systems. *IEICE Transaction on Fundamentals*, vol E84-5, n°1, pp55-60, 2001.

[Alexander, 2005] Alexander N. Personal Photo Annotation. Thèse de doctorat en Informatique University of East Anglia, Norwich – Angleterre, Septembre 2005.

[Ali *et al*., 2006] Ali, S., Torabi, T. and Ali, H. A Case for Business Process Deployment for Location Aware Applications. *IJCSNS International Journal of Computer Science and Network Security*, Vol.6 No.8A, 2006.

[Ames *et al*., 2007] Ames M. and Naaman M. Why We Tag: Motivations for Annotation in Mobile and Online Media. **In** : Proc. of Conference on Human Factors in computing systems (CHI 2007), San Jose, CA, USA, 2007.

[Ayed *et al.*, 2008] Ayed, D. and coll. CADeComp: Context-aware deployment of component-based applications. *Journal of Network and Computer Applications*. vol. 31, pp.224-257, 2008.

[Ajmani *et al.*, 2006] Ajmani, S., Liskov, B., Shrira, L. Modular software upgrades for distributed systems. **In**: Proc. of European Conference on Object-Oriented Programming (ECOOP), Jul. 2006.

[Baldaulf *et al.*, 2007] Baldauf, M., Dustdar, S. and Rosenberg, F. A survey on context-aware systems. *Int. J. Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp.263–277. 2007.

[Barnes *et al.*, 2002] Barnes, S.J. The mobile commerce value chain: Analysis and Future Developments. *International Journal of Information Management*, vol. 22, pp. 91-110, 2002.

[Barkhuus *et al.*, 2008] Barkhuus, L., Brown, B., Bell, M., Sherwood, S., Hall, M., and Chalmers, M. 2008. From awareness to repartee: sharing location within social groups. **In** : Proc. of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems (CHI '08). ACM, pp.497-506. DOI= http://doi.acm.org/10.1145/1357054.1357134, 2008.

[Becker *et al.*, 2003] Becker, J.; Kuropka, D.: Topic-based Vector Space Model. **In** : Proc. of the 6th International Conference on Business Information Systems. 2003. S. 7-12.

[Benamar *et al.*, 2008] Benamar, A., BelKhatir, N and Bendimerad, F.T. A Proposition of Generic Deployment Platform for Component Based Applications. *Journal of Software Engineering*, vol 2 (1), pp. 23-38, 2008 ISSN 1819-4311.

[Bergeret *et al.*, 2009] Bergeret, G., Viana, W., Filho, B., F. Junior, J. C., Gensel, J., Oliver, M-V., Martin, H. Une architecture sensible au contexte pour le développement des systèmes d'administration pour l'enseignement. **In** : Proc. of the 5th French-Speaking Conference on Mobility and Ubiquity Computing. UbiMob '09.

[Berners-Lee *et al.*, 2001] Berners-Lee, T., Hendler, J., Lassila, O. The Semantic Web. *Scientific Americain*, 2001.

[Bilasco *et al.*, 2005] Bilasco, IM., Gensel, J. and Villanova-Oliver, M. STAMP: Adaptable Templates for Synchronized Multimedia Presentations. **In:** Proc. of IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), pp.645-648, 2005.

[Bloehdorn *et al.*, 2005] Bloehdorn, S., Petridis, K., Saathoff, C., Simou, N., Tzouaras, V. , Avrithis, Y., Handschuh, S., Kompatsiaris, Y., Staab, S. and Strintzis, M.G. Semantic annotation of images and videos for multimedia analysis. **In** : Proc. of the 2nd European Semantic Web. 2005.

[Buscaldi *et al.*, 2006] Buscaldi, D., Rosso, P., Peris, P. Inferring geographical ontologies from multiple resources for geographical information retrieval. **In**: Proc. of 3rd SIGIR Workshop on Geographical Information Retrieval, August 2006.

[Butler *et al.,* 2002] Butler, M., Giannetti, F., Gimson, R.B. and T. Wiley. Device Independence and the Web. *IEEE Internet Computing*, vol. 6, pp. 81-86, 2002.

[Castells *et al.*, 2007] Castells, P., Fernandez, M., Vallet., D. An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval. *IEEE Transactions on Knowledge and Data Engineering*, Volume 19 , Issue 2, Pages 261-272, ISSN:1041-4347, 2007.

[Carrillo Ramos, 2007] Carrillo Ramos, A. Agents ubiquitaires pour un accès adapté aux systèmes d'information : le framework PUMAS **[en ligne]**. Thèse de doctorat en Informatique. Université Joseph Fourier, Grenoble I, 2007, 210p. Disponible sur : <http://hal.archives-ouvertes.fr/docs/00/13/69/31/PDF/These-Carrillo-vd.pdf>.

[Carton *et al.*, 2007] Carton, A., Clarke, S., Senart, A. and Chill, V. Aspect-Oriented Model-Driven Development for Mobile Context-Aware Computing. **In** : Proc. of First International Workshop on Software Engineering for Pervasive Computing. Applications, Systems, and Environments (SEPCASE'07) 0-7695-2970-4/07, IEEE, 2007.

[Carzaniga *et al.*, 1998] Carzaniga A., Fuggetta A., Hall R.S., Van Der Hoek A., Heimbigner D., Wolf A.L. A Characterization Framework for Software Deployment Technologies. **In** : Technical Report CUCS-857-98, University of Colorado, 1998.

[Cdecas, 2009] Cdecas. Muséum des ordinateurs de poche. Disponible sur http://cdecas.free.fr/computers/pocket/museum.php, 2009.

[Cemerlang *et al.*, 2006] Cemerlang P., Lim J-H., You Y., Zhang J., Chevallet J-P. Towards automatic mobile blogging. **In** : Proc. of IEEE International Conference on Multimedia and Expo, 2006, 2033--2036.

[Chaari *et al.*, 2005] Chaari, T. Laforest, F. SEFAGI: Simple Environment For Adaptable Graphical Interfaces - Generating user interfaces for different kinds of terminals. **In** : Proc. of International Conference on Enterprise Information Systems (ICEIS), Cyprus. 2005.

[Chai *et al.*, 2008] Chai, Y-M., Zhu, X-Y., Jia, J-P. OntoAlbum: An Ontology Based Digital Photo Management System. **In** : Proc. of ICIAR 2008, LNCS 5112, pp. 263–270, 2008.

[Chan *et al.*, 2003] Chan, A. T and Chuang, SN. MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing. *IEEE Trans. On Software Engineering*, 29(12): 1072–1085, 2003.

[Chakravarthy *et al.*, 2006] Chakravarthy, A., Ciravegna, F. and Lanfranchi, V. AKTiveMedia: cross-media document annotation and enrichment. **In** : Proc. of the fifteenth international Semantic Web conference (ISWC 2006).

[Chefrour, 2005] Chefrour, D. Developing Component Based Adaptative Applications in Mobile Enviroments. **In** : Proc. of ACM Symposium on Applied Computing (SAC), New Mexico, USA, 2005.

[Chen *et al.*, 2000] Chen, G., Kotz, D. A Survey of Context-Aware Mobile Computing Research. Dartmouth Computer Science Technical Report TR2000-381, Hannover, 2000.

[Chen *et al.*, 2004] Chen, H., Finin,T., Joshi, A. Semantic Web in the Context Broker Architecture, **In**: Proc. of Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04), pp.227-241, 2004.

[Chevallet *et al.*, 2005] Chevallet, J-P., Lim, J-H., Vasudha, R. SnapToTell: A Singapore Image Test Bed for Ubiquitous Information Access from Camera. **In** : Proc. of ECIR, LNCS 3408, pp. 530–532, 2005.

[Christensen, 2001] Christensen, C.B. Place and Experience: a Philosophical Topography. Mind, *Oxford University Press*, Vol. 110, N. 439, pp. 789-792, 2001.

[Chu *et al.*, 2004] Chu H., Song H., Wong C., Kurakake S., Katagiri M. Roam, A Seamless Application Framework. *Journal Of Systems And Software*, vol. 69, nº 3, pp. 209-226, 2004.

[ComScore, 2007] Mobile Phone Web User Nearly Equal PC Based Internet User in Japan. Disponible sur: http://www.comscore.com/press/release.asp?press=1742, 2007.

[Cong *et al.*, 2008] Cong, C., Torabi, T. A Framework for Over the Air Provider-Initiated Software Deployment on Mobile Devices. Software Engineering. **In** : Proc. of ASWEC 2008 - 19th Australian Conference, pp 633-638, 2008. DOI:10.1109/ASWEC.2008.4483255.

[Cui *et al.*, 2008] Cui, Y. and Roto, V. How people use the web on mobile devices. **In** : Proc. of International Conference on World Wide Web (WWW '08), pp. 905-914, 2008. DOI : http://doi.acm.org/10.1145/1367497.1367619.

[Davis *et al.*, 2005] Davis, M., Smith, M., Canny, J., Good, N., King, S. Janakiraman R. Towards Context-Aware Face Recognition. **In** : Proc. of ACM Multimedia Conference, Singapore, 2005.

[Dey *et al.*, 2000] Dey, A.K., Abowd, G.D. Towards a Better Understanding of Context and Context-Awareness. **In :** Proc. of the Conference on Human Factors in Computing Systems (CHI'2000), Workshop on the What, Who, Where, When, and How of Context-Awareness, 2000.

[Dey, 2001] Dey, AK. Understanding and Using Context. *Personal and Ubiquitous Computing*, vol.5, n°1, pp.4-7, 2001.

[Donsez, 2006] Donsez, D. Courtage et déploiement dynamiques de composants pour l'infrastructure d'équipements UPnP. **In** : Proc. of UbiMob'06: 3emes Journées Francophones Mobilité et Ubiquité, New York, NY, USA. ACM Press. 2006.

[Dockhorn *et al.*, 2005] Dockhorn P., Pires, F., Sinderen, M. Architectural Patterns for Context-Aware Services Platforms. **In**: Proc. of the Second International Workshop on Ubiquitous Computing (IWUC 2005), pp 3-19, 2005

[Elfatatry, 2007] Elfatatry, A. Dealing with change: components versus services. *Commun. ACM*, vol.50, issue.8, pp. 35-39. 2007 DOI= http://doi.acm.org/10.1145/1278201.1278203.

[Elliott *et al.*, 2008] Elliott, B. and Özsoyoglu, Z. M. 2008. Annotation suggestion and search for personal multimedia objects on the web. **In** : Proc. of the 2008 International Conference on Content-Based Image and Video Retrieval, pp.75-84, 2008.

[Ejigu *et al.*, 2008] Ejigu, D., Scuturici, M., Brunie, L. Hybrid Approach to Collaborative Context-Aware Service Platform for Pervasive Computing. *JCP*, vol. 3, pp.40-50, 2008).

[Ester *et al.*, 1996] Ester, M., Kriegel, H-P., Sander, J ., Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. **In** : Proc. of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226-231. ISBN 1-57735-004-9. 1996.

[Fjellheim, 2006] Fjellheim, T. Over-the-air Deployment of Applications in Multi-Platform Environments. **In** : Proc. of the 2006 Australian Software Engineering Conference (ASWEC'06) 1530-0803/06 IEEE.

[Freund *et al.*, 1999] Freund Y and Schapire R.E. A Short Introduction to Boosting. **In** : Journal of Japanese Society for Artificial Intelligence, 14(5):771-780, September, 1999.

[Freyne *et al.*, 2007] Freyne, J., Varga, E., Byrne, D., Smeaton, A. F., Smyth, B., Jones, G.J.F. Realising context-sensitive mobile messaging. **In** : Proc. of MONET, pp. 25-30, 2007.

[Gajos *et al.*, 2004] Gajos K, Weld D. Supple: Automatically Generating User Interfaces. **In** : Proc. of IUT'2004, Intelligent User Interfaces, 2004.

[Gamma *et al.*, 1995] Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns – Elements of Reusable Object-Oriented Software. **In** : Addison–Wesley, 1995.

[Ganneau, 2009] Ganneau, V. Modèle Utilisateur pour la Plasticité des Interfaces Homme-Machine en Mobilité. Thèse de Doctorat, Université Joseph Fourier-Grenoble1, France, janvier 2009.

[Garnaud *et al.*, 2006] Garnaud, E., Smeaton, A., Koskela, M. Evaluation of a Video Annotation Tool Based on the LSCOM Ontology. **In** : Proc. of International Conference on Semantics And digital Media Technologies (SAMT). 2006.

[Gong *et al.*, 2005] Gong, Z., Hou U, L., Cheang CW. Web Image Semantic Clustering. **In** : Proc. of ODBASE, pp. 1416-1431, 2005.

[Gruber, 2008] Gruber, T. Ontology. Entry in the Encyclopedia of Database Systems, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag. Provides a definition of ontology as a technical term for computer science, tracing its historical context from philosophy and AI. 2008.

[Gruber, 2007] Gruber, T. Collective knowledge systems: Where the Social Web meets the Semantic Web. Web Semant. vol. 6, pp 4-13. 2007. DOI= http://dx.doi.org/10.1016/j.websem.2007.11.011

[Grütter *et al.*, 2007] Grütter, R., Bauer-Messmer, B. 2007. Towards Spatial Reasoning in the Semantic Web: A Hybrid Knowledge Representation System Architecture. Lecture Notes in Geoinformation and Cartography. 349-364.

[Gu *et al.*, 2005] Gu, T., Keng, H., Zhang, Q. A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications*, vol. 28, pp.1-18, 2005.

[Halaschek *et al.*, 2005] Halaschek-Wiener, C., Golbeck, J., Schain, A., Grove, M., Parsia, B., Hendler, J. PhotoStuff – An Image Annotation Tool for the Semantic Web. **In** : Proc. 4th International Semantic Web Conference (ISWC2005); 2005.

[Hammiche *et al.*, 2008] Hammiche, S.;Lopez, B.; Benbernou, S.; Hacid, M.-S. Query Rewriting for Semantic Multimedia Data Retrieval. **In**: Proc. of. Advances of Computational Intelligence in Industrial Systems p.351-p.372, 2008.

[Haralick *et al.*, 1973] Haralick, R. M., Shanmugam, K., Dinstein, I. Textural Features for Image Classification. **In** : Systems, Man and Cybernetics, IEEE Transactions on, vol.3, no.6, pp.610-621, Nov. 1973

[Hardman *et al.*, 1993] Hardman, L., Van Rossum, G., Bulterman, D.C.A. Structured Multimedia Authoring. **In :** Proc. of ACM International Conference on Multimedia, Anaheim, California, pp. 283-289. 1993.

[Harmer, 2003] Harmer, J. A. Mobile Multimedia Services. **In** : BT Technology Journal 21, vol. 3, pp169-180, 2003. DOI= http://dx.doi.org/10.1023/A:1025175518841.

[Hazan, 2008] Hazan, G. Case study of Superwaba applications. 2008. Disponible en ligne sur : http://www.superwaba.com.br/en/casos.asp.

[Hendler, 2008] Hendler, J. Web 3.0: Chicken Farms on the Semantic Web. Computer, vol. 41, no1, pp. 106-108, 2008.

[Henricksen *et al.*, 2004] Henricksen K., Indulska J. "A Software Engineering Framework for Context-Aware Pervasive Computing". PerCom, Orlanda, USA, Mars, pp.77-86, 2004.

[Hens *et al.*, 2007] Hens, R., Turck, F., Dhoedt, B. Runtime Deployment Adaptation for Resource Constrained Devices. **In** : Proc. of IEEE International Conference on Pervasive Services. pp 335-340. 2007.

[Hollink *et al.*, 2004] Hollink L., Nguyen G., Schreiber G., Wielemaker J., Wielinga B., Worring M. Adding Spatial Semantics to Image Annotations. **In** : Actes du 4ème International Workshop on Knowledge Markup and Semantic Annotation, 2004.

[House *et al.*, 2006] House, N.A.V. Distant closeness: Cameraphones and public image sharing. **In** : Proc. of UBICOMP '06 PICS Workshop 2006.

[Hui *et al.,* 2007] Hui, L., Darabi, H., Banerjee, P., Jing Liu. Survey of Wireless Indoor Positioning Techniques and Systems. **In** : Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol.37, no.6, pp.1067-1080, 2007.

[Hwang *et al.*, 2007] Hwang, A., Ahern, S., King, S., Naaman, M., Nair, R., and Yang, J. 2007. Zurfer: mobile multimedia access in spatial, social and topical context. **In** : Proceedings of the 15th international Conference on Multimedia (Augsburg, Germany, September 25 - 29, 2007). MULTIMEDIA '07. ACM, New York, NY, 557-560. DOI= http://doi.acm.org/10.1145/1291233.1291370

[Indulska *et al.,* 2003] Indulska, J., Robinson, R., Rakotonirainy, A., Henricksen, K. Experiences in Using CC/PP in Context-Aware Systems. **In** : Chen, M.-S., Chrysance, P.K., Sloman, M. Zaslavsky, A. (eds.): Proceedings of the 4th International Conference on Mobile Data Management (MDM 2003) (Melbourne, Australia, January 21-24, 2003), Lecture Notes in Computer Science, vol. 2574. Springer-Verlag, Berlin Heidelberg (2003), pp. 247-261.

[Jiang *et al.*, 2006] Jiang, M., Zhihui Y. A Component-Based Service Creation Framework for Mobile Applications. **In** : Proc. of IEEE International Conference on Information Reuse and Integration, pp.307-312, 2006.

[Jones *et al.*, 2001] Jones, C., H. Alani and D. Tudhope. Geographical Information Retrieval with Ontologies of Place. In Proceedings of COSIT'01, pp. 322-335, 2001

[Keshav *et al.*, 2005] Keshav, S. Why cell phones will dominate the future internet. **In** : ACM SIGCOMM Computer Communication Review, vol. 35, pp. 83-86, 2005. DOI : http://doi.acm.org/10.1145/1064413.1064425.

[Ketfi, 2004] Ketfi, M. Une approche générique pour la reconfiguration dynamique des applications à base de composants logiciels. Thèse de doctorat en Informatique. Université Joseph Fourier, Grenoble I, 2004, 202p.

[Kim *et al.*, 2006] Kim, Y., Kim, E., Kim, J. Ontology Based Software Reconfiguration in a Ubiquitous Computing Environment. **In** : The Sixth IEEE International Conference on Computer and Information Technology, 2006. pp.260-267, 2006.

[Kirsch *et al.*, 2005] Kirsch, M., Villanova-Oliver, M., Gensel, J., Martin, H. Une Formalisation du Contexte dans les Environnements Coopératifs Nomades. **In** : Actes de la deuxièmes Journées Francophones: Mobilité et Ubiquité 2005, Grenoble, France, 2005.

[Kirsch, 2006] Kirsch Pinheiro, M. Adaptation Contextuelle et Personnalisée de l'Information de Conscience de Groupe au sein des Systèmes d'Information Coopératifs **[en ligne]**. Thèse de doctorat en Informatique. Université Joseph Fourier, Grenoble I, 2006, 272p. Disponible sur : <http://hal.archives-ouvertes.fr/docs/00/10/84/95/PDF/TheseManueleKirschPinheiro-Final.pdf>.

[Koskela *et al.*, 2007] Koskela, M., Smeaton, AF and Laaksonen, J. Measuring Concept Similarities in Multimedia Ontologies: Analysis and Evaluations. **In** : IEEE Transactions on Multimedia, 9 (5). pp. 912-922. ISSN 1520-9210, 2007.

[Kriens, 2007] Kriens, P. SOA and OSGi. 2007. Disponible sur: http://www.osgi.org/blog/2007/09/soa-osgi.html

[Koolwaaij *et al.*, 2006] Koolwaaij, J., Tarlano, A., Luther, M., Nurmi, P., Mrohs, B., Battestini, A. and Vaidya, R. Context watcher: sharing context information in everyday life. **In** : Proc. of the IASTED International Conference on Web Technologies, Applications, and Services: WTAS 2006: Acta Press 2006.

[Korpipää *et al.*, 2003] Korpipää, P and Mäntyjärvi, J. An Ontology for Mobile Device Sensor-Based Context. **In** : P. Blackburn *et al.* (Eds.): CONTEXT 2003, LNAI 2680, pp. 451–458, 2003.

[Kuo *et al.*, 2007] Kuo, B. Y., Hentrich, T., Good, B. M., and Wilkinson, M. D. Tag clouds for summarizing web search results. **In** : Proceedings of the 16th International Conference on World Wide Web (Banff, Alberta, Canada, May 08 - 12, 2007). WWW '07. ACM, New York, NY, 1203-1204. DOI= http://doi.acm.org/10.1145/1242572.1242766.

[Lacerda *et al.*, 2008] Lacerda, Y.A., de Figueiredo, H.F., de Souza Baptista, C., Sampaio, M.C. PhotoGeo: A Self-Organizing System for Personal Photo Collections. **In** : IEEE International Symposium on Multimedia (ISM). Volume, Issue, 15-17 Dec. 2008 Page(s):258 - 265. Digital Object Identifier 10.1109/ISM.2008.81.

[Lee *et al.*, 2005] Lee, I. and Kim, J. Use Contexts for the Mobile Internet: A Longitudinal Study Monitoring Actual Use of Mobile Internet Services. **In** : International Journal of Human-Computer Interaction, vol. 18, pp. 269-292, 2005.

[Lemlouma, 2004] Lemlouma T. Architecture de négociation et d'adaptation de Services Multimédia dans des Environnements Hétérogènes. Thèse de Doctorat, Institut National Polytechnique de Grenoble, Grenoble, France, avril 2004.

[Lestideau *et al.*, 2001] Lestideau, V., Belkhatir, N., Cunin, P. Un environnement de déploiement automatisé d'applications d'entreprise. **In** : Revue Génie Logiciel, Mai. 2001, pp. 23-31.

[Lehikoinen *et al.*, 2007] Lehikoinen, J., Aaltonen, A., Huuskonen, P., Salminen, I. Personal Content Experience: Managing Digital Life in the Mobile Age. **In** : John Wiley & Sons, Ltd, 2007. ISBN: 978-0-470-03464-4; 382 pages.

[Li *et al.*, 2005] Li, B., Zhou, Y., Wang, Y., and Mo, J. 2005. Matrix-based component dependence representation and its applications in software quality assurance. *SIGPLAN Not*. Vol. 40, pp 29-36, 2005.

[Liu *et al.,* 2006] Liu P., Zhao T., Yu X. Application-oriented comparison and evaluation of six semantic similarity measures based on WordNet, **In** : International Conference on Machine Learning and Cybernetics, 2006.

[Louberry *et al.*, 2008] Louberry, C., Dalmau, M., and Roose, P. Software architecture for dynamic adaptation of heterogeneous applications. **In** : Proceedings of the 8th international Conference on New Technologies in Distributed Systems - NOTERE '08. 2008. DOI= http://doi.acm.org/10.1145/1416729.1416770

[Lum *et al.,* 2003] Lum, W.Y., Lau, F.C.M. User-Centric Content Negotiation for Effective Adaptation Service in Mobile Computing. **In** : *IEEE Transactions on Software Engineering*, vol. 29, no. 12, pp. 1100-1111, doi:10.1109/TSE.2003.1265524. 2003.

[Madigan *et al.*, 2005] Madigan, D., Einahrawy, E., Martin, R., Krishnan, P., Krishnakumar, A. Bayesian indoor positioning systems, **In** : IEEE INFOCOM 2005, vol. 2, pp. 1217–1227, 2005.

[Matellanes *et al.*, 2005] Matellanes A., Evans A., Erdal B. Creating an application for automatic annotations of images and video. **In** : International Workshop on Semantic Web Annotations for Multimedia (SWAMM), Edinburgh, Scotland, 2006.

[Miller *et al.*, 2005] Miller CA., Funk W., Goldman, R., Meisner, J. and Wu, P. Implications of Adaptive vs. Adaptable UIs on Decision Making: Why "Automated Adaptiveness" is Not Always the Right Answer. **In** : Proceedings of the 1st International Conference on Augmented Cognition, Las Vegas, NV; July 22-27, 2005.

[Miron *et al.*, 2007] Miron, A.D., Gensel, J. and Villanova-Oliver, M. Towards the Geo-spatial Querying of the Semantic Web with ONTOAST. **In** : 7th International Symposium on Web and Wireless GIS (W2GIS 2007), Cardiff, UK.

[Muller *et al.*, 2005] Muller J., Lenhart T., Henrici D., Hillenbrand M. and Muller P. Developing Web Applications for Mobile Devices. **In** : Proc. of the First International Conference on Distributed Frameworks for Multimedia Applications (DFMA'05), 2005.

[Moisuc, 2007] Moisuc, B. Conception et Mise en Oeuvre de Systèmes d'Information Spatio-Temporelle Adaptatifs : le framework ASTIS. Thèse de Doctorat, préparée au Laboratoire d' Informatique de Grenoble dans l'Université Joseph Fourrier, 2007.

[Monaghan *et al.*, 2006] Monaghan, F. and O'Sullivan, D. Automating Photo Annotation using Services and Ontologies. **In** : MDM'06: Proceedings of the 7th International Conference on Mobile Data Management, IEEE Computer Society, Washington DC, USA, pp 79-83. 2006.

[Naaman et *al.*, 2004] Naaman, M., Harada, S., Wang, Q., Garcia-Molina, H., Paepcke, A. Context Data in Geo-Referenced Digital Photo Collections. **In** : Proc. Twelfth ACM International Conference on Multimedia, October 2004.

[Naphade *et al.*, 2006] Naphade, M., Smith J.R., Tesic J., Chang S.-F., Hsu W., Kennedy L., Hauptmann A., Curtis J. Large-Scale Concept Ontology for Multimedia. **In** : IEEE Multimedia, 13(3) July-Sept, pp.86–91, 2006.

[Nars, 2005] Nars E. Visualisation multi-échelles pour photos personnelles. **In** : Actes de la 17ème conférence francophone sur l'Interaction Homme-Machine (IHM 2005), ACM Press, International Conference Proceedings Series, pp. 259-262, Toulouse, Septembre 2005.

[OMA *et al.*, 2001] OMA (Open Mobile Alliance). User Agent Profiling Specification. Specification disponible sur: http://www.openmobilealliance.org/tech/affiliates/wap/wap-248-uaprof-20011020-a.pdf.

[O'Hare *et al.*, 2006] O'Hare, N., Lee, H., Cooray, S., Gurrin, C., Jones, Smeaton, A. F. Mediassist: Using content-based analysis and context to manage personal photo collections. **In** : Proc. of CIVR, pp. 529-532, Tempe, Arizona, USA, 2006.

[O'Hare *et al.*, 2007] O'Hare, N., Gurrin,C., Jones,G. J. F., Lee,H., O'Connor, N. E. and Smeaton, A.F., 2007. Using text search for personal photo collections with the MediAssist system. **In** : Proceedings of ACM SAC 2007.

[O'Reilly, 2005] O'Reilly, T. What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software, September 30, 2005. Disponible sur: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

[Paterno *et al.*, 2004] Paterno F., Mori G., Santoro C. Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions. **In** : IEEE Transactions on Software Engineering, vol. 30, nº 8, p. 507-520, 2004.

[Parra *et al.*, 2008] Parra, C.A., Duchien, L. Model-Driven Adaptation of Ubiquitous Applications. **In** : 1st International Workshop on Context-aware Adaptation Mechanisms for Pervasive and Ubiquitous Services (CAMPUS 08). ECEASST, 2008.

[Papazoglou, 2003] Papazoglou, M. P. Service-Oriented Computing: Concepts, Characteristics and Directions. **In :** Procs of the 4th International Conference on Web Information Systems Engineering (WISE 03), Dec. Washington, DC, USA. IEEE Computer Society, pp. 3-12, 2003.

[Polakovic *et al.*, 2005] Polakovic, J., Özcan, A. E., Stefani, J.-B. Reconfiguration dynamique d'un système à composants, **In** : 4ème Conférence Francophone autour des Composants Logiciels (avec CFSE-RENPAR 2005), Le Croisic, France, April 2005.

[Popescu *et al.*, 2007] Popescu, A., Millet, C and Moëllic, P. Ontology driven content based image retrieval. **In** : Proc. of the 6th ACM international Conference on Image and Video Retrieval (Amsterdam, The Netherlands, July 09 - 11). CIVR '07. ACM, New York, NY, 387-394, 2007.

[Puerta, 2005] Puerta, A. A Better Future for UI Tools through Engineering. **In** : Proc. of the ACM CHI 2005 Workshop: The Future of User Interface Design Tools, 2005.

[Preuveneers *et al.*, 2007] Preuveneers, D., Berbers, Y. Towards context-aware and resource-driven self-adaptation for mobile handheld applications. **In** : Proc. of the ACM symposium on Applied computing, pp. 1165-1170. 2007.

[Preuveneers *et al.*, 2004] Preuveneers, D., Bergh, J. V.D., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V., De Bosschere., K. Towards an extensible context ontology for Ambient Intelligence. **In** : Second European Symposium on Ambient Intelligence. Volume 3295 of LNCS., Eindhoven, The Netherlands, Springer (2004) pp. 148 – 159. 2004.

[Preuveneers *et al.*, 2007] Preuveneers, D and Berbers, Y. Towards context-aware and resource-driven self-adaptation for mobile handheld applications. **In** : Proceedings of the 2007 ACM symposium on Applied computing, Seoul, Korea: ACM, 2007, pp. 1165-1170.

[Prekop et al., 2003] Prekop, P. and Burnett, M. Activities, context and ubiquitous computing. *Special Issue on Ubiquitous Computing Computer Communications*, pp. 26-37, 2003.

[Raento *et al.*, 2005] Raento, M., Oulasvirta, A., Petit, R. ContextPhone: a prototyping platform for context-aware mobile applications. **In** : Pervasive Computing, IEEE, vol. 4, pp. 51-59,2005.

[Rey, 2006] Rey, G. Méthode pour la modélisation du contexte d'interaction.Ingénierie des Systèmes d'Information. Vol. 11, pp.141-166, 2006.

[Ryan et al., 1997] Ryan, N., Pascoe, J. and Morse, D. Enhanced reality fieldwork: The context-aware archaeological assistent. *Computer Applications in Archaeology*. Vol. 8, pp.22-32, 1997.

[Robertson *et al.*, 1994] Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M., Gatford, M. Okapi at TREC-3. **In** : Proceedings of the Third Text REtrieval Conference (TREC 1994). Gaithersburg, USA, November 1994.

[Roisin, 1999] Roisin, C. Documents structurés multimédia, Habilitation de diriger les recherches, spéc. Informatique, Institut National Polytechnique de Grenoble. 1999.

[Rutheford *et al.*, 2002] Rutherford, M. J., Anderson, K. M., Carzaniga, A., Heimbigner, D., and Wolf, A. L. Reconfiguration in the Enterprise JavaBean Component Model. **In** : Proceedings of the IFIP/ACM Working Conference on Component Deployment. J. M. Bishop, Ed. Lecture Notes In Computer Science, vol. 2370. Springer-Verlag, London, 67-81. 2002.

[Saha *et al.*, 2001] Saha, S., Jamtgaard, M., Villasenor, J. Bringing the wireless Internet to mobile devices, Computer, vol.34, no.6, pp.54-58, 2001. Disponible sur : http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=928622&isnumber=20074.

[Salton *et al.*, 1975] Salton, G., Wong, A. and Yang, C. S. A Vector Space Model for Automatic Indexing. Communications of the ACM, vol. 18, nr. 11, pp 613–620. 1975.

[Sarvas *et al.*, 2004a] Sarvas R., Herrarte E., Wilhelm A., Davis M., Metadata creation system for mobile images, **In** : Proc. of International Conference on Mobile Systems, Applications, and Services (MobiSys '04), pp. 36-48, 2004.

[Sarvas et *al.*, 2004b] Sarvas R., Viikari M., Pesonen J., Nevanlinna H. MobShare: Controlled and Immediate Sharing of Mobile Images. **In** : the Proceedings of ACM Multimedia 2004, New York, NY, USA. ACM Press 2004.

[Sarvas *et al.*, 2005] Sarvas, R., Oulasvirta, A., Jacucci, G. Building social discourse around mobile photos: a systemic perspective. **In** : proc. of 7th international conference on human computer interaction with mobile devices and services. Salzburg, Austria, pp. 31-38, 2005.

[Schreiber *et al.*, 2001] Schreiber, G., Dubbeldam, B.,Wielemaker, J., and Wielinga, B. Ontology-Based Photo Annotation. **In** : IEEE Intelligent Systems, vol. 16, pp. 66-74 , 2001.

[Schroeter *et al.*, 2003] Schroeter, R., Hunter, J. and Kosovic, D. Vannotea, A collaborative video indexing, annotation and discussion system for broadband networks. **In** : Proceedings of the K-CAP. Workshop on Knowledge Markup and Semantic Annotation, October 2003, Florida.

[Schilit *et al.*, 1994] Schilit, B. and Theimer, M. Disseminating active map information to mobile hosts. *IEEE Network*, Vol. 8, pp.22-32, 1994.

[Schweer *et al.*, 2007] Schweer, A., Hinze, A., The Digital Parrot: Combining Context-Awareness and Semantics to Augment Memory, Workshop "Supporting Human Memory with Interactive Systems". **In** : HCI Conference, September 4th, 2007, Lancaster, UK.

[Shadbolt *et al.*, 2006] Shadbolt, N., Berners-Lee, T. and Hall, W. The Semantic Web Revisited. **In** : IEEE Intelligent Systems, IEEE Educational Activities Department, Piscataway, NJ, USA, 21 (3), 96-101, 2006.

[Shevade *et al.*, 2007] Shevade, B., Sundaram, H., and Xie, L. Modeling personal and social network context for event annotation in images. **In** : Proc. of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries. JCDL '07. ACM, New York, NY, 127-134. 2007. DOI= http://doi.acm.org/10.1145/1255175.1255200.

[Shrestha, 2007] Shrestha S: Mobile Web Browsing: Usability Study. **In** : Proc. of the 4th Intl. Conf. on Mobile Technology, Applications and Systems – Mobility, 2007.

[Sinclair *et al.*, 2008] Sinclair, J. Cardew-Hall. The folksonomy tag cloud: when is it useful? **In** : Journal of Information Science, vol. 34, pp. 15-29, 2008.

[Strimpakou *et al.*, 2006] Strimpakou, M. A., Roussaki, I. G., Anagnostou, M. E., A Context Ontology for Pervasive Service Provision. **In** : Actes de International IEEE Conference on Advanced Information Networking and Applications (AINA'06), 2006.

[Szummer *et al.*, 1998] Szummer, M., Picard, R.W. "Indoor-outdoor image classification," Content-Based Access of Image and Video Database. **In** : IEEE International Workshop on , vol., no., pp.42-51, 3 Jan 1998.

[Szyperski, 2002] Szyperski C., Component Software: Beyond Object-Oriented Programming. Second edition, ACM Press, Component Software Series, Addison-Welsey. 2002.

[Tan *et al.*, 2006] Tan, X., Chen, S., Zhou, Z., Zhang, F. Face recognition from a single image per person: A survey. **In** : Pattern Recognition, vol. 39, Sep. 2006, pp. 1725-1745.

[Thevenin, 2001] Thevenin, D. Adaptation en Interaction Homme-Machine : Cas de la plasticité. Thèse de doctorat de l'université Joseph Fourier, Grenoble, 2001.

[Toyama *et al.*, 2003] Toyama K., Logan R., Roseway A. Geographic location tags on digital images. **In :** Proc. of 11th ACM international Conference on Multimedia (MULTIMEDIA '03), Berkeley, CA, USA, ACM Press, pp. 156-166, 2003.

[Tuffield *et al.*, 2006] Tuffield, M., Harris, S., Dupplaw, D. P., Chakravarthy, A., Brewster, C., Gibbins, N., O'Hara, K., Ciravegna, F., Sleeman, D., Wilks, Y. and Shadbolt, N. R., 2006. Image annotation with Photocopain. **In :** Proc. of First International Workshop on Semantic Web Annotations for Multimedia (SWAMM 2006) at WWW2006, 2006.

[Vailaya *et al.*, 2001] Vailaya, A., Figueiredo, M.A.T., Jain, A.K., Hong-Jiang Zhang, Image classification for content-based indexing. **In** : IEEE Transactions on Image Processing, vol.10, no.1, pp.117-130, Jan 2001.

[Viana *et al.*, 2004] Viana, W., Filho, B., Magalhaes, Katy., Giovano, C., Andrade, R., Machado, J. Mobis: A Solution For The Development Of Secure Applications For Mobile Device. **In** :

Proc. of the Telecommunications and Networking Conference – ICT, 2004. DOI : http://dx.doi.org/10.1007/b99377.

[Viana *et al.*, 2007a] Viana, W., Filho, B., Oliver, M-V., Gensel, J., Martin, H. PhotoMap Automatic Spatiotemporal Annotation for Mobile Photos. **In** : W2GIS -7th International Symposium on Web and Wireless Geographical Information Systems, Cardif. Web and Wireless Geographical Information Systems. Berlin : Springer Berlin / Heidelberg, 2007. v. 4857. p. 187-201.

[Viana *et al.*, 2007b] Viana, W., Filho, B., Gensel, J., Oliver, M-V., Martin, H. A Semantic Approach and a Web Tool for Contextual Annotation of Photos Using Camera Phones. **In** : 8th International Conference on Web Information Systems Engineering, 2007, Nancy. Web Information Systems Engineering WISE 2007. Berlin / Heidelberg : Lecture Notes in Computer Science - Springer, 2007. v. 4831. p. 225-236.

[Viana *et al.*, 2008a] Viana, W., Andrade, R. XMobile: A MB-UID environment for semi-automatic generation of adaptive applications for mobile devices. Journal of Systems and Software, v. 81, p. 382-394, 2008.

[Viana *et al.*, 2008b] Viana, W., Bringel Filho, J., Gensel, J., Villanova-Oliver, M., Martin, H. PhotoMap: from location and time to context-aware photo annotations. Journal of Location-Based Services , v. 2, p. 211-235, 2008.

[Viana *et al.*, 2008c] Viana, W, Oliver, M-V., Gensel, J., Martin, H. Photo Context as a Bag of Words. **In** : Tenth IEEE International Symposium on Multimedia, 2008, Berkeley, CA, USA. Tenth IEEE International Symposium on Multimedia, 2008. p. 310-315.

[Viana *et al.*, 2008d] Viana, W, Moisuc, B., Oliver, M-V., Gensel, J., Martin, H. Semantic keyword-based retrieval of photos taken with mobile devices. **In** : MOMM - The 6th International Conference on Advances in Mobile Computing and Multimedia, Linz, Austria. The 6th International Conference on Advances in Mobile Computing and Multimedia. New York, NY, USA : ACM, 2008. v. 1. p. 192-199.

[Viana *et al.*, 2008e] Viana, W., Villanova-Oliver, M., Gensel, J., and Martin, H. 2007. Annotation contextuelle automatique avec PhotoMap. **In** : Proc. of the 4th French-Speaking Conference on Mobility and Ubiquity Computing (Saint Malo, France, May 28 - 30, 2008). UbiMob '08, vol. 277. ACM, New York, NY, 89-90. DOI= http://doi.acm.org/10.1145/1376971.1376992

[Viana *et al.*, 2008f] Viana, W., Bringel Filho, J., Gensel, J., Villanova-Oliver, M., Martin, H. Une architecture pour le développement de systèmes Web adaptatifs. Application au domaine de l'administration des enseignements. Rapport Scientifique de l'année 2008.

[Viana *et al.*, 2009a] Viana, W., Gensel, J., Oliver, M-V., Martin, H. Semantic indexing of georeferenced photos. *Journal of GIS and Spatial Analysis*, v. 19, p. 169-189, 2009.

[Viana *et al.*, 2009b] Viana, W, Moisuc, B., Miron, A. Dia, Oliver, M-V., Gensel, J., Martin, H.. Towards the Semantic and Context-Aware Management of Mobile Multimedia. *Multimedia Tools and Applications*, 2009 (Accepté).

[Viana *et al.*, 2009c] Viana, W., Filho, B., F. Junior, J. C., Gensel, J., Oliver, M-V., Martin, H. CAUS: Uma arquitetura sensível ao contexto e orientada a componentes para sistemas de administração de ensino. **In** : SBCUP - I Simpósio Brasileiro de Computação Ubíqua e Pervasiva, 2009, Bento gonçalves.

[Viana *et al.*, 2009d] Viana, W., Filho, B., Gensel, J., Oliver, M-V., Martin, H. Aide au développement et au déploiement d'applications mobiles et sensibles au contexte : l

architecture CoMMediA. **In** : INFORSID 2009, Toulouse. Actes de l'atelier ERTSI. 2009. v. 1. p. 150-166.

[Viola *et al.*, 2001] Viola, P. and Jones, M. Rapid object detection using boosted cascade of simple features. **In** : IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[Villanova-Oliver, 2002] Villanova-Oliver, M. Adaptabilité dans les systèmes d'information sur le Web : Modélisation et mise en œuvre de l'accès progressif **[en ligne]**. Thèse de doctorat en Informatique. Institut National Polytechnique de Grenoble, 2002, 216p. Disponible sur : <http://tel.archives-ouvertes.fr/docs/00/04/70/61/PDF/tel-00006759.pdf>.

[Voorhees *et al.*,1999] Voorhees E.M. The TREC-8 question answering track report. **In** : Proc. of the 8th Text Retrieval Conference, Gaithersburg, Maryland, USA, 1999, p. 77-82.

[Wagelaar, 2005] Wagelaar, D. Context-Driven Model Refinement. Model Driven Architecture. Springer Berlin / Heidelberg, pp. 189-203, ISSN : 978-3-540-28240-2, 2005.

[Wagelaar *et al.*, 2006] Wagelaar, D and Van Der Straeten, R. A Comparison of Configuration Techniques for Model Transformations. **In** : ECMDA-FA, LNCS 4066, pp. 331-345, 2006.

[Wang *et al.*, 2005] Wang, A. J. A., Qian, K. Component-Oriented Programming. Wiley-Interscience, 2005.

[Wang *et al.*, 2006] Wang, C., Jing F., Zhang L., Zhang H. Scalable search-based image annotation of personal images. **In** : Proceedings of the 8th ACM international Workshop on Multimedia information Retrieval (Santa Barbara, California, USA, October 26 - 27, 2006). MIR '06. ACM, New York, NY, 269-278. 2006.

[Wang *et al.*, 2007] Wang, S., Jing, F., Jing, F., He, J., Du, Q., and Zhang, L. IGroup: presenting web image search results in semantic clusters. **In** : Proc. of the SIGCHI Conference on Human Factors in Computing Systems. CHI '07. ACM. 2007.

[Wei *et al.,* 2008] Wei, E. J. and Chan, A. T. Semantic Approach to Middleware-Driven Run-Time Context-Aware Adaptation Decision. **In** : Proceedings of the 2008 IEEE international Conference on Semantic Computing (August 04 - 07, 2008). ICSC. IEEE Computer Society, Washington, DC, 440-447. DOI= http://dx.doi.org/10.1109/ICSC.2008.49

[Wesson *et al.*, 2005] Wesson, JL and Van Der Walt, DF. Implementing Mobile Services: Does the Platform Really Make a Difference. **In** : Proceedings of SAICSIT, Pages 208-216, 2005.

[Wilson *et al.*, 2006] Wilson, P. I. and Fernandez, J. Facial feature detection using Haar classifiers. **In** : J. Comput. Small Coll. 21, 4 (Apr. 2006), 127-133.

[W3C, 2007] W3C (World Wide Web Consortium). Standard Composite Capability/Preference Profiles (CC/PP) version 2: Structure and Vocabularie. Disponíble sur: http://www.w3.org/Mobile/CCPP/. 2007.

[Yamaba *et al.*, 2005] Yamaba, T., Takagi, A. and Nakajima, T. Citron: A context information acquisition framework for personal devices. **In** : Proc. of 11th International Conference on Embedded and real-Time Computing Systems and Applications, pp.489-495. 2005.

[Zachariadis et *al.*, 2006] Zachariadis S., Mascolo, C and Emmerich, W. The SATIN Component System - A Meta Model For Engineering Adaptable Mobile Systems. **In** : IEEE Transactions on Software Engineering, 32(11):910-927. 2006.

[Zhang *et al.*, 2008] Zhang, W and Hansen, KM. Towards Self-managed Pervasive Middleware using OWL/SWRL ontologies. **In** : Fifth International Workshop Modeling and Reasoning in Context (MRC 2008), Held at HCP 08. Delft, The Netherlands, 9-12 June 2008.

[Zheng *et al*., 2006] Zheng, Y., Chan, ATS. and Ngai, G. Applying Coordination for Service Adaptation in Mobile Computing. **In** : IEEE Computer Society 1089-7801, September-October 2006.

[Zheng *et al*., 2007] Zheng, D., Wang, J., Jia, Y., Han,W. Deployment of Context-Aware Component-Based Applications Based on Middleware. **In** : Proc of. Ubiquitous Intelligence and Computing, pp. 908-918, 2007.

[Zobel *et al.*, 2006] Zobel, J and Moffat, A. Inverted files for text search engines. **In** : ACM Computing Surveys, Vol. 38, No. 2, 2006.

[Zittrain *et al.,* 2009] Zittrain, Jonathan, The Future of the Internet - And How to Stop It. Jonathan Zittrain, THE FUTURE OF THE INTERNET - AND HOW TO STOP IT, Yale University Press, Penguin UK/Allen Lane; Oxford Legal Studies Research Paper No. 36/2008. Disponible sur SSRN: http://ssrn.com/abstract=1125949

Mobilité et sensibilité au contexte pour la gestion de documments multimédias personnels : CoMMediA

RESUME : La sensibilité au contexte est une propriété qui identifie les systèmes dont le comportement repose sur l'exploitation d'un ensemble d'informations, appelé *contexte d'utilisation*. Malgré leur hétérogénéité, l'évolution rapide et soutenue des dispositifs mobiles (DM) vers des terminaux multifonctionnels, multi-capteurs et connectables sur le Web, a transformé ces DM en des environnements d'accueil idéaux pour l'exécution d'applications sensibles au contexte. Les DM deviennent également, par leur popularité, les principaux outils de création de documents multimédias personnels (DMP). Dans ce nouveau cadre d'usage, l'information qualifiée de contexte joue un rôle déterminant. En effet, le contexte capté lors de la création d'un DMP peut offrir des informations, appelées métadonnées, suffisamment riches pour le décrire, et, par la suite, l'identifier. Ce travail de thèse se situe ainsi à l'intersection de deux thématiques de recherche : l'Informatique Sensible au Contexte et la Gestion de Documents Multimédias. La prise en compte du contexte de création dans la gestion de DMP est une thématique nouvelle pour laquelle l'adoption d'un modèle sémantique de représentation pour les informations contextuelles, et l'élaboration de mécanismes d'acquisition et d'exploitation de ces informations, sont encore des problèmes non résolus. Nous proposons une architecture sensible au contexte, appelée CoMMediA (Context-aware Mobile Multimedia Architecture), pour la gestion de DMP, inspirée des architectures d'adaptation de Systèmes d'Information (SI) mobiles. Notre contribution s'articule autour : *i)* d'un modèle sémantique de représentation des métadonnées (ContextMultimedia), décrit en OWL et reposant sur une nouvelle notion de contexte, *ii)* d'un intergiciel adaptatif de capture du contexte de création de DMP (appelé DevAC), qui prend en compte l'hétérogénéité et les limitations intrinsèques des DM lors du déploiement d'applications mobiles, et *iii)* d'une plate-forme d'enrichissement de métadonnées contextuelles (ContextAnnotator). Notre objectif est de promouvoir la découverte automatique de métadonnées et d'illustrer également l'exploitation de la richesse sémantique des métadonnées pour l'organisation, l'annotation, le partage et la recherche de DMP. De plus, DevAC et nos applications mobiles de gestion de DMP s'appuient sur une infrastructure à composants, et sur une description sémantique des services offerts par une application. Le but ultime est d'aider le concepteur à développer et à déployer des applications multimédias et sensibles au contexte qui puissent s'adapter aux modèles des dispositifs mobiles des utilisateurs, sans qu'il soit besoin de réécrire le code d'une même application.

MOTS-CLES : Multimédia, Sensibilité au Contexte, Annotation, Web Sémantique, Déploiement Adaptatif.

ABSTRACT : Context-awareness corresponds to a feature computing systems have when they can adapt their overall behavior to a set of information, called the *context of use*. In spite of their heterogeneity, the continuous evolution of mobile devices (MD) towards multipurpose, multi-sensors, and Web-enabled devices, transforms them into an ideal environment for context-aware applications. More and more, MDs are becoming the main tool for creating personal multimedia documents (PMD). In this new scenario, high-level context information can also play an important role. In fact, information related to the context in which the document was captured or created can offer information rich enough to describe, and, thereafter, identify such documents. This work addresses the intersection of two research domains: Context-aware Computing and Multimedia Management. Context-awareness in the management of PMD is a new research approach, where the proposition of semantic models for representing contextual information and of mechanisms for acquiring and exploiting this high-level information are still unsolved problems. We propose a new context-aware architecture for the management of PMD called CoMMediA (Context-aware Mobile Multimedia Architecture), which is inspired by frameworks oriented to the adaptation of Mobile Information Systems. Our contributions consists of: I) a semantic model for representing multimedia metadata (Context Multimedia), described in OWL and following a new *notion of context*, II) an adaptive middleware for context acquisition and multimedia creation (called DevAC), which takes into account the heterogeneity and the intrinsic limitations of MDs during the deployment of mobile applications, and III) a platform that enriches contextual metadata (ContextAnnotator). Our objective is to promote the automatic production of metadata, and also to propose mechanisms for integrating this high-level and semantic metadata in the processes of organization, annotation, sharing, and keyword-based retrieval of DMP. Moreover, DevAC and our mobile applications of DMP management are based on a component middleware and a semantic description of services. The ultimate goal is to assist developers in the development and deployment of multimedia and context-aware applications which can be adapted according to the models of users' MDs, without the need to rewrite their code.

KEYWORDS : Multimedia Management, Context-Awareness, Annotation, Semantic Web, Adaptive Deployment.

Windson Viana de Carvalho UJF