MASTER EN SCIENCES INFORMATIQUES

Systèmes sensibles au contexte
Adaptation et modélisation du contexte

Ntumbabu Kambala, Yves

Award date:
2016

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2015–2016

Systèmes sensibles au contexte : Adaptation et modélisation du contexte

Yves Ntumbabu Kambala

Maître de stage : Prof.Bruno Dumas

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Prof.Bruno Dumas

Co-promoteur : Prof.Kim Mens

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Systèmes sensibles au contexte : Adaptation et modélisation du contexte

Yves Ntumbabu Kambala

Résumé

Avec l'avènement de l'informatique ubiquitaire, l'évolution vers une informatique plus mobile ainsi que le développement des réseaux de communication ont transformé les habitudes de l'utilisateur. Dans l'exécution de ses tâches, l'utilisateur est de plus en plus mobile et passe facilement d'un ordinateur à un téléphone portable, d'un environnement à un autre. On dit alors qu'il change souvent de contexte. Cela a entraîné l'émergence des applications dites sensibles au contexte dont le but principal est de s'adapter en fonction de ces changements réguliers de contexte. Ce travail se focalise sur ces systèmes sensibles au contexte, et plus précisément sur la gestion du contexte et les mécanismes d'adaptation au contexte.

Après une présentation des concepts de base et d'une revue des différentes approches existantes, nous nous intéressons au développement d'une application sensible au contexte en mettant l'accent sur la modélisation du contexte à l'aide d'une ontologie et des données extraites du profil Facebook de l'utilisateur.

Mots-clés : modélisation du contexte, sensibilité au contexte, adaptation, adaptation de contenu, adaptation d'interfaces utilisateurs, application dépendante du contexte, modèle ontologique, Facebook, réseaux sociaux.

Abstract

With the advent of the ubiquitous computing, mobile computing and networks communication transformed user habits. In performing its tasks, the user is more mobile, and can easily switch from a computer to a smartphone, from a place to another. We can then say that the user changes the context. That situation led to the emergence of applications called "context-aware" whose main goal is to adapt their behaviour according to the context changes.

After a presentation of basic concepts and a review of various existing approaches, we focus on context modelling using an ontology and data from a Facebook account.

keywords : context modeling, context awareness, adaptation, content adaptation, user interfaces adaptation, Context-Aware Application, Ontology modeling, Facebook, social networks.

Remerciement

Avant toute chose, je tiens à remercier l'ensemble du personnel de la faculté d'informatique de l'université de Namur pour la formation reçue pendant mes trois années passées ici, mais également pour leur accueil, leur sympathie et leur disponibilité tout au long de mon stage.

Je souhaite remercier tout particulièrement mes promoteurs, le professeur Bruno Dumas de l'université de Namur et le professeur Kim Mens de l'université de Louvain, pour les encadrements, les explications et la confiance qu'ils m'ont accordée tout au long de ce travail.

J'adresse également un grand merci à ma famille et toutes les personnes qui m'ont aidé et soutenu d'une manière ou d'une autre durant mon parcours universitaire.

Table des matières

Table des figures

Liste des tableaux

Glossaire

API Application Programming Interface. Désigne un ensemble de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.

API Graph API offerte par Facebook qui permet d'accéder au graphe social.

ASCC/Mark Appelé aussi MArk I, il est l'un des premiers calculateurs universels construits aux États-Unis.

Facebook réseau social en ligne qui permet à ses utilisateurs de publier des images, des photos, des vidéos, des fichiers et documents, d'échanger des messages, joindre et créer des groupes et d'utiliser une variété d'applications.

Graphe social Dans le contexte Internet, le graphe social est une représentation des utilisateurs et leurs relations. Dans le contexte du réseau social Facebook, le graphe social reprend les relations (amitié, «like», etc.) entre un utilisateur et des objets (autres utilisateurs, groupes, événements, pages, etc.).

IDE Integrated Development Environment. Logiciel composé d'un ensemble d'outils qui aide le programmeur dans le développement logiciel.

J2EE Java 2Platform Enterprise Edition. Plateforme portable, multi-utilisateurs, sécurisée et normalisée pour des déploiements de services écrits en langage Java. Elle fournit une spécification des règles à respecter lors de l'écriture de logiciels client-serveur.

JENA Framework Java pour le développement des applications du Web sémantique.

Librairie Regroupement de code d'une ou plusieurs fonctions écrites par un autre programmeur ou un groupe de programmeur.

Protégé Logiciel de création d'ontologies.

Réseau social Ensemble d'individus reliés entre eux par des liens sociaux. Dans ce travail, ce mot renvoie à sa forme numérique sur le Web.

UIDL Langage de description d'interface graphique.

UML Formalisme pour spécifier, construire, visualiser et décrire les artefacts d'un système logiciel.

URL Uniform Resource Locator. Adresse globale d'un objet (un document ou d'autres ressources), typiquement une page Web, sur l'Internet, intégrant le protocole d'accès à l'objet.

UWP Plateforme de développement créé par Microsoft, qui permet développer des applications universelles. Une application universelle est une application qui tourne à la fois sur Windows 10 et Windows 10 mobile.

Acronymes

ACM Association for Computing Machinery

CMF Context Management Framework

ENIAC Electronic Numerical Integrator Analyser and Computer

GPS Global Positioning System

IEEE Institute of Electrical and Electronics Engineers

IHM Interface Homme-Machine

IoT Internet of things

OS Operating system

PC Personal Computer

PDA Personal Digital Assistant

RFID radio frequency identification

UML Unified Modelling Language

UWP Universal Windows Platform

W3C World Wide Web Consortium

XML Extensible Markup Language

Chapitre 1

Introduction

1.1 Contexte général

Ces vingt dernières années, les ordinateurs ont connu une avancée fulgurante. On est passé des gros calculateurs (ASCC/Mark, ENIAC, ...) à des ordinateurs plus petits et omniprésents dans notre environnement (laptop, smartphone, tablette, etc.). Cette évolution vers la miniaturisation couplée à l'évolution des réseaux informatiques, apporte un bon nombre de changements dans le paysage informatique :

— La multiplication des ordinateurs (systèmes omniprésents, systèmes multi-utilisateurs, smartphones, systèmes embarqués, etc.) ;
— La multiplication des objets connectés (grands réseaux, IoT, etc.) ;
— La multiplication des services ;
— Le changement dans l'interaction entre l'homme et la machine.

On parle alors de la troisième ère de l'informatique, l'informatique ubiquitaire, dans laquelle l'ordinateur est omniprésent dans l'environnement et impacte la vie au quotidien de l'utilisateur. On passe de l'ordinateur fixe au portable (laptop, smartphone, tablette, systèmes embarqués...), des systèmes centralisés à des systèmes décentralisés, de l'utilisateur fixe à un utilisateur plus mobile qui change régulièrement d'environnement. Ces changements apportés par l'informatique ubiquitaire ont un impact direct dans la conception et le développement logiciel, ils posent de nombreux défis en ingénierie logicielle ainsi qu'en interac-

tion homme-machine. Dans le cycle de développement, les concepteurs devront prendre en compte les différents contextes d'usage possibles. Il peut s'agir de différents types de plates-formes et leurs spécificités, de différents types d'utilisateurs, ou encore de différents environnements dans lesquels le système peut se trouver. Une nouvelle génération d'applications a donc vu le jour comme réponse à cette évolution. Il s'agit des applications orientées contexte ou sensibles au contexte (Context aware application). Ces applications offrent des services mieux adaptés à la situation de l' utilisateur, et tiennent compte de certaines caractéristiques contextuelles telles que ses préférences personnelles, son environnement physique, son emplacement géographique, ses déplacements, etc. Dans ces applications, le processus d'adaptation s'exécute à différents niveaux (Figure 1.1.1) :

— Au niveau comportemental ou noyau fonctionnel : modification des fonctionnalités du logiciel et adaptation des services en fonction du contexte actuel. Cette notion concerne l'ingénierie du logiciel.

— Au niveau présentation ou interface utilisateur : modification de l'interface utilisateur en fonction du contexte actuel. Cette notion concerne l'interaction homme-machine.

— Éventuellement, au niveau de la base de données afin de maintenir la cohérence du système.

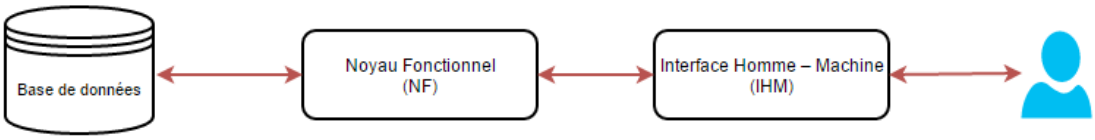


FIGURE 1.1.1 – Vue simplifiée de l'interaction entre les composants d'un logiciel

Pour illustrer notre propos, prenons le cas de l'application Contact sur Windows 10. Cette application permet à un utilisateur de gérer sa liste de contacts.

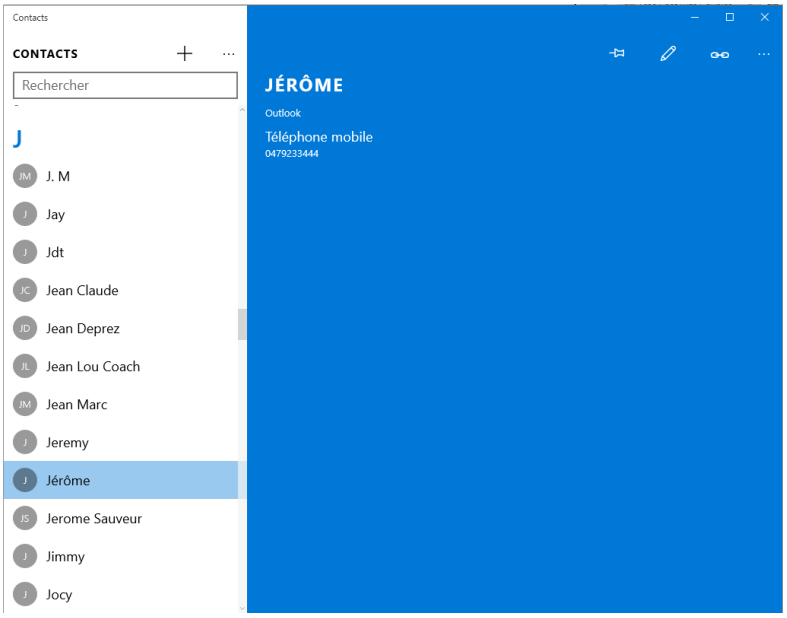


FIGURE 1.1.2 – Application Contact sur Windows 10

Sur la version PC (Figure 1.1.2), étant donné que l'application a accès à un écran plus large, elle affiche sur un même écran la liste des contacts ainsi que les détails des contacts. Ce qui n'est pas le cas avec la version mobile (Figure 1.1.3) dans laquelle, la liste des contacts et les détails se retrouvent sur deux écrans distincts. L'application adapte donc son interface utilisateur (présentation) en fonction de la taille de l'écran. Sur l'écran des détails de la version mobile de l'application, on remarque que l'application possède des fonctionnalités supplémentaires telles que lancer un appel téléphonique ou envoyer un message au contact. Ce qui n'est pas le cas dans la version PC. L'application adapte ses fonctionnalités en fonction de la plate-forme et des ressources dont elle dispose. Il y a adaptation du noyau fonctionnel.

1.2 Buts

L'objectif poursuivi dans ce travail est de comprendre ce qu'est un système sensible au contexte ainsi que les concepts liés au développement de ces applications :

- C'est quoi le contexte ?
- Comment modéliser le contexte ?
- Comment utiliser le contexte ? Comment s'adapter au contexte ?

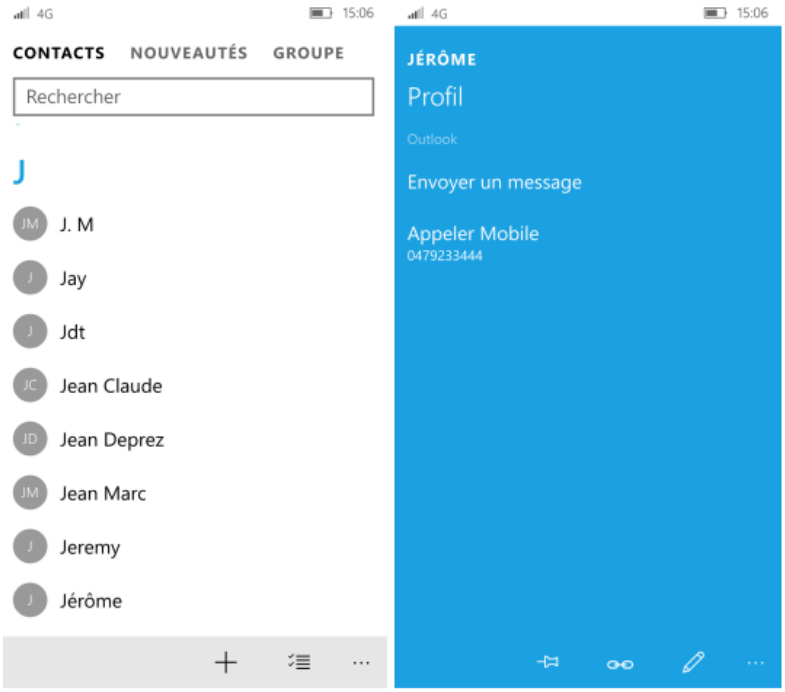


FIGURE 1.1.3 – Application Contact sur Windows phone

1.3 Méthodes de travail

Le développement des applications sensibles au contexte est un processus assez complexe qui fait intervenir plusieurs concepts tant en interaction homme-machine qu'en ingénierie du logiciel.

Il est donc logique de commencer par une première phase de recherche et de documentation afin de mieux cerner le sujet, de comprendre les concepts clés et de parcourir les différents travaux existants. Pour cela, plusieurs recherches (recherches thématiques et recherches par arbres de citations) ont été effectuées grâce à des outils numériques tels que :

— Google Scholar ;

— ACM ;

— IEEE ;

— etc.

Cette phase est assurément la plus importante du travail, car elle a permis de poser les fon-

dements et de réunir dans un état de l'art, plusieurs approches de gestion du contexte sur lesquelles nous nous sommes basés par la suite.

Une fois cette phase de recherche terminée, un prototype a été développé afin de mettre en pratique les concepts théoriques étudiés. Pour mettre en place ce prototype, nous sommes passés par les étapes suivantes :

— Conception de l'architecture de l'application en s'inspirant des travaux existants ;
— Étude de la structure du réseau social en ligne Facebook : ses objets et les relations entre eux ;
— Conception d' une ontologie qui représente les éléments de Facebook et les relations entre ces éléments ;
— Modification de l'ontologie en supprimant les éléments non pertinents pour le modèle du contexte
— Implémentation du modèle du contexte avec Protégé comme outil d'édition des ontologies ; définition des règles de raisonnement pour l' inférence du contexte ;
— Intégration du modèle du contexte avec d'autres éléments du contexte physique tels que l'heure, la météo et la localisation ;
— Développement d'un client qui fait appel au serveur de contexte et qui s'adapte en fonction du contexte actuel

1.4 Organisation du mémoire

Afin d'atteindre les objectifs fixés, nous avons réalisé un état de l'art qui reprend les concepts de base des systèmes sensibles au contexte. Pour des raisons de clarté, ces concepts clés sont repartis dans des chapitres différents.

Le chapitre 1 sert d'introduction au sujet.

Le chapitre 2 est consacré à la notion de contexte. Cette notion est très souvent utilisée, mais il est difficile de trouver une définition précise de ce qu'est le contexte. Ainsi dans ce chapitre, nous essayons de parcourir les différentes définitions que l'on retrouve dans la littérature, les méthodes de captures du contexte, ainsi que les différentes techniques et approches de modélisation du contexte.

Le chapitre 3 se focalise sur les différentes architectures existantes des systèmes sensibles au contexte.

Dans le chapitre 4, nous abordons la notion d'adaptation. Nous essayons de détailler ce qu'on entend par adaptation au contexte ainsi que les différents mécanismes d'adaptation.

Dans le chapitre 5, nous mettons en place un prototype en nous inspirant des approches existantes parcourues dans l'état de l'art. Ce chapitre présente l'application *Mon Guide*, une application qui adapte son contenu par rapport au profil de l'utilisateur et à certaines informations qui l'entourent. Ce chapitre présente de manière détaillée :

— L'architecture générale de l'application ;

— L'approche de modélisation du contexte choisie et les raisons de ce choix ;

— Les différentes adaptations que réalise l'application.

Chapitre 2

Le contexte

2.1 Introduction

Dans cette section, nous allons introduire la notion de contexte. Dans un premier temps, nous présentons les différentes définitions existantes sur cette notion de contexte, ainsi que les principes de la sensibilité au contexte. On parle généralement de « context-awareness ». Par la suite, nous allons parcourir les différentes approches de modélisation de contexte et présenter une architecture générique pour la gestion du contexte.

2.2 Définitions du contexte

2.2.1 Définition générale

Il n'est pas aisé de donner une définition exacte de ce qu'est le contexte. Nous allons essayer de parcourir les différentes définitions données dans certains dictionnaires français et essayer d'en tirer un sens commun :

- **Le Larousse :**
  (a) « Ensemble des conditions naturelles, sociales, culturelles dans lesquelles se situe un énoncé, un discours.»

(b) « Ensemble des circonstances dans lesquelles se produit un événement, se situe une action : replacer un fait dans son contexte historique.»

(c) « Ensemble du texte à l'intérieur duquel se situe un élément d'un énoncé et dont il tire sa signification.»

(d) « Ensemble d'informations caractérisant l'état de l'unité centrale d'un ordinateur à tout moment de l'exécution d'un programme.»

(e) « Ensemble des éléments (phonème, morphème, phrase, etc.) qui précèdent et/ou suivent une unité linguistique à l'intérieur d'un énoncé.»

- **Le Petit Robert :**
  (a) « Ensemble des circonstances dans lesquelles s'insère un fait».

- **Le-dictionnaire.com :**
  (a) « Ensemble du texte situé autour d'une phrase ou d'un mot, dont ce mot ou cette phrase tire sa signification précise.»

  (b) « Ensemble des circonstances entourant un événement.»

Les définitions données ci-haut présentent toutes des points communs. Premièrement, l'idée d'un élément central, un élément sujet. C'est l'élément dont on veut étudier le contexte. Il peut s'agir d'un mot, une phrase, un énoncé, un discours, un ordinateur, une application. Nous retrouvons également l'idée d'un ensemble d'éléments qui entoure le sujet et qui possède certaines informations. Ces informations sont utiles pour la compréhension, l'interprétation, le fonctionnement du sujet. Ces informations constituent le contexte.

2.2.2 La notion de contexte en informatique ubiquitaire

Comme souligné dans les définitions de la section précédente, la notion de contexte permet d'améliorer l'interprétation ou la compréhension d'un terme, d'une action, d'un fait... et par conséquent, améliore grandement la communication entre deux personnes ou entre un homme et un ordinateur (interaction homme-machine). Le «contexte» a donc une importance capitale en interaction homme-machine et en informatique de manière générale. C'est pourquoi depuis plusieurs années, un grand nombre de chercheurs s'intéresse à cette notion de «contexte». Le «contexte» a été modélisé et exploité dans de nombreux domaines de l'informatique principalement en informatique ubiquitaire. Sa définition et son utilisation ont

été débattues au sein de la communauté scientifique pendant de nombreuses années sans parvenir à un consensus clair [25]. Dans [68], la notion de contexte a été introduite en l'associant à 1'informatique ubiquitaire : «l'ensemble des informations à prendre en compte en vue d'une adaptation».

Le terme «context aware» a été introduit pour la première fois par Schilit [57] pour désigner un système doté d'un modèle du contexte. Selon Schilit, le contexte consiste en la localisation, les personnes et les objets à proximité ainsi que les différentes modifications. Étudier le contexte revient donc à répondre à ces trois questions :

— « Où es-tu ? »,

— « Avec qui es-tu ? »,

— « De quelles ressources disposes-tu à proximité ? ».

Partant de cette approche, la notion de contexte désigne tout changement dans l'environnement physique, le comportement des utilisateurs et les ressources de la plate-forme. Ces idées sont reprises dans les travaux [51] et [23]. Un peu plus tard, Brown [11] propose de restreindre la notion de contexte aux éléments de l'environnement de l'utilisateur. Par la suite, il propose de prendre en compte également les informations sur l'heure, la saison, la température, l'identité et la localisation de l'utilisateur[55].
Andy Ward [67] définit le contexte comme étant l'ensemble des états de tous les environnements possibles d'un système.

En 1998, Pascoe [51] donne une définition assez similaire à celle de Ward en reprenant l'idée d'ensemble d'états. Il définit le contexte comme étant un sous-ensemble d'états conceptuels ou physiques ayant un intérêt pour une entité particulière. Un an plus tard, Dey apporte quelques précisions : « le contexte couvre toutes les informations pouvant être utilisées pour caractériser la situation d'une entité. Une entité est une personne, un lieu, ou un objet qui peut être pertinent pour l'interaction entre l'utilisateur et l'application, y compris ces deux derniers »[23]. Cette définition étant un peu plus générique, elle couvre pratiquement tous les travaux antérieurs.

Dans leur étude sur la plasticité des interfaces utilisateurs, Thevenin et coll. [65] aboutissent à une définition assez proche de la notion de contexte d'interaction. Le contexte peut être vu comme le triplet **<utilisateur, plate-forme, environnement>** où :

- L'utilisateur désigne la personne qui utilise le système interactif. Il peut être décrit par ses compétences dans le domaine applicatif, sa familiarité avec les systèmes interactifs,

ses capacités physiques et mentales, sa langue, sa culture, etc.

- La plate-forme désigne les requis matériels et logiciels nécessaires à l'interaction. Typiquement, la surface d'affichage, les différentes modalités disponibles, les dispositifs d'interaction, les capacités de calcul et de communication sont à considérer également. Tous ces éléments sont susceptibles d'influencer l'interaction.
- L'environnement est l'ensemble des entités (objets, personnes et événements) périphériques à la tâche courante et pouvant avoir un impact sur le comportement du système ou son interaction avec l'utilisateur. Il peut être décrit par ses conditions lumineuses, sonores, sociales, etc.

C'est sur cette définition de Thevenin et coll. [65] que nous nous baserons dans la suite du travail.

2.3 Les informations du contexte

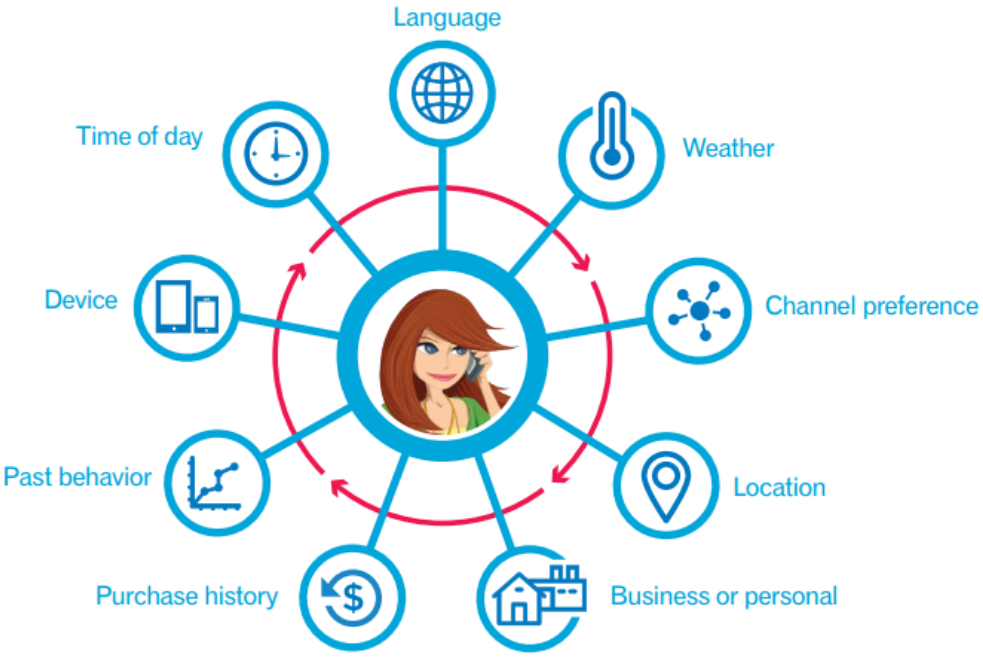


FIGURE 2.3.1 – Les informations du contexte

Comme nous l'avons remarqué, le contexte est un élément important, qui contient beaucoup d'informations utiles pour le fonctionnement d'un système. Très longtemps associé à la localisation, il contient beaucoup d'autres informations notamment sur le temps, l'emplacement, l'activité de l'utilisateur, etc. [72]. Pour avoir une vue globale des différentes informations que peut contenir le contexte, nous allons les catégoriser en nous basant sur les trois axes de la définition de Thévenin [65].

— L'environnement

  - La localisation : Elle ne se limite pas uniquement à la position absolue d'une entité, mais également la position relative par rapport à d'autres entités (proximité, distance par rapport à, etc.), l'orientation, l'altitude ;
  - L'information temporelle : donne les informations sur le temps absolu, le temps relatif, le moment de la journée (avant-midi, après-midi, soirée, nuit), jour ouvrable ou weekend, la saison ... Cette information permet également d'établir un historique permettant d'enrichir le contexte. En effet, l'enchaînement et l'ordonnancement d'actions ou d'événements dans le temps peuvent avoir un impact sur la décision prise par l'application.
  - L'information sur l'environnement physique : donne les informations sur les entités périphériques au système, la luminosité, le son, la température, la pression, la pollution ...

— La plate-forme

  - Le type : il permet de préciser s'il s'agit d'un ordinateur (Windows, Linux, Mac OS, etc.), un téléphone portable (Android, IOS, WindowsPhone, etc.), une tablette (iPad surface, Android, etc.) ...
  - Les ressources : donne les différentes ressources disponibles sur la plate-forme ainsi que ses capacités. Par exemple la taille de l'écran, présence d'une caméra, présence de capteurs, etc.
  - L'autonomie : donne des informations sur le niveau de batterie, l'alimentation
  - La connectivité : donne des informations sur les capacités du système à se connecter à un réseau. Ex : Wifi, 4G, 3G, Bande passante...

— L'utilisateur :

  - Activité : donne des informations sur ce que fait l'utilisateur actuellement ainsi que ce qu'il compte faire juste après.

http ://www.experian.fr/blogs/business-strategies/2015/09/marketing-contextuel-5-etapes-pour-bien-debuter

- Profile d'utilisateur : englobe généralement tout ce qui concerne les préférences, les intérêts, les habitudes d'un utilisateur, modalités...

Ces informations peuvent être utilisées de différentes manières, elles peuvent être soit directement interprétées, soit combinées avec d'autres données afin d'obtenir des informations supplémentaires sur le contexte. Un système qui permet de détecter les informations du contexte et de s'adapter en fonction est qualifié de sensible au contexte.

2.4 Approche générique pour la gestion du contexte

La conception des applications sensibles au contexte n'est pas une tâche facile, car les méthodes classiques de développement logiciel sont difficilement applicables dans le développement des applications sensibles au contexte. La difficulté réside essentiellement dans la gestion et l'utilisation du contexte. C'est pourquoi d'après [23], il est important de bien séparer le processus d'acquisition du contexte de son utilisation au sein de l'application, lorsque l'on veut développer une plate-forme générique de développement et déploiement d'applications sensibles au contexte. Dans leurs travaux, Dey et coll. [23] définissent quelques abstractions. Ces abstractions seront reprises par plusieurs chercheurs proposant ainsi diverses architectures à plusieurs couches, ce qui facilite une interprétation de haut niveau du contexte ainsi qu'une nette séparation entre la gestion du contexte et l'application. Ces architectures diffèrent dans la complexité, les noms ainsi que la disposition de leurs couches, mais se basent toutes sur une architecture conceptuelle générale (figure 2.4.1) qui sépare la capture du contexte et son utilisation en ajoutant entre elles, des couches d'interprétation et d'inférence. Cette architecture présente les cinq couches suivantes : capture du contexte, interprétation/Agrégation du contexte, stockage/historique du contexte, dissémination du contexte et application [17].

2.4.1 Capture du contexte

La première couche est la couche d'acquisition du contexte. Il est important de préciser qu'il existe deux types de contexte [70] :

- Le contexte physique qui désigne des informations qui peuvent être capturées par des senseurs électroniques. Nous pouvons citer à titre d'exemple la température, la localisation, la pression de l'air, la lumière, le mouvement, le son, le toucher... Ces informations sont généralement utilisées dans diverses applications et sont assez faciles à

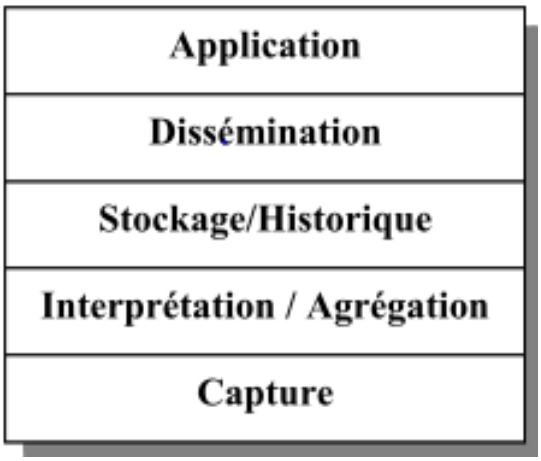


FIGURE 2.4.1 – Les couches abstraites de l'approche générique

obtenir grâce aux différents capteurs existants que nous présenterons dans la suite de ce travail.

- Le contexte virtuel qui désigne des informations spécifiées par les utilisateurs ou capturées à partir des interactions de l'utilisateur, y compris les préférences de l'utilisateur, les buts et les tâches. Ces informations sont obtenues grâce à des programmes informatiques appelés parfois «capteurs logiques» [35].

Il est donc important de signaler que le mot « capteur» ici ne désigne pas uniquement un dispositif électronique, mais également tout dispositif physique ou logiciel qui peut aider à obtenir des informations du contexte. En fonction du type de contexte, les capteurs peuvent être classés en trois groupes [35].

1. **Capteurs physiques**

   Le capteur physique est probablement le type de capteur le plus connu et le plus utilisé. Il s'agit d'un dispositif électronique qui permet de mesurer une grandeur physique et qui la transforme en signal électronique. De nos jours, il existe un très grand nombre de capteurs physiques qui permettent de capturer diverses informations du contexte. Le tableau 2.4.1 donne une liste non exhaustive de capteurs physiques existants.

2. **Capteurs virtuels**

   Contrairement aux capteurs physiques, les capteurs virtuels sont immatériels. Basés sur des composants logiciels, ils fournissent des informations du contexte à partir d'applications ou services logiciels. Par exemple, il est possible de déterminer la localisation

Capteurs physiques	Contexte capturé
Capteur audio	Bruit, niveau de décibels, musique
Caméra vidéo	Émotion, la présence, le comportement
Détecteur de mouvement	Présence, nombre d'utilisateur
Capteur de pression	Pressé, occupé, mouvement de la main
Capteur de lumière	Lumière ambiante, luminosité
Accéléromètre	Mouvement, état physique, vibration
Bluetooth	Localisation
Infrarouge	Localisation
RFID	Localisation, activité, situation
GPS	Localisation
Environmental sensors	Météo, Température, humidité
Événement	Événement, notification, etc.

TABLE 2.4.1 – Exemples de capteurs physiques

d'un individu sans utiliser un capteur physique (GPS), mais en utilisant un capteur virtuel qui déterminera la position approximative de l'individu en utilisant les données de son agenda électronique, ses déplacements (voyage, bus, train,), e-mails, etc. Il est également possible de détecter l'activité de l'utilisateur sur un ordinateur en analysant les événements de la souris ou les saisies à partir du clavier [7].

3. **Capteurs logiques**

   Il y a certaines informations que ni un capteur physique ni un capteur virtuel n'est en mesure de les fournir. C'est là que les capteurs logiques entrent en scène. Ils permettent d'inférer des informations du contexte en combinant les données issues de plusieurs sources (capteur physique ou virtuel). Par exemple, la Kinect de Microsoft [71] utilise un capteur logique pour la reconnaissance de mouvement. Ce capteur combine les données d'une caméra couleur RGB et d'un capteur de profondeur afin de récupérer des données plus précises sur le mouvement de l'utilisateur.

Afin de faciliter l'accès aux données capturées, chaque capteur est généralement accompagné de pilotes logiciels et d'une API. Ces composants jouent le rôle d'interface de communication entre le fournisseur de données (un capteur) et le consommateur de données (une application ou un service).

2.4.2 Couche d'interprétation et d'agrégation du contexte

Les données récoltées par la couche de capture sont généralement qualifiées de données brutes c'est-à-dire que ces données ne sont qu'une collection de valeurs qui n'ont pas spécialement beaucoup de sens. Par exemple, un capteur GPS nous donne des coordonnées GPS d'un lieu. Ces coordonnées sont très techniques et peu parlantes pour un être humain. On préfère généralement recevoir plutôt comme donnée une adresse complète composée d'un nom de rue, un numéro, une ville et un pays. D'où l'importance d'une couche d'interprétation qui offre des mécanismes de traitement et d'interprétation des données brutes issues des capteurs afin d'en tirer des informations du contexte de plus haut niveau. Ces informations sont généralement plus faciles à comprendre et à utiliser. Lors du traitement dans cette couche, les données sont susceptibles de subir un ensemble de transformations :

- L'extraction : le système récupère l'information pertinente et la présente sous une autre forme. Ex. : Le système récupère les coordonnées GPS et les transforme en une adresse (avenue, numéro, ville, pays).
- La quantification : le système récupère un ensemble de données issues d'un même capteur afin d'en donner du sens. Prenons par l'exemple le cas de l'accéléromètre. Le système n'a pas besoin uniquement d'une valeur atomique, mais d'une suite de valeur afin de détecter le mouvement de l'objet, comme dans le cas d'une chute par exemple.
- L'agrégation : le système combine deux ou plusieurs données issues de capteurs différents afin d'augmenter la précision ou de produire une nouvelle information. En effet, certaines valeurs fournies par des capteurs n'ont pas beaucoup d'intérêt lorsqu'elles sont isolées. Par exemple si l'on veut détecter la présence d'une personne, on peut soit utiliser un capteur infrarouge soit un capteur ultrason. Le problème est que les détecteurs de présence à infrarouges, étant passifs, risquent de ne pas détecter les mouvements légers. Par contre, des détecteurs à ultrasons étant trop sensibles peuvent détecter des présences non humaines tels que le passage d'une mouche, chose que l'on ne souhaite clairement pas. Pour éviter cet inconvénient tout en gardant une sensibilité importante, on couple les capteurs infrarouges avec des capteurs ultrasons. Cette combinaison permet d'augmenter la fiabilité des détecteurs et par conséquent d'éliminer les effets indésirables. On peut aussi coupler les données concernant le nombre de personnes présentes dans une pièce avec le niveau de bruit dans la pièce. Ainsi on pourra détecter s'il y a une réunion dans cette pièce ou pas.

Étant données les diverses sources d'informations à disposition, il peut y avoir différents conflits. Deux sources différentes peuvent communiquer des informations contradictoires

ou imprécises. La couche devra s'assurer de résoudre les conflits et de maintenir une certaine cohérence [22] [1].

2.4.3 Couche de stockage et historique du contexte

Une fois l'information sur le contexte capturée grâce aux différents capteurs et interprétée grâce aux différents mécanismes de la couche d'interprétation, il faut bien la stocker quelque part. C'est là que la troisième couche intervient. Elle permet de garder une trace des différents contextes et de créer un historique de contexte. Le stockage peut s'organiser dans un fichier ou dans une base de données, de manière centralisée ou distribuée. Le fait de garder une trace des différents contextes peut s'avérer important et peut servir dans beaucoup de cas. Prenons comme exemple : un utilisateur italien qui se rend en vacances aux États-Unis. Le système pourra alors se baser sur l'historique pour détecter d'où vient l'utilisateur, dans ce cas-ci l'Italie, et pourra lui proposer les restaurants italiens de la région, une conversion de prix dollar-euro, etc.

Lorsque l'on souhaite stocker des données, il faut les stocker de manière à les rendre facilement réutilisables par une application ou un service. C'est pourquoi il est impératif de choisir un bon modèle ou format pour ces données et s'assurer que toutes les données respectent les contraintes liées à ce format. Ainsi, pour le stockage du contexte, un modèle est également requis. Le but est d'obtenir un modèle de contexte qui permettra de stocker, d'interpréter et de raisonner sur les données, ce qui faciliterait notamment le partage d'informations du contexte entre les différentes couches et également avec les différentes applications.

Il existe plusieurs approches pour modéliser le contexte. Les principales techniques de modélisation du contexte ont été synthétisées dans les travaux de Strang et Linnhoff-Popien [60] et ceux de Bettini et coll. [10]. Le but est d'avoir une approche qui est à la fois extensible (possibilité d'ajouter de nouveaux types d'information), évolutive et qui facilite la manipulation des données (facilité de mise à jour), et l'exécution de requêtes.

A) **Modèles clé-valeur**

Le principe général d'une modélisation par clé-valeur est de représenter l'information sous la forme d'une paire (clé, valeur) où :

- La clé est un identifiant unique qui est le nom de l'information du contexte représenté. Par exemple : météo, heure, localisation.

- La valeur est comme son nom l'indique, la valeur assignée à l'information du contexte représentée par la clé. En fonction de la clé et de la granularité, cette valeur peut être de différents types (entier, une chaine de caractère ou un booléen) avec diverses unités. Par exemple, on peut avoir comme information sur la température : (Température = 8) ou (Température = Froid).

Le contexte sera donc représenté par un nuplet de pair (clé, valeur) qui représente donc chaque information du contexte. Par exemple (Nom = «contexte1», Utilisateur ="user1", Localisation ="Université de Namur", Temps ="lundi 6-04-2016 8 h 51 min 20 s", Température = 18°C). Dans cet exemple, en lisant les informations concernant la localisation, le temps et la température, on comprend que le contexte *contexte1* indique que l'utilisateur *user1* se trouve en plein cours. Le grand avantage que possède cette approche est qu'elle est légère et facilement implémentable. Pour exploiter le contexte, il suffit de parcourir les contextes disponibles par le nom. Une fois que l'on a trouvé le contexte recherché, on peut facilement trouver les différentes informations en appliquant un pattern matching sur les différentes clés.

Cette approche supporte bien les opérations d'ajout, lecture, modification et suppression. Cependant, ce modèle manque d'expression et de complétude. En effet, sa structure trop « plate » ne permet pas de définir les structures complexes.

B) **Modèles basés sur XML**

Ces approches se basent sur le langage XML qui est un langage simple de construction, mais qui offre beaucoup de possibilités. XML est basé sur une structure hiérarchique de tag. Un tag possède généralement un ou plusieurs attributs et peut avoir ou non du contenu. Ce contenu peut être du texte ou une autre balise. Ce qui permet de faire des constructions imbriquées sous forme d'arbre. La profondeur dépendra du contexte décrit.

Les langages basés sur les profils sont de manière générale les plus utilisés. Nous pouvons citer à titre d'exemple : CC/PP (Composite Capabilities / Preference Profile) [49] et UAProf (User Agent Profile) [33] qui sont tous les deux standardisé par le W3C. Plusieurs autres langages se basent sur ces deux langages afin d'étendre leurs capacités. À titre d'exemple, nous pouvons citer le CSCP (Comprehensive Structured Context Profiles) [12]. Contrairement à CC/PP, CSCP ne se base pas sur une structure hiérarchique fixe. Il permet une certaine flexibilité qui facilite l'expression des informations du contexte. Comme autres langages, on peut citer également Context Extension [34], ConteXtML [39], etc. La figure **??** montre un exemple CC/PP utilisé pour la modélisation de la localisation de l'utilisateur [34] :

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:cscp="context-aware.org/CSCP/CSCPProfileSyntax#"
    xmlns:dev="context-aware.org/CSCP/DeviceProfileSyntax#"
    xmlns:net="context-aware.org/CSCP/NetworkProfileSyntax#"
    xmlns="context-aware.org/CSCP/SessionProfileSyntax#"

    <SessionProfile rdf:ID="Session">
        <cscp:defaults rdf:resource=
"http://localContext/CSCPProfile/previous#Session"/>
        <device>
            <dev:DeviceProfile>
                <dev:hardware>
                    <dev:Hardware>
                        <dev:memory>9216</dev:memory>
                    </dev:Hardware>
                </dev:hardware>
            </dev:DeviceProfile>
        </device>
    </SessionProfile>
</rdf:RDF>
```

FIGURE 2.4.2 – Exemple code CSCP

C) **Modèles graphiques**

UML (Unified Modeling Langage) est probablement l'un des outils de modélisation les plus utilisés dans la communauté informatique. Il est composé d'un ensemble d'outils de modélisation graphique (Diagramme UML) qui permettent de modéliser différents problèmes. Grâce à sa généricité, UML peut également être utilisé pour modéliser le contexte. Un exemple de modélisation de contexte avec UML est détaillé par Bauer dans sa publication [8]. Une autre approche est d'utiliser une extension de ORM (Object-Role Modeling). Le concept de base de ORM est la notion de « fact ». Modéliser un domaine revient donc à identifier un ensemble de type de « fact » et de « roles ». Dans ses travaux [32], Henricksen propose une extension du modèle ORM de base afin de gérer la modélisation du contexte. Il propose les concepts additionnels suivants (Figure 2.4.3) :

— Le type static fact (reste inchangé en fonction de l'entité représentée) et dynamic fact (profiled, sensed ou derived).

— Le type « history fact » (permet de gérer l'aspect temporel du contexte).

— La notion de dépendance entre deux « fact » : relation dans laquelle un changement dans l'un entraîne automatiquement des changements dans l'autre[32].

D) **Modèles orientés objet**

Le principal avantage de ces approches est l'utilisation des mécanismes de la programmation orientée objet à savoir les notions d'héritage, de polymorphisme, d'encapsulation et de réutilisabilité. Dans ces approches, le contexte est représenté par un objet, les détails du contexte sont encapsulés dans cet objet et l'accès aux informations du contexte est assuré via des interfaces bien définies. À titre d'exemple, on peut citer « CUES »[58] et « Active Object Model » [20]

E) **Modèles basés sur la logique**

« Une logique définit les conditions dans lesquelles une expression de conclusion ou de fait peut être dérivée (processus connu sous le nom raisonnement ou inférence) à partir d'un ensemble d'autres expressions ou des faits » [60]. Dans une approche basée sur la logique, le contexte est donc défini comme un ensemble de faits, d'expressions et de règles.

F) **Modèles basés sur les ontologies**

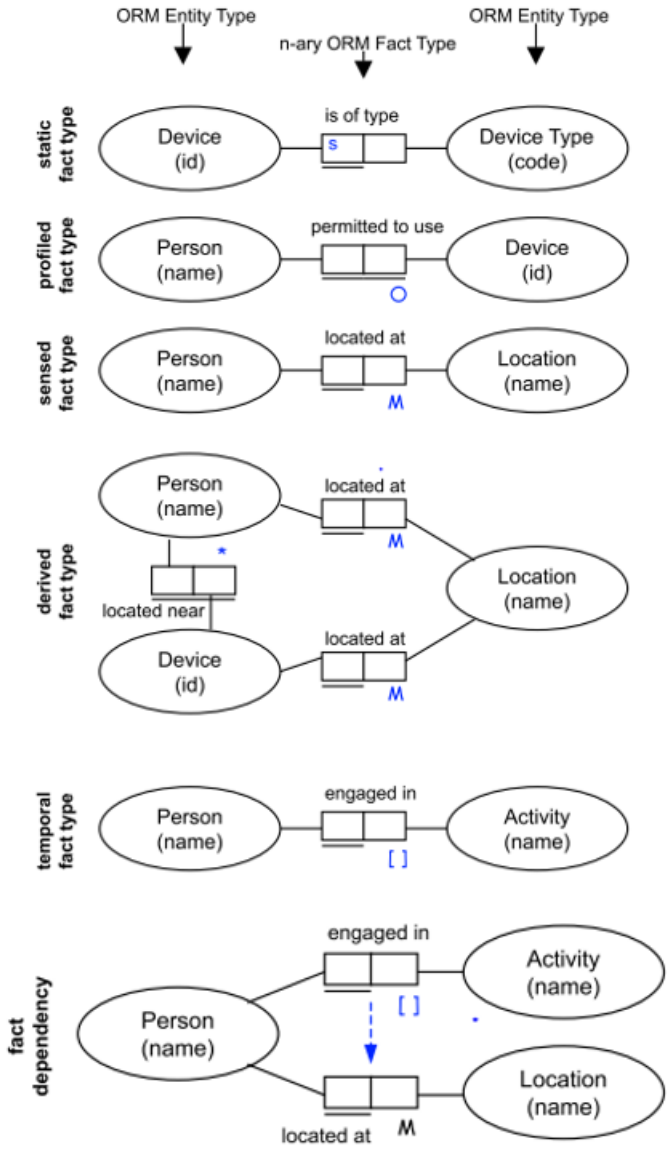
Les ontologies sont généralement utilisées pour représenter des concepts et les relations entre ces concepts. Elles permettent de modéliser des informations ou des concepts de notre quotidien dans un format compréhensible et utilisable par l'ordinateur. De ce fait, les ontologies se présentent comme une solution alternative pour modéliser le contexte [19]. Les différents avantages que présente l'utilisation d'une approche ontologique sont [54] :

— Une ontologie facilite le partage de connaissances dans un système distribué.

— Grâce à la sémantique déclarative, il y a possibilité de construire des raisonnements sur les informations du contexte.

— Avec une représentation explicite d'une ontologie commune, l'interopérabilité des applications et des terminaux est assurée.

[40] présente des exigences pour une bonne modélisation du contexte par avec une approche ontologique :



[60]

FIGURE 2.4.3 – Modélisation graphique du contexte

— Simplicité : les expressions ainsi que les relations utilisées doivent être assez simples afin de simplifier la tâche aux développeurs.

— Flexibilité et extensibilité : l'ontologie doit permettre l'ajout de nouveaux éléments

RDF(S)	OWL(2)
Avantages	
• Fournit les éléments de base pour décrire et organiser des connaissances. • Relativement simple • Traitement et raisonnement rapide	• Syntax forte • Version améliorer de RDF(S) • Grands nombres d'outils • Plus expressif que RDF(S) • Standard W3C
Inconvenients	
• Expressivité limitée	• Assez complexe • Faible performance (demande plus de temps et de puissance dans le traitement)

TABLE 2.4.2 – Langage ontologie - Sémantique Web

contextuels ainsi que les relations.

— Généricité : L'ontologie doit supporter différents types de contexte.

— Expressivité : L'ontologie doit être assez souple pour décrire toutes les informations du contexte dans les moindres détails.

Comme technique de modélisation ontologique, nous pouvons citer OWL, RDF ou encore le langage CoOL [61]

Synthèse sur les modèles de contexte

Comme nous l'avons vu dans cette section, il y existe plusieurs approches pour modéliser le contexte. Évidemment, la liste de techniques de modélisation parcourue n'est pas exhaustive. Il s'agit des approches les plus référencées dans la littérature. Il est important de signaler que le choix du type d'approche dépend généralement de la complexité des informations du contexte à modéliser. Il n'y a pas de bonne ou mauvaise technique de modélisation. Chacune des techniques de modélisation de contexte citées possède ses avantages et ses inconvénients. Le tableau 2.4.3 présente de façon synthétique ces différentes approches.

De nombreux chercheurs sont arrivés à la conclusion que les ontologies sont théoriquement la meilleure approche pour représenter le contexte. Bien qu'une ontologie est assez complexe à mettre en place et qu'il n'est pas aisé de les manipuler, cette approche permet l'évolutivité et la résistance aux ambiguïtés [7][66].

```
<instance xmlns=http://demo.heywow.com/schema/cool
          xmlns:a=http://demo.heywow.com/schema/aspects
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <contextInformation>
    <entity system="urn:phonenumber">+49-179-1234567</entity>
    <characterizedBy>
      <aspect name="GaussKruegerCoordinate">
        <observedState xsi:type="a:o2GaussKruegerType">367032533074</observedState>
        <units>10m</units>
      </aspect>
      <certaintyOfObserver>90</certaintyOfObserver>
    </characterizedBy>
  </contextInformation>
</instance>
```

FIGURE 2.4.4 – Modélisation par ontologie

Quel que soit le modèle de contexte choisi, les informations suivantes doivent s'y retrouver :

— Le type d'information du contexte : il s'agit du nom d'une catégorie d'information telle que la localisation, la température, etc. Il est important de bien choisir les noms des types d'information. La figure 2.3.1 montre des exemples des types d'informations.

— La valeur de l'information du contexte : il s'agit de la valeur assignée à l'information du contexte. Elle contient généralement la valeur directement récupérée par le capteur. Cette valeur possède généralement une unité. Par exemple : Température = 4°.
Dans certains cas, il est préférable d'utiliser une information plus abstraite (information interprétée) que celle récupérée par les capteurs. Par exemple : Température = froid.

2.4.4 Couche de transmission du contexte

Cette couche se présente comme une couche intermédiaire entre les applications et le système de gestion du contexte. Elle permet de garantir, aux différents applications et services, un accès transparent aux informations du contexte. Ces informations sont récupérées et traitées dans les couches inférieures. La mise en place de cette interface facilite grandement le développement d'applications clientes. Ces dernières peuvent accéder au contexte de deux

Modèle	Avantages	Inconvénients	Degré de formalisme	Expressivité	Implémentation	Résistance aux ambiguïtés
Clé-valeur	• Simple • Flexible • Rapidité de traitement et de recherche	• Couplage fort avec l'application • Manque d'expressivité et de structure • Pas de validation	–	–	++	–
XML	• Représentation structurée • Validation des schémas • Flexible	• Pas de standard • Traitement plus complexe en fonction de la profondeur du schéma	+	±	+	–
Graphique	• Facilité de lecture • Relations entre composants	• Pas de standard • Requiert une technologie de modèle pour l'implémentation	±	+	±	–
OO	• Permet une modélisation relationnelle • Intégré en utilisant les langages de programmation OO	• Difficile de retrouver l'information • Pas de validation	+	+	+	+
Logique	• Traitement rapide • Approche très formelle	• Limité à certains domaines • Couplage fort avec l'application • Pas de standard	++	+	–	+
Ontologique	• Haut degré de formalisme • Mécanisme de raisonnement • Validation, Cohérence et résistance aux ambiguïtés • Standardisation	• Traitement complexe pour les ontologies de grandes tailles • Difficile à implémenter	+	+	–	++

TABLE 2.4.3 – Les informations du contexte

manières :

— La méthode synchrone : le serveur envoie les informations sur demande explicite du client. Ce dernier envoie une requête au serveur pour demander l'information et attend la réponse du serveur avant de continuer le traitement. Il s'agit d'un service à la demande.

— La méthode asynchrone : elle s'appuie sur le système de notification. Le client s'inscrit auprès du serveur afin de recevoir l'information du contexte souhaitée. Le serveur notifie le client à chaque fois qu'il reçoit une nouvelle valeur de l'information demandée. C'est la méthode « publish/subscribe ».

Étant donné que les systèmes actuels évoluent dans un environnement dynamique et que les informations du contexte changent très souvent, l'approche asynchrone est généralement recommandée. Elle est aussi la plus performante, car elle est non bloquante.

2.4.5 Couche application

C'est la couche dans laquelle sera implémentée l'application qui utilise le contexte. L'application utilisera les mécanismes de la quatrième couche pour accéder au contexte. Comme nous l'avons vu dans la section précédente, elle peut utiliser des méthodes synchrones ou asynchrones. Une fois le contexte obtenu, l'application devra donc réagir en fonction des différents contextes. Elle devra donc implémenter les différentes méthodes de réaction au changement de contexte. Ces réactions sont généralement une ou plusieurs adaptations sur différents composants de l'application. Une telle application est donc qualifiée de sensible au contexte. Il est important de signaler également que l'application peut encapsuler son propre mécanisme interne de gestion et d'interprétation du contexte et y ajouter certaines informations. La notion d'adaptation sera abordée en détail dans le chapitre 4.

2.5 En résumé

Ce chapitre nous a permis de poser les fondations de notre étude sur les systèmes sensibles. Nous avons abordé la notion de contexte. Différentes définitions ont été données au contexte dans les travaux existants. Malgré le manque de consensus dans la définition, plusieurs chercheurs s'accordent sur certaines caractéristiques que possède le contexte [53]. Ces caractéristiques sont :

— Il n'y a pas de contexte sans contexte : le contexte dépend généralement de la finalité.

— Le contexte est interprétable : la capture des informations du contexte n'est pas une fin en soi. Ces données peuvent être interprétées afin d'obtenir d'autres informations de plus haut niveau.

— Le contexte possède plusieurs acteurs : il peut s'agir de l'utilisateur ou du système.

— Le contexte est infini et évolutif : les informations sur le contexte sont très nombreuses, voire infinies, et changent au cours du temps. Le contexte n'est pas figé, il évolue.

Nous avons également vu dans ce chapitre, une approche de gestion de contexte. L'avantage de cette approche en plusieurs couches est qu'elle permet une séparation entre la capture du contexte et son utilisation. Dans la couche de modélisation, nous avons parcouru les

différentes approches existantes pour modéliser le contexte. Ces approches sont résumées dans le tableau 2.3.1.

Chapitre 3

État de l'art sur les systèmes sensibles au contexte

3.1 Définition

Traduction de l'expression anglaise «context-awareness», la sensibilité au contexte est un terme qui a été introduit par Schilit [57]. Ce terme désigne la capacité d'un système à s'adapter aux changements contextuels. Un système sensible au contexte est conscient du contexte dans lequel il s'exécute, et adapte son exécution en fonction des variations dans le contexte. La définition la plus utilisée par la majorité des chercheurs dans leurs travaux est celle de Dey et coll. [22] : « Un système est sensible au contexte s'il utilise le contexte pour fournir des informations et des services pertinents pour l'utilisateur. La pertinence dépend de la tâche demandée par l'utilisateur ».

3.2 Catégories des systèmes

Dans la littérature, plusieurs infrastructures dépendantes du contexte ont été proposées. Elles se différencient les unes des autres généralement dans leurs architectures, leurs implémentations, mais également dans leurs approches de modélisation du contexte.

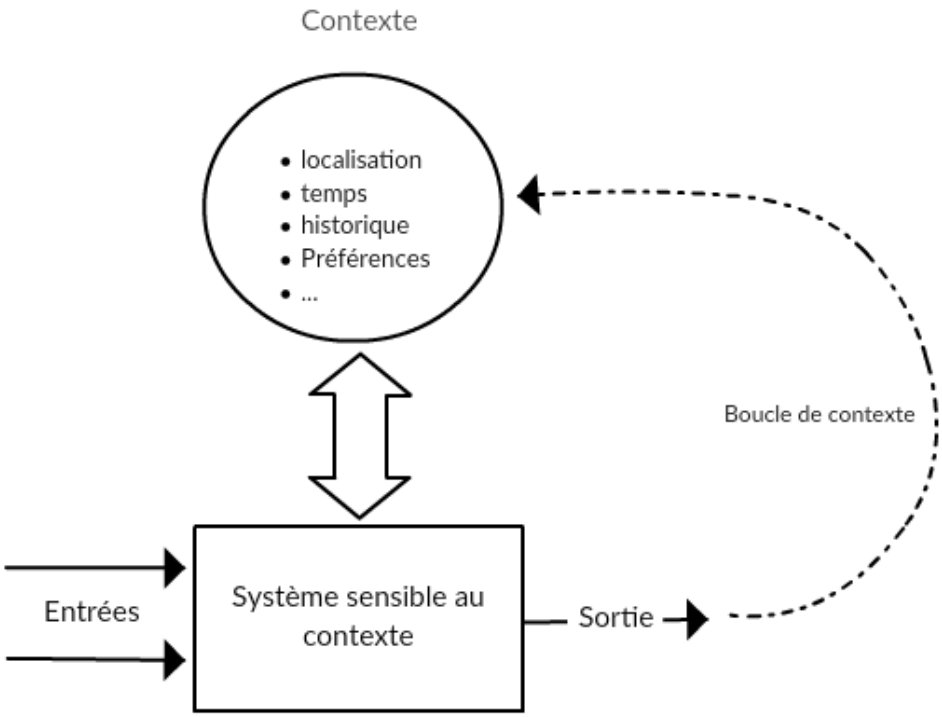


FIGURE 3.1.1 – Application sensible au contexte

3.2.1 En fonction de la méthode d'acquisition

En se basant sur la méthode d'acquisition de l' information, on peut distinguer les trois approches suivantes [7] :

— **L'acquisition directe**. Cette approche est utilisée lorsque les capteurs sont directement connectés au client. L'application cliente a directement accès à l'ensemble d'informations depuis les capteurs, il n'y a pas de couches intermédiaires, les pilotes des capteurs sont intégrés à l'application. L'architecture se résume à un couplage direct entre la couche de capture et l'application cliente.

— **L'approche middleware**. Cette approche se base sur une infrastructure à plusieurs couches et permet l'utilisation des méthodes d'encapsulation. Les avantages de cette approche

sont certainement la gestion de l' hétérogénéité de l' information et la facilité d'extension du système grâce au mécanisme d'encapsulation. Ce qui n'était pas le cas avec une approche directe.

— **L'approche serveur de contexte**. Cette approche a pour but d'alléger le terminal client dans les traitements du contexte, et lui permet d'accéder à des sources de données distantes et distribuées. Il s'agit de l'une des approches les plus utilisées, car très souvent, les terminaux utilisant des applications sensibles au contexte sont souvent limités en puissance de calcul, en espace mémoire, etc. Cette approche serveur peut être vu comme une extension de l'approche middleware, et a comme principal avantage : des meilleures performances dans la gestion du contexte. Cependant, ces performances dépendent beaucoup du protocole de communication client-serveur choisi, de la qualité du réseau, etc.

3.2.2 En fonction de la coordination des composants et des processus

Les deux approches principales en fonction de la coordination des composants et des processus sont : le modèle widget et le modèle Blackboard [31].

— **Le modèle Widget** a pour caractéristique principale de permettre l'accès direct aux informations du contexte via une requête à un widget. Le widget est un intermédiaire entre une application qui souhaite obtenir des informations et une source d'informations. Ces applications doivent s'inscrire auprès du widget pour pouvoir accéder à ces informations. Il s'agit d'une sorte de modèle publish/suscribe[26] sur chaque widget. Le principal inconvenant de cette approche est que l'inscription et la désinscription des applications auprès des widgets sont moins robustes et moins flexibles lorsqu'il y a un changement dans le contexte.

— **Le modèle Blackboard**, contrairement au modèle widget, centralise toutes les informations dans un seul élément appelé le «Blackboard». Les applications s'inscrivent auprès de ce seul élément afin d'avoir accès aux différentes informations du contexte. Ce modèle est centré sur les données. Toutes les informations sont traitées par le Blackboard qui se charge de notifier les applications inscrites lorsqu'un événement donné se produit.

Le principal avantage de cette approche est que le changement de capteur ou de configuration n'a pas d'impact sur les applications, car elles ont accès aux informations uniquement par le Blackboard qui est finalement une sorte de boite noire pour les applications, car elles ignorent la manière dont il traite l'information. Il est donc facile d'ajouter des capteurs ou de faire des changements dynamiques.

3.2.3 En fonction du modèle du contexte

Les infrastructures diffèrent aussi par leur modèle de contexte. Le modèle du contexte est une composante majeure des systèmes dépendants du contexte. Il permet de modéliser, définir et interpréter le contexte. Les différents modèles de contexte ont déjà été présentés dans la section 2.4.3.

3.3 Frameworks et architectures existantes

On retrouve dans la littérature plusieurs frameworks pour la sensibilité du contexte. Le tableau suivant reprend les principaux systèmes et leurs approches respectives de modélisation du contexte [7].

Nom	Type Architecture	Modèle de contexte	Méthode Interprétation du contexte	Historique de contexte	Sécurité et Confidentialité
CASS	Middleware centralisé	Modèle relationnel	Moteur d'inférence et base de connaissances	✗	✓
CoBra	Agent (module d'acquisition du contexte)	Ontologies (OWL)	Moteur d'inférence et base de connaissances	✗	✓
CMF	Blackboard	Ontologies (RDF)	Service de reconnaissance de contexte	✓	✗
Context Toolkit	Widget	Clé-valeur	Interprétation de contexte et agrégation	✓	✓
CORTEX	Sentient object model	Modèle relationnel	Module de contexte	✓	✓
Gaia	MVC (étendue)	Modèle logique	Service de reconnaissance de contexte basé sur la logique du premier ordre	✓	✓
Hydrogen	Architecture en couche (3)	Modèle orienté-objet	Interprétation et agrégation des données brutes	✗	✗
SOCAM	Serveur de contexte	Ontologies (OWL)	Moteur d'inférence	✓	✓

TABLE 3.3.1 – Tableau synthèse des frameworks pour la sensibilité au contexte

Chapitre 4

L'adaptation

4.1 Introduction

Comme souligné dans l'introduction générale, l'avènement de l'informatique ubiquitaire apporte certaines contraintes. En effet, il est important de développer des systèmes (sensibles au contexte) capables de prendre en compte le dynamisme qu'apporte l'informatique ubiquitaire et de réagir en fonction des différents changements dans l'environnement.

La sensibilité au contexte a été introduite comme étant la capacité qu'a un système de réagir aux changements qui peuvent survenir dans le contexte dans lequel il se trouve. On dit alors que le système s'adapte au contexte et il est qualifié de système adaptable au contexte. C'est cette notion d'adaptation que nous allons aborder en détail dans ce chapitre. Nous allons, dans un premier temps, définir ce qu'est une adaptation. Nous allons parcourir les définitions des différents dictionnaires ainsi que celles que l'on retrouve dans la littérature. Par la suite, nous allons voir les différents types d'adaptation ainsi que les techniques d'adaptation.

4.2 Définitions de l'adaptation

Comme dans le chapitre 2, nous allons commencer par quelques définitions que l'on retrouve dans les différents dictionnaires :

- **Le Larousse** :
  (a) « Action d'adapter ou de s'adapter à quelque chose : adaptation aux circonstances. »
  (b) « Action d'adapter une œuvre, un texte pour un public, une technique artistique différents ; œuvre ainsi réalisée. »
- **Le-dictionnaire.com** :
  (a) « Fait d'adapter, de s'adapter, ou résultat de ce fait. »
  (b) « Arrangement d'une œuvre littéraire ou cinématographique. »
- **Le grand dictionnaire terminologique** :
  (a) « Action, pour une organisation, de modifier une conduite, une situation ou des règles en fonction de l'évolution du milieu, de conditions nouvelles ou d'une situation particulière. »
  (b) « Opération qui consiste à apporter des modifications à un logiciel ou à un système informatique, à la fin de son développement, dans le but d'améliorer ses performances dans un contexte précis d'utilisation. »

De ces définitions, nous pouvons retenir que l'adaptation est le fait de modifier ou ajuster dans un but donné. Il peut s'agir de la modification d'un comportement, d'une œuvre, d'un logiciel en fonction d'une propriété, un contexte, un public, etc.

En informatique, l'adaptation est le fait qu'un composant logiciel ou un système modifie ses propriétés dans certaines conditions. Dans ce travail, on se focalise spécialement sur l'adaptation au contexte. On parle alors des systèmes adaptables au contexte qui, en fonction des variations du contexte, adaptent leurs composants (Interface utilisateur, noyau fonctionnel, données).

4.3 Importance de l'adaptation

L'informatique a connu une avancée fulgurante ces vingt dernières années et de nombreux progrès ont été réalisés. On note de manière générale que :

— Les logiciels sont de plus en plus sophistiqués ;
— Les réseaux sont également de plus en plus grands ;
— Les capteurs sont de plus en plus présents dans l'environnement ;

— Les ordinateurs tendent de plus en plus vers la miniaturisation (mini-ordinateur, tablette, smartphone) ;

— Etc.

Cette évolution a donné lieu à ce qu'on appelle la troisième de l'informatique, l'informatique ubiquitaire, celle dans laquelle l'utilisateur est de plus en plus mobile , il utilise diverses plates-formes et évolue dans des environnements plus variés. Il devient très difficile de connaître en amont le contexte d'usage (utilisateur, plate-forme, environnement [65]) lors de la conception d'un système, car il est de plus en plus varié, variable et potentiellement imprévisible. Ce changement peut paraître séduisant au premier abord, mais il apporte de nouveaux défis en ingénierie du logiciel et en interaction homme-machine. Lors du processus de conception, les concepteurs devront prendre en compte :

– **La diversité des utilisateurs :** généralement conçues pour une classe donnée d'utilisateur, les applications doivent de nos jours élargir leurs possibilités en s'adaptant à divers profils d'utilisateur. Étant donné que la technologie est plus en plus accessible à tout le monde, une application est susceptible d'avoir différents types d'utilisateurs : des utilisateurs initiés à l'informatique aux utilisateurs non initiés, des personnes jeunes aux personnes plus âgées, des personnes valides aux personnes vivant avec un handicap, etc. Il est clair que l'application doit se présenter de manière différente en fonction du type d'utilisateur. Elle va par exemple proposer la visualisation du résultat d'une demande à une personne sourde et à une personne aveugle, elle proposera une description audio du résultat. En fonction donc des préférences de l'utilisateur, l'application doit être capable de s'adapter afin d'offrir à l'utilisateur des services adéquats.

– **La diversité des plates-formes :** Le nombre et la diversité des plates-formes ont littéralement explosé ces dernières années. À l'ordinateur classique viennent s'ajouter le notebook, le smartphone, le PDA, la tablette numérique, la liseuse, la smartwatch, etc. De plus avec l'informatique ubiquitaire et l'internet des objets, tout objet qui nous entoure est probablement appelé à devenir un jour un dispositif intelligent avec lequel l'homme peut interagir. On parle alors très souvent de smart TV, de réfrigérateurs intelligents, de vêtements intelligents, etc. Bien que certains de ces dispositifs n'en sont qu'à leurs débuts, les concepteurs d'applications devront néanmoins prendre en compte cette diversité de plate-forme lors de la conception d'une application afin qu'elle puisse s'adapter au plus grand nombre possible de plates-formes en prenant en compte les spécificités de chaque plate-forme allant des gros calculateurs au dispositif limité en puissance de calcul, des grands écrans au plus petit écran, des plates-formes fixes aux plates-formes mobiles, etc.

– **Diversité des environnements d'interaction :** Grâce à la facilité de mobilité, la miniaturisation des ordinateurs et l'expansion des réseaux informatiques, l'utilisateur est de plus en plus mobile, se déplace avec son ordinateur et continue de réaliser ses tâches tout en changeant régulièrement d'environnement. Par exemple, un utilisateur peut gérer ses rendez-vous grâce à une application donnée dans son bureau, puis se déplacer dans une salle de réunion tout en continuant d'utiliser l'application. Cette application devra cependant prendre en considération le changement d'environnement et adapter son fonctionnement en désactivant par exemple les fonctions de rappels sonores afin de ne pas perturber la réunion de l'utilisateur. Grâce à l'expansion des capteurs, les différentes informations sur l' environnement dans lequel l'utilisateur se trouve (localisation, température, heure, etc.) sont accessibles. Les concepteurs devront donc prendre en comptes ces nouvelles contraintes lors de la conception de nouvelles applications.

Toutes ces nouveautés sont donc à prendre en compte dans le processus de conception et de développement d'application. Avec toutes ces nouvelles exigences, il est donc important de rendre les applications, ainsi que leurs interfaces, adaptables à ce genre de contexte susceptible de varier.

4.4 Exemples d'adaptation au contexte

Après avoir introduit la notion d'adaptation et évoqué son importance, voici quelques scénarios qui illustrent l'adaptation d'une application au contexte :

Exemple 1 : Considérons une application de la société Carrefour. Cette application propose, en fonction de la localisation de l'utilisateur, le supermarché le plus proche de l'utilisateur et affiche un itinéraire pour s'y rendre. Une fois dans le supermarché, l'application affiche les différentes promotions proposées sur des articles que l'utilisateur achète régulièrement.

Exemple 2 : Considérons une application qui sert d'assistant personnel. L'application permet à l'utilisateur de gérer son emploi du temps, son agenda, ses mails, etc. L'application permettra les adaptations suivantes :

- Si l'utilisateur est en train de conduire par exemple, l'application lira les mails grâce à la synthèse vocale.

- Si l'utilisateur se trouve dans un milieu bruyant, elle suggéra une lecture personnelle de l'utilisateur.
- Lorsque l'utilisateur se trouve en réunion, l'application pourra désactiver la sonnerie, etc.

4.5 Difficultés liées à l'adaptation

Construire des applications adaptables se présente comme une solution idéale pour répondre aux défis posés par la variabilité du contexte. Cette adaptation doit se faire à tous les niveaux du logiciel c'est-à-dire :

— au niveau de l'interface utilisateur ;

— au niveau du noyau fonctionnel ;

— au niveau des données.

Garantir cette adaptation n'est pas toujours facile. On pourrait croire que la connaissance en amont des différents contextes d'utilisation de l'application permettrait une approche de développement au cas par cas c'est-à-dire pour chaque contexte proposer une solution d'adaptation aux trois niveaux du logiciel. Cette approche présente les inconvénients suivants :

- Il faut connaître en amont tous les contextes d'usage possibles de l'application c'est-à-dire anticiper tous les environnements possibles, tous les utilisateurs possibles et toutes les plates-formes possibles [64]. En réalité, ce n'est pas toujours le cas. Il est très difficile de connaître tous les contextes dans lesquels l'utilisateur peut se trouver. Il faut prévoir également une solution lorsque l'on se retrouve dans un contexte non pris en charge.
- Même si les différents contextes sont connus en amont, ce nombre sera forcement très grand. La multiplicité des plates-formes et des contextes d'usage compromet clairement cette pratique. Il sera donc difficile de faire un traitement au cas par cas, car cela demanderait aux développeurs un travail conséquent. Par exemple si pour chaque plate-forme une interface graphique doit être développée, comment maîtriser le développement et la maintenance de ce grand nombre d'interfaces ? Comment garantir la cohérence entre ces différentes interfaces ? Comment garantir une certaine continuité de l'interaction tout en maintenant la convivialité lorsque l'utilisateur passe d'une plate-forme à une autre ?

- Les différentes technologies employées dans chaque plate-forme se présentent également comme une difficulté pour une adaptation, car cela demanderait une maîtrise de toutes ces technologies pour un développement au cas par cas.

4.6 Raisons de l'adaptation

Après avoir mis en œuvre une application, plusieurs raisons peuvent conduire à l'adapter. Ces raisons peuvent être classées en quatre catégories [38] :

4.6.1 Adaptation correctionnelle

Dans certains cas, on remarque que l'application en cours de son exécution ne se comporte pas correctement, ou comme prévu. La solution est d'identifier le module de l'application qui cause le problème et de le remplacer par une nouvelle version supposée correcte. Cette nouvelle version fournit la même fonctionnalité que l'ancienne, elle se contente de corriger ses défauts.

4.6.2 Adaptation évolutive

Au moment du développement de l'application, certaines fonctionnalités ne sont pas prises en compte. Avec l'évolution des besoins de l'utilisateur, l'application doit être étendue avec de nouvelles fonctionnalités. Cette extension peut être réalisée en ajoutant un ou plusieurs modules pour assurer les nouvelles fonctionnalités ou même en modifiant les modules existants pour étendre leurs fonctionnalités tout en gardant l'architecture de l'application.

4.6.3 Adaptation perfective

L'objectif de ce type d'adaptation est d'améliorer les performances de l'application. À titre d'exemple, on se rend compte que l'implémentation d'un module n'est pas optimisée. On décide alors de remplacer l'implémentation du module en question. Un autre exemple peut être un module qui reçoit beaucoup de requêtes et qui n'arrive pas à les satisfaire. Pour éviter

la dégradation des performances du système, on diminue la charge de ce module en installant un autre module qui partage la charge avec le premier.

4.6.4 Adaptation adaptative

Même si l'application développée s'exécute correctement, parfois son environnement d'exécution (comme le système d'exploitation), les composants matériels (ou d'autres applications ou données dont elle dépend) changent. Dans ce cas, l'application est adaptée en réponse aux changements affectant son environnement d'exécution. C'est ce type d'adaptation que nous abordons dans ce mémoire.

4.7 Caractérisation

Après avoir présenté les différentes raisons d'utilisation de l'adaptation dans une application, nous énumérons dans cette section les différentes classifications utilisées dans les travaux existants sur l'adaptation. Cette classification élaborée par [18] se fait sur plusieurs axes : statique ou dynamique, centralisée ou distribuée, comportementale ou architecturale.

4.7.1 Adaptation statique et adaptation dynamique

L'adaptation statique ou manuelle consiste à préparer plusieurs versions de la même ressource avant son exploitation. Cette ressource peut être un document, une image, une vidéo ou même une application. La ressource est adaptée manuellement avant de pouvoir la réutiliser dans son nouveau contexte. Cette technique a été adoptée pour plusieurs développements au début de l'apparition des terminaux mobiles afin de pouvoir exploiter des ressources initialement prévues pour des PC standards. Avec une adaptation statique, le résultat est dédié au nouveau contexte d'utilisation de la ressource ce qui garantit la sûreté de son exploitation. Cependant, elle présente l'inconvénient de la difficulté d'évolution pour la prise en charge de nouveaux contextes d'utilisation.

Contrairement à l'approche statique, l'adaptation dynamique effectue les transformations sur la ressource au cours de son utilisation. Un système d'adaptation automatique intercepte les requêtes des utilisateurs et effectue à la volée des transformations suivant les caractéristiques du contexte d'utilisation actuel. Cependant, des problèmes d'adaptation peuvent surgir dans un nouveau contexte d'utilisation inconnu, en fournissant un résultat d'adaptation qui n'est peut-être pas adéquat à ce contexte. De plus, l'adaptation dynamique augmente automatiquement la latence de renvoi de l'information demandée. Ce qui peut gêner l'utilisateur.

Malgré un grand intérêt pour l'adaptation dynamique, l'adaptation statique reste aussi un moyen sûr et efficace pour assurer l'exploitation des ressources dans différents contextes d'utilisation. Cependant, l'utilisation des deux approches de façon complémentaire semble être la solution idéale.

4.7.2 Adaptation verticale et adaptation horizontale

L'adaptation peut concerner une ressource locale à une machine ou distribuée sur plusieurs machines. Dans le cas où la ressource est distribuée, le processus d'adaptation peut être centralisé ou réparti sur les différents pairs où les différentes parties de la ressource ont été stockées. Si l'adaptation est centralisée alors elle est qualifiée de verticale, sinon elle est horizontale.

4.7.3 Adaptation comportementale et Adaptation architecturale

L'adaptation peut être qualifiée de comportementale (ou algorithmique) lorsqu'elle modifie le comportement de la ressource sans affecter sa structure interne. Ceci facilite l'implantation et l'exploitation de l'adaptation. En effet, puisque la structure est la même, la nouvelle ressource reste directement exploitable comme la ressource d'origine. L'implantation est aussi facilitée, car cette adaptation n'effectue pas de changements profonds qui demanderaient une réévaluation des modifications effectuées.

L'adaptation est architecturale lorsque la structure interne de la ressource peut être modifiée en fonction des besoins applicatifs de l'utilisateur. Cette solution est intéressante, car elle offre un haut degré d'adaptabilité, mais elle reste très difficile à implanter. Elle ne reste possible que si l'on dispose d'une connaissance suffisante sur l'architecture de l'application et ses différents composants.

4.8 Acteurs de l'adaptation

Les acteurs sont les intervenants dans le processus d'adaptation, l'entité qui déclenche l'adaptation. Ce processus comporte les 4 étapes suivantes : initiative, suggestion, décision, exécution [24].

Selon les types d'acteurs à la base de l'adaptation (initiative), on distingue :

— Un système adaptable, lorsque l'adaptation peut être réalisée à tout moment sur demande explicite de l'utilisateur. Par exemple, l'utilisateur peut choisir l'aspect physique des éléments de l'interface en fonctions de ses préférences.

— Un système adaptatif, lorsque l'adaptation est déclenchée automatiquement c'est-à-dire sans intervention explicite de l'utilisateur. Le système prend en compte les besoins et habitudes de l'utilisateur et déclenche l'adaptation adéquate. La mise à jour du modèle utilisateur est réalisée par le système lui-même, par observation des actions et des réactions de l'utilisateur.

4.9 Instants d'adaptation

D'un point de vue du système, on distingue, trois principaux moments d'adaptation : au développement, à l'exécution et à l'installation.

— Adaptation au développement : l'adaptation est alors réalisée lors de la conception par les développeurs qui ont prévu l'ensemble des cibles possibles. Par exemple, les outils comme UIML [2] ou AUIML [47] permettent la production d'interfaces adaptées à différentes cibles, mais ces IHM ne peuvent pas s'adapter à l'exécution.
— Adaptation à l'exécution : comme son nom l'indique, l'IHM s'adapte ici à la volée. Ex. : La boîte à outils XAML de visual Studio.
— Adaptation à l'installation du logiciel : le produit s'adapte et/ou est configuré pour correspondre à la cible donnée. Par exemple le Finder de Mac OS, peut être configuré à l'installation pour proposer un sous-ensemble des fonctionnalités du système à travers des menus simplifiés (Finder simplifié).

4.10 Mécanismes d'adaptation

4.10.1 Paradigmes existants

Afin de mettre en place une adaptation, il existe plusieurs paradigmes offrant des caractéristiques bien précises et permettant de répondre aux besoins d'un système sensible au contexte :

— La programmation événementielle : elle permet à un composant de modifier son comportement lorsqu'un événement particulier se produit. Ce paradigme offre des capacités de réactivité et de dynamicité maximales. Il permet de réduire le couplage entre les différents composants d'un système.

— Architecture orientée service : il s'agit d'une approche distribuée qui permet de gérer l'hétérogénéité des différents composants dans un système. Cette approche garantit également l'interopérabilité.

— La programmation par composant : introduit dans les années 90, ce paradigme permet de concevoir une application comme un assemblage d'entités logicielles (composant). Ce qui garantit la modularité, la flexibilité, la réutilisabilité et surtout la facilité de maintenance des systèmes.

— La programmation par aspect : basée sur le mécanisme de tissage d'aspect, ce paradigme permet une bonne séparation des «préoccupations». Ce qui assure une certaine modularité et une facilité dans la maintenance du code.

4.10.2 Techniques existantes

Dans cette section, nous allons parcourir quelques techniques qui peuvent être utilisés pour mettre en place une adaptation :

— MOP (Meta Object Protocol) et réflexivité : l'utilisation des langages supportant la réflexivité permet de modifier facilement le comportement des composants d'une application. La réflexivité permet à un composant de faire deux choses :

  1. s'observer. Le composant analyse son propre état. C'est la phase d'introspection.
  2. agir sur lui même. Le composant modifie son comportement. C'est la phase d'intercession.

  La réflexivité peut donc permettre à une application de modifier dynamiquement son comportement suite à des changements dans son environnement.

— Proxy : «design pattern» qui permet de rediriger l'appel d'un objet donné vers un autre objet, appelé «proxy», qui va représenter l'objet appelé. [46]

— Container : permet de modifier le comportement d'une application en paramétrant l'assemblage des composants. En effet, le conteneur gère les différentes propriétés d'un composant et ses interactions avec d'autres composants.

— Tissage : il s'agit du mécanisme de base de la programmation par aspect comme nous l'avons vu précédemment. Grâce au tissage d'aspect, on peut facilement altérer le comportement d'une application lors de son exécution en ajoutant ou supprimant dynamiquement des fonctionnalités[46].

— Interception middleware : les appels de méthode et le passage de réponses à travers les couches d'un logiciel peuvent être interceptés et redirigés vers différentes entités liées au middleware [46]

— Adaptation par contrat : permets de spécifier de manière explicite les règles d'adaptation à l'aide d'une «politique»(policy). La gestion d'un système basée sur une politique (politique-based management) consiste à spécifier un ensemble de règles qui sont appliquées dans des situations précises tout en assurant la cohérence et la sécurité. La séparation entre la politique et l'implémentation d'un système offre des avantages. Elle permet de changer dynamiquement la stratégie de gestion du système en modifiant la politique. Ainsi, on modifie le comportement d'un système sans changer son implémentation [17]. Dans une adaptation, cette «politique» consiste en un fichier qui rassemble l'ensemble des règles d'adaptation à appliquer dans un contexte donné.

— Génération de code :consiste à générer et insérer dynamiquement du code dans un programme en cours d'exécution afin de modifier son comportement. Cette approche reste assez difficile à mettre en place et assez coûteuse en termes de performances.

4.11 Adaptation des interfaces utilisateurs

Il est important lors d'une adaptation au contexte d'adapter également l'interface utilisateur à la plate-forme qui exécute l'application, car les différences de ressources matérielles nécessaires à l'interaction entre l'utilisateur et le système, comme la taille de l'écran ou les capacités d'interactions, diffèrent d'une plate-forme à l'autre. Ainsi l'interface utilisateur sur un téléphone portable sera potentiellement différente de celle d'un ordinateur de bureau.

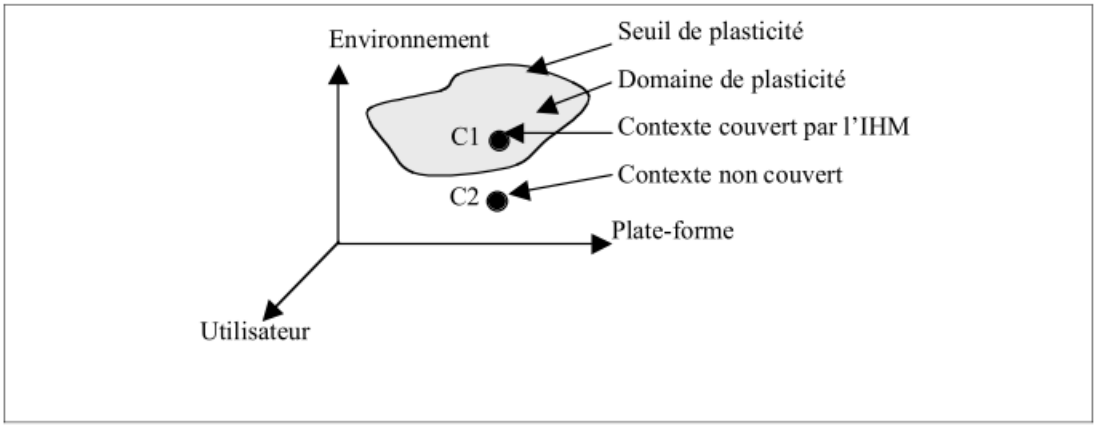
Pour répondre à cette diversité de plate-forme, on parle souvent de deux types d'interfaces utilisateurs :

— Les interfaces plastiques qui sont des interfaces possédant la capacité de s'adapter ou d'être adapté au contexte actuel tout en maintenant l'utilisabilité. Par contexte, on entend bien sûr la plate-forme, mais aussi l'utilisateur et l'environnement [65].

— Les interfaces multicibles qui se contentent de s'adapter uniquement aux différentes plates-formes.

Dans les sections suivantes, nous aborderons les différents outils qui permettent de dévelop-

per ce genre d'interface.



[14]

FIGURE 4.11.1 – Plasticité et contexte

4.11.1 Support au développement

On distingue classiquement deux catégories d'outils :

- Les outils de codage qui se résument essentiellement aux différentes librairies permettant de développer un système en codant.
- Les outils de spécification qui permettent d'écrire des spécifications et de générer l'interface utilisateur.

4.11.2 État de l'art sur les outils de spécification d'IHM adaptatif

a) UIML

UIML est un langage déclaratif cherchant à masquer la diversité des plates-formes et des langages de développement — Swing, Waba, HTML, WML, VoiceXML, etc.— il s'agit d'un langage simple pour lequel il existe aujourd'hui des compilateurs pour Java/JFC, PalmPilot, HTML, VoiceXML, et WML. Mais avec UIML, la spécification de la présentation est dépendante de la plate-forme. Si le développeur vise une IHM pour WML et pour Java/JFC, il se doit

de produire deux spécifications distinctes : l'une utilisera les tags WML, et la seconde utilisera les classes AWT/Swing avec des gestionnaires de présentation (Layout manager) Java. Pour pallier, ce problème [2] fait une double proposition :

— Un système d'abstraction des interacteurs selon le principe des OIA. Celui-ci fait partie des spécifications UIML 2.0 avec la combinaison des tags <peers> et <presentation>.

— Un système de gestion de la présentation qui fait encore l'objet de travaux [4]

b) AUIML

AUIML (Abstract User Interface Mark-up Language)[47] est un langage visant la génération automatique de code multiplateforme (Dynamic HTML, Java Swing, Palm-Pilot). Il résout l'un des principaux problèmes de UIML puisque la spécification est indépendante de la plateforme cible. En effet, à aucun moment, le programmeur n'utilise un JButton ou un tag «<href ...» dans sa description AUIML. AUIML [21] permet de décrire l'interface en termes :

— des éléments manipulés (données initiales, structures et données complexes telles que les images ou le son) ;

— des éléments d'interaction : types simples mêlant de la structure et des fonctionnalités (choix, groupe, table, arbre) ;

— des éléments d'actions : permet de décrire un micro dialogue pour la gestion d'événements entre l'interface et les données.

Un autre point intéressant de AUIML par rapport à UIML est l'utilisation explicite d'un gestionnaire de navigation [21]. Ce gestionnaire construit la structure de l'interface, détermine le choix des systèmes de navigation en fonction de la structure des données décrites. Par contre, cet outil, propriété d'IBM, est très peu publié et n'est pas encore finalisé.

c) UsiXML

UsiXML[43] est un des langages les plus référencés dans la littérature lorsqu'on parle des UIDL. En effet avec UIML et XFORMS, il fait partie des UIDL standardisés. Comme la plupart des UIDL, il est déclaratif et se base sur XML. Il permet de spécifier différents types d'interfaces utilisateurs : les interfaces utilisateurs graphiques (GUI), les interfaces utilisateurs

orientées caractères (CUI), les interfaces utilisateurs orales et les interfaces utilisateurs multimodales. Cette spécification peut être faite à différents niveaux d'abstraction et l'on peut ainsi obtenir plusieurs interfaces utilisateurs finales en fonction du contexte. Sur ce point, UsiXML se rapproche un petit peu de l'approche IDM. Ce langage possède les caractéristiques :

- UsiXML décrit à un niveau élevé de l'abstraction les éléments de constitution de l'interface utilisateur d'une application : widgets, commandes, récipients, modalités, techniques d'interaction.
- UsiXML est indépendant du dispositif : une interface utilisateur peut être décrite de manière autonome en ce qui concerne les modalités et les dispositifs utilisés dans les interactions telles que la souris, écran, clavier, système d'identification de voix, etc.
- UsiXML est indépendant de la plate-forme : une interface utilisateur peut être décrite de manière autonome en ce qui concerne les diverses plates-formes de calcul, telles que le téléphone portable, le PC de poche, PC de comprimé, ordinateur portable, ordinateur de bureau
- UsiXML permet la spécification de plusieurs modèles impliqués dans le développement d'une interface utilisateur comme : le modèle de taches, le modèle du domaine, le modèle de présentation, le modèle de dialogue, et le contexte d'utilisation.
- Les logiciels utilisés sont : ReversiXML, TransformXML, GrafiXML
- Ce langage permet la composition d'interfaces (utilisant ComposiXML) et aussi une adaptation au contexte. Il est certainement l'un des plus puissants UIDL.

d) QTK

QTK[29] est un outil de spécification d'interface homme-machine construit au-dessus du langage Oz [Mozart]. Oz et QTk proposent au développeur un environnement de programmation rapide s'appuyant sur des principes issus de l'approche MB-IDE. En particulier, QTk permet de bien séparer la présentation d'une interface, des données à présenter. Ainsi il est aisé d'associer plusieurs interfaces graphiques à la même application. Ce principe est utilisé dans l'application FlexClock [29]. QPK permet de construire de manière simple une interface graphique qui fait de l'adaptation dynamique en fonction de la surface d'affichage. QTk propose un langage d'expression des différentes interfaces ainsi que des mécanismes de choix. Cependant, le concepteur doit décrire manuellement les différentes interfaces(vues) possibles.

e) XMMVR

XMMVR[50] est un langage basé sur XML qui permet de spécifier des scènes 3D avec interaction vocale dans un environnement de réalité virtuelle. Il est souvent qualifié de langage hybride, car il est composé d'autres langages. En effet, le langage se base sur deux standards qui sont VoiceXML ou X+V pour la partie interaction vocale et X3D ou VRML pour la partie 3D. Il permet la conception d'interface multimodale dans un environnement de réalité virtuelle.

f) HOMER UIMS

HOMER [56] est un générateur d'IHM à deux utilisateurs (dual user interface), l'un d'entre eux étant atteint de cécité. Une IHM construite avec HOMER doit satisfaire les requis suivants : l'accès aux deux utilisateurs doit être simultané, les besoins des deux parties doivent être satisfaits (quitte à utiliser des métaphores d'interaction ou une structure d'IHM différentes), à tout moment la sémantique (concepts et fonctions) doit être identique pour les deux utilisateurs.

HOMER inclut un langage de spécification et utilise les notions de méta-intéracteur (meta-widget)[69] et de métaphore d'interaction pour produire une IHM. Le langage est interfacé avec une boîte à outils graphique, pour l'interaction visuelle, et une boîte à outils multimodale, pour l'interaction non visuelle. Il permet la spécification des présentations et du dialogue.

g) XUL

XUL[48] est un langage créé dans le cadre du projet Mozilla et utilisé pour définir son interface graphique. Ce langage possède les caractéristiques :

- C'est un langage conçu spécialement pour créer des interfaces utilisateurs portables.
- Il utilise des éléments graphiques, des «widgets abstraits» pour la construction des interfaces.
- Le développement d'une application XUL se rapproche du développement d'une application cliente ou client-serveur traditionnel.

- L'une des grandes richesses de XUL est son extensibilité, et la facilité de réutilisation d'éléments.
- Spécificité par rapport aux autres UIDL :
- Les applications résultantes ne peuvent pas être appliquées aux dispositifs mobiles.

h) SUNML

SUNML[27] est un langage développé par l'équipe Rainbow, laboratoire I3S Sophia Antipolis. Il permet de définir des interfaces utilisateurs. Il est basé sur XML. Ce langage possède les caractéristiques :

- Il permet la description des composants de l'interface utilisateur de manière abstraite et basique.
- Il permet de gérer les événements des composants.
- Il comporte 19 balises et est très facilement extensible.

i) XAML

XAML[45] est un langage de description des interfaces, réalisé par Microsoft. Il nous permet de réaliser des interfaces pour applications bureau, applications web, animations... Ce langage possède les caractéristiques :

- Il utilise les composants du framework .Net : les éléments XAML ont leur équivalent en objets du CLR (Common Language Runtime) et les attributs XML deviennent les propriétés de ces objets.
- Les éléments du langage XAML correspondent à des balises de positionnement et à des «widgets», des composants d'interface utilisateur graphique. Ces composants de même que le texte utilisent des graphismes vectoriels permettant d'ajuster leur taille sans perte de définition.
- Les applications réalisées sont indépendantes du support et adaptables.
- XAML facilite grandement la création, l'édition et la réutilisation d'interfaces utilisateurs graphiques pour les applications
- Il peut être manipulé en utilisant l'IDE Visual studio ou l'extension «open source»

j) UIDE (User Interface Design Environment)

UIDE[62] est un UIMS permettant la génération automatique d'applications avec une aide adaptative. Il utilise des connaissances sur l'application, les tâches et les utilisateurs pour concevoir automatiquement une application capable de contrôler son exécution et fournir une aide adaptée. PODIUM (Personalized Designer : an Intelligent User-Interface Manager) [13] est un UIMS intégrant des connaissances sur les utilisateurs pour la génération automatique d'interfaces. Les interfaces générées sont des applications de physique devant s'adapter à la connaissance de l'utilisateur physicien.

k) XIML

XIML (eXtensible Interface Markup Language) [52]est un langage de description d'interface graphique pour différentes plates-formes. L'objectif de ce langage est d'aider au développement d'interfaces multi-plates-formes en diminuant le temps de développement et en minimisant les incohérences entre plates-formes. Dans ce but, il couvre tout le cycle de conception d'une interface en permettant la description :

- des aspects abstraits d'une interface (tâches, concept du domaine, utilisateur) ;
- des aspects concrets d'une interface (présentation, dialogue) ;
- des relations entre les éléments décrits (une présentation Y «présente» la donnée X).

XIML est un langage ouvert et extensible. La structure de base de ce langage est formée de trois grandes parties : le composant, l'attribut et la relation.

- Un composant est un élément décrivant les tâches, les concepts du domaine, les utilisateurs (aspects abstraits d'une interface), les présentations ou le dialogue (aspects concrets d'une interface).
- Un attribut appartient à un composant. À partir d'un ensemble d'attributs, il est possible de décrire les caractéristiques ou les propriétés d'un composant.
- Une relation décrit un lien entre deux ou plusieurs composants.

Le cas EMMA

EMMA est souvent reprise dans la liste des UIDL. Cependant, il s'agit d'un cas particulier comme expliquer [36]. Il est généralement utilisé comme langages de description de données échangées entre deux modules d'une IHM. Si un composant A veut transmettre des données à un composant B, A génère un document EMMA que B utilisera pour récupérer les données. On est donc loin de la description d'une interface utilisateur.

4.11.3 Synthèse

De nombreux travaux de recherche ont été effectués sur l'adaptation, généralement sur les interfaces plastiques, c'est-à-dire des interfaces adaptables au contexte. Il existe plusieurs langages de spécification de ce type d'interface. Le tableau 4.11.1 est une compilation des résultats de notre recherche sur les différentes approches existantes. Ce tableau est bien évidemment non exhaustif et reprend pour chaque langage :

— Le nom du langage

— Le type d'interaction : indique le type d'interaction que supporte le langage. Il peut s'agir d'une interaction multimodale, une interaction tactile, une interaction via une interface graphique, etc.

— Le contexte : indique le type contexte supporté par le langage sur base des axes de Thevenin [65]. Il peut s'agir de l'utilisateur, l'environnement ou le terminal.

— Acteur : indique l'entité à qui déclenche l'adaptation. Il peut s'agir de l'utilisateur ou du système.

— Granularité : indique le niveau de l'adaptation. Ex. : adaptation de la présentation, de l'interaction, des taches, etc.

— La disponibilité :

Ce tableau nous montre que la majorité de ces langages proposent l'utilisateur et le système comme acteurs de l'adaptation. Cela veut dire que ces langages permettent de créer des interfaces qui sont adaptables et adaptatives. De plus, ces adaptations se font généralement par rapport à la plate-forme.

Il est important de signaler que ces approches offrent des moyens d'interaction limités et ne permettent pas d'accéder aux ressources des terminaux utilisés pour détecter leurs pro-

UIDL	Type d'interaction	Contexte	Acteur	Granularité	Disponibilité
UIML	GUI, Multimodal	Plateforme, Utilisateur	Système/Utilisateur	Présentation	Commercial - Virginia Tech. (Standard)
UsiXML	Interaction multimodal, GUI,CUI	Plateforme, Utilisateur	Système/Utilisateur	Présentation Interface abstraite Interface concrète	Recherches – UCL (Standard)
XMMVR	Interaction Multimodale dans un environnement de réalité virtuel	Utilisateur, Environnement	Système/Utilisateur	Taches abstraites	-
TeresaXML	GUI	Plateforme	Système/ Utilisateur	Taches abstraites	Recherche - HCI Group of ISTI-C.N.R
EMMA	Interaction Multimodale	Utilisateur	Utilisateur (Meta IHM)	-	-
SunML	GUI	Plateforme	Système	Taches abstraites	Recherches- université de Noce
XISL	Interaction Multimodale	Utilisateur	Utilisateur (Meta IHM)	Taches élémentaires	Public
X3D	Interaction Multimodale pour 3D	Utilisateur, Environnement	Utilisateur (Meta IHM)	Présentation Taches abstraites Taches élémentaires	Web3d -Public
XIML	GUI	Plateforme, Utilisateur	Système/ Utilisateur		W3C - Public
XForms	GUI	Plateforme	Système/ Utilisateur		Public (Standard)
MXML	GUI	Plateforme	Système/ Utilisateur		Propriétaire - Adobe Systems
SISL	Interaction Multimodale	Utilisateur	Utilisateur (Meta IHM)		
XAML	GUI	Plateforme	Système/ Utilisateur	Présentation	Propriétaire –Microsoft
XUL	GUI	Plateforme	Système/ Utilisateur	Présentation	Propriétaire - Mozilla
DISL	Interaction Multimodale pour Mobile	Utilisateur, Environnement, Plateforme	Système/Utilisateur		Recherche -Paderborn University
GIML	Interface Multimodale	Utilisateur	Système/Utilisateur		Recherche- Technical University of Dresden
ISML	GUI	Utilisateur, Plateforme	Système/Utilisateur		Recherche - Bournemouth University
RIML	GUI, Multimodal	Utilisateur, Environnement, Plateforme	Système/Utilisateur		Industry: SAP Research, IBM Germany, and Nokia Research Center CURE, UbiCall, and Fuijitsu Invia
SeescoaXML	GUI	Plateforme	Système/Utilisateur	Présentation Taches abstraites Taches élémentaires	Recherche - Expertise Centre for Digital Media Limburgs University Centrum
AUIML	GUI	Plateforme, Utilisateur	Système/Utilisateur	Présentation Taches abstraites Taches élémentaires	Propriétaire - IBM
XICL	GUI	Utilisateur	Système/Utilisateur		Recherche – Université de Rio Grande do Norte

TABLE 4.11.1 – Tableau des UIDL

priétés. Ces langages implémentent généralement une gestion implicite du contexte, ce qui entraîne la perte du lien entre le contexte et l'adaptation.

Il existe également des produits commerciaux pour l'aide à la création des interfaces utilisateurs comme JBuilder, Eclipse, Visual Studio de Microsoft... Ces outils sont puissants pour la création des interfaces normales, mais semblent limités dans l'adaptation d'interface, car ils ne permettent pas la génération complète du code et exigent un développement ad hoc pour chaque type de terminaux et chaque contexte. Ceci les rend peu utiles dans le domaine de la sensibilité au contexte où de nombreux types d'interfaces utilisateurs doivent être pris en compte.

L'utilisation des langages de description d'interface couplée avec un framework pour la gestion du contexte semble être la meilleure solution pour mettre en œuvre des systèmes(interfaces) sensibles au contexte.

-

Chapitre 5

Étude de cas

Une étude de cas a été réalisée afin de mettre en pratique les notions vues dans les chapitres précédents. L'état de l'art et l'étude des concepts clés des systèmes sensibles au contexte nous ont permis de définir les bases à partir desquelles nous allons appuyer notre travail de réflexion sur les choix que nous aurons à prendre durant l'implémentation de notre prototype. Le but est de réaliser une application qui intègre un module de gestion du contexte ainsi qu'un module de gestion d'adaptations en fonction des différents contextes modélisés.

Le prototype implémenté est une application qui servira de guide touristique à l'utilisateur, qui lui suggérera différents lieux à visiter en fonction des données sur le contexte courant, à savoir la position géographique, l'heure, la température et surtout les informations sur l'utilisateur même.

Dans la suite de ce document, nous essaierons de décrire notre solution, en détaillant son schéma de fonctionnement et son architecture. Nous aborderons également la réflexion sur le choix du modèle de gestion du contexte. Nous détaillerons ensuite les différentes adaptations au niveau de la partie cliente. Nous reprendrons enfin l'ensemble des étapes importantes sous forme de diagrammes de séquence.

5.1 Enquête contextuelle

La première phase de notre étude était l'analyse contextuelle. Cette analyse a été réalisée sur un ensemble de 10 personnes (6 hommes et 4 femmes) possédant toutes un smartphone et dont l'âge variait entre 21 et 35 ans. L'enquête consistait à un ensemble de questions autour de l'adaptation de l'application et de l'utilisation de données. Les questions étaient posées de manière ouverte afin de permettre aux interviewés de donner leurs avis et ainsi récolter leurs exigences en matière d'adaptation au contexte.

Après analyse des résultats de l'enquête, nous avons décidé de centrer notre modèle de contexte sur les données de l'utilisateur. Les détails de l'enquête sont disponibles dans l'annexe de ce document.

5.2 Fonctionnalités

L'objectif premier était de développer un module de gestion du contexte. Il s'agit de la partie la plus importante du travail et sur laquelle la majorité du temps a été passée. Ainsi les fonctionnalités clientes sont peu nombreuses, mais suffisantes pour illustrer le fonctionnement d'une application qui s'adapte au contexte. Ces fonctionnalités se résument principalement à :

— Une connexion à l'application grâce à un compte Facebook afin de transmettre les données personnelles et les préférences de l'utilisateur au serveur de contexte ;

— L'affichage d'un ensemble d'informations en rapport avec le contexte actuel de l'utilisateur. Ces informations sont principalement un ensemble de lieux suggérés en rapport avec le contexte, typiquement les préférences de l'utilisateur, le lieu et l'heure. L'application donne également des informations sur la météo du lieu actuel de l'utilisateur.

— La possibilité d'afficher la carte de la ville. Cette carte indique les lieux suggérés ainsi que les transports en commun à proximité de l'utilisateur pour s'y rendre.

— Traduire l'application en fonction de la langue de son téléphone ou de sa langue principale si elle est indiquée sur son profil Facebook.

5.2.1 Personas

1. Arthur Banisme

   Âge : 35 ans

   Profession : Cadre

   Situation familiale : Arthur Banisme est marié et a un enfant.

   Niveau informatique : limité aux logiciels de bureautique, mails et certains réseaux sociaux sur ordinateur. Il possède un smartphone.

   Caractéristiques : vit en zone rurale (Castillon), aime bien l'art et les voyages, et fait également du sport trois fois par semaine.

   But : Lorsqu'il visite une ville donnée, Arthur souhaite connaître les musées, les salles de sport et éventuellement les zoos pour y emmener son fils de 8 ans.

2. Arnaud Al

   Âge : 19 ans

   Profession : étudiant

   Situation familiale : Célibataire

   Niveau informatique : Possède un smartphone et est très familier avec la technologie et les réseaux sociaux.

Caractéristiques : vit en zone urbaine (Liège), aime la musique, le football. Arnaud est plutôt extraverti et aime sortir avec ses amis les week-ends.

But : Lorsqu'il se trouve dans une autre ville, Arnaud souhaite connaître les bons plans et les lieux d'intérêts à proximité.

5.3 Architecture

L'architecture proposée se base sur l'approche serveur et l'approche Blackboard que nous avons vue dans la section 3.2. En effet, la complexité des traitements nécessaires à l'obtention d'une information du contexte nous a emmenés à séparer l'application en soi, de la gestion du contexte. Le contexte est géré par un serveur de contexte qui se présente comme une boite noire pour l'application cliente. Cette dernière ne fait que l'interroger pour savoir dans quel contexte on se trouve.

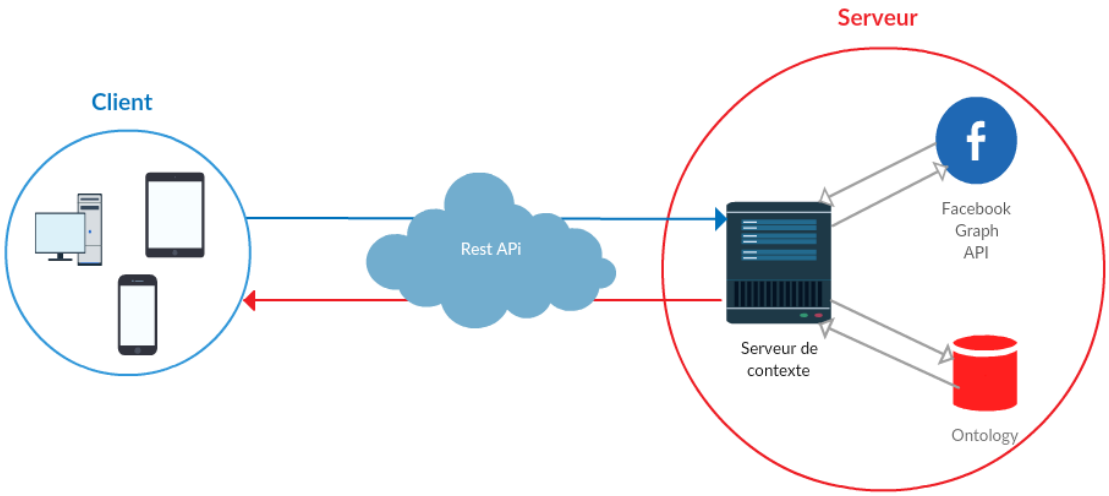


FIGURE 5.3.1 – Architecture

Le principal avantage de cette approche est qu'on arrive à alléger considérablement le client et à avoir de meilleures performances pour le traitement du contexte.

Les points suivants permettront de passer à la loupe les différentes parties qui composent notre solution.

5.3.1 Le serveur

La figure 5.3.1 montre la structure générale du fonctionnement de la solution. Elle présente la communication entre la partie cliente et la partie serveur. Dans cette section, nous allons aborder en détail le fonctionnement de la partie serveur.

Lorsque le serveur reçoit la requête du client, il lance le processus de traitement du contexte. Ce processus se base sur les couches abstraites de Dey [23]. Avant de présenter en détail les traitements effectués dans chaque couche, nous allons d'abord bien préciser le contexte qui sera exploité par le serveur.

Le contexte

Pour rappel, le contexte peut être vu comme étant le triplet <Utilisateur, terminal, Environnement> [65]. De manière générale, les informations concernant le terminal et l'environnement sont les aspects du contexte les plus souvent utilisés dans les travaux sur la gestion du contexte au détriment de l'aspect utilisateur.

Dans notre solution, nous avons opté pour une gestion du contexte centrée sur l' utilisateur. On parle alors de contexte social. Le contexte social désigne tout simplement l'environnement de l'utilisateur, c'est-à-dire tout ce qui l'entoure et qui influence son comportement ou ses préférences [37]. Typiquement ses relations et contacts, son humeur, ses préférences, ses habitudes, sa culture, etc.

Le défi était de pouvoir collecter un maximum d'informations sur l'utilisateur. Contrairement aux informations sur l'environnement et sur le terminal, les informations sur l'utilisateur ne sont pas capturées par des capteurs physiques. Ces informations sont généralement capturées de manière suivante :

— Elles peuvent être spécifiées directement par l'utilisateur à l'aide des formulaires d'inscription ou d'un menu dans lequel l'utilisateur peut ajouter ou éditer ses préférences.

— Elles peuvent être capturées, à partir des interactions de l'utilisateur, par des composants informatiques appelés aussi capteurs logiques.

Dans notre solution, nous avons opté pour l'utilisation d'un capteur virtuel, car imposer à l'utilisateur de spécifier toutes les informations le concernant peut devenir très vite fastidieux. La question qui se pose est donc de trouver un moyen de récupérer les informations de l'utilisateur sans lui demander de les spécifier lui même.

Ces dernières années, les réseaux sociaux en ligne ont envahi la toile en s'imposant comme une norme dans la société actuelle. Le plus connu d'entre eux est probablement Facebook dont le nombre d'utilisateurs grandit d' une manière exponentielle au fil des années. Ce service permet aux personnes de créer des liens virtuels avec d'autres utilisateurs, de maintenir des relations, d'indiquer les activités réalisées... de créer une véritable identité en ligne. Vu la popularité de ce réseau social et le grand nombre des utilisateurs, il est primordial d'étudier sa structure et son fonctionnement.

L'analyse de la structure de Facebook peut nous conduire à des résultats intéressants puisqu'outre l'aspect relationnel, Facebook est une source très riche en informations personnelles : nom, prénom, date de naissance, email, adresse, date de naissance, école, employeur, état matrimonial, orientation politique, etc. L'aspect le plus intéressant de Facebook est qu'il peut représenter une image plus ou moins fidèle de la véritable personnalité de l'utilisateur, de ses préférences et de ses relations sociales [5] [6]. Typiquement, deux personnes interconnectées sur Facebook ont généralement :

— des liens familiaux
— des amis ou des connaissances en commun
— des passions ou des opinions politiques en communs
— un même lieu de travail, une même école, etc.

Lorsqu'une personne aime les pages Facebook de plusieurs équipes de football, cela implique qu'il aime réellement le football. On peut donc récupérer une grande quantité d'informations fiables sur le profil Facebook. De ce fait, Facebook est considéré dans notre solution comme étant la première source d'information sur l'utilisateur. Au lieu de demander à l'utilisateur d'entrer les informations sur lui, nous demandons l'accès à son profil Facebook afin de modéliser le contexte et de lui fournir des informations adaptées à ses préférences.

L'accès aux données de l'utilisateur se fait grâce à l'API Graph, il s'agit de l'interface principale mise à disposition par Facebook par laquelle les applications peuvent lire et écrire sur le graphe social de Facebook [1] (Figure 5.3.2). Dans notre solution, nous nous contenterons de lire le graphe social et de récupérer les informations importantes.

Dans les lignes qui suivent, nous allons voir comment sont récupérées ces informations qui formeront le contexte.

1. Représentation, sous forme de graphe, de relations constituée de l'ensemble des membres Facebook et des éléments pour lesquels ils ont déclaré un intérêt (amis, entreprises, évènements, lieux, etc..).

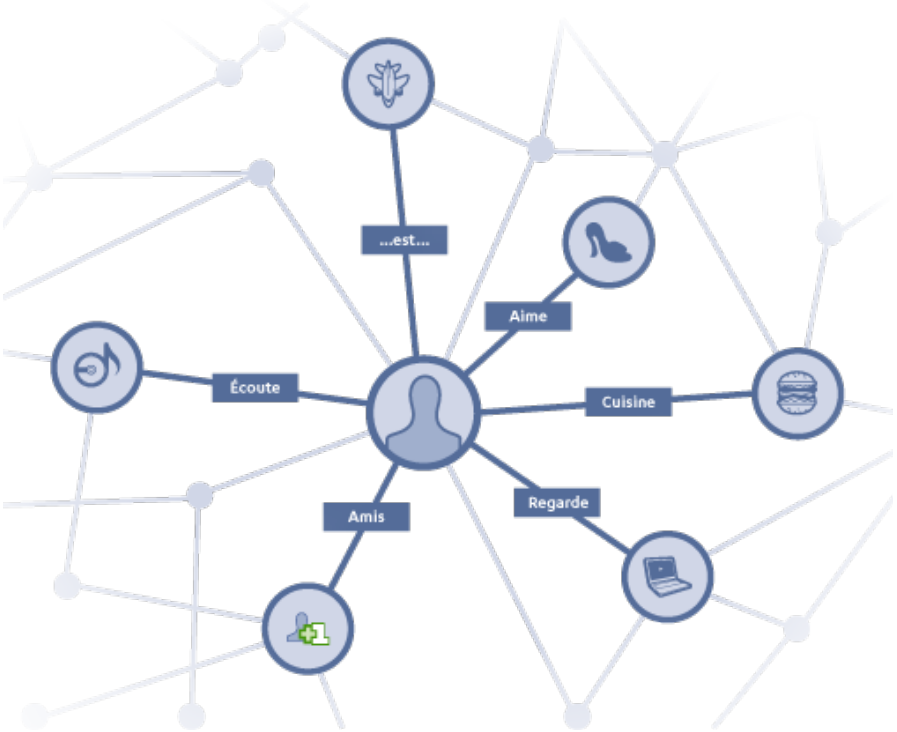


FIGURE 5.3.2 – Graphe social Facebook

Couche de capture

Cette couche regroupe les méthodes de capture des informations sur le contexte actuel de l'utilisateur. Comme indiqué précédemment, dans ce travail nous nous sommes plus concentrés sur l'aspect utilisateur du contexte. La couche de capture se résume donc à un ensemble de méthodes qui font appel à l'API Graph afin de récupérer un maximum d'informations sur l'utilisateur.

Pour lire ces informations, il faut effectuer un appel aux méthodes de l'API Graph accompagné d'une clé d'accès(acces token). Il s'agit des requêtes REST, qui renvoient l'objet demandé au format JSON.

http :/www.blog-nouvelles-technologies.fr/wp-content/uploads/2016/04/open-graph-est-toujours-impressionnant-pour-les-applications-1.png

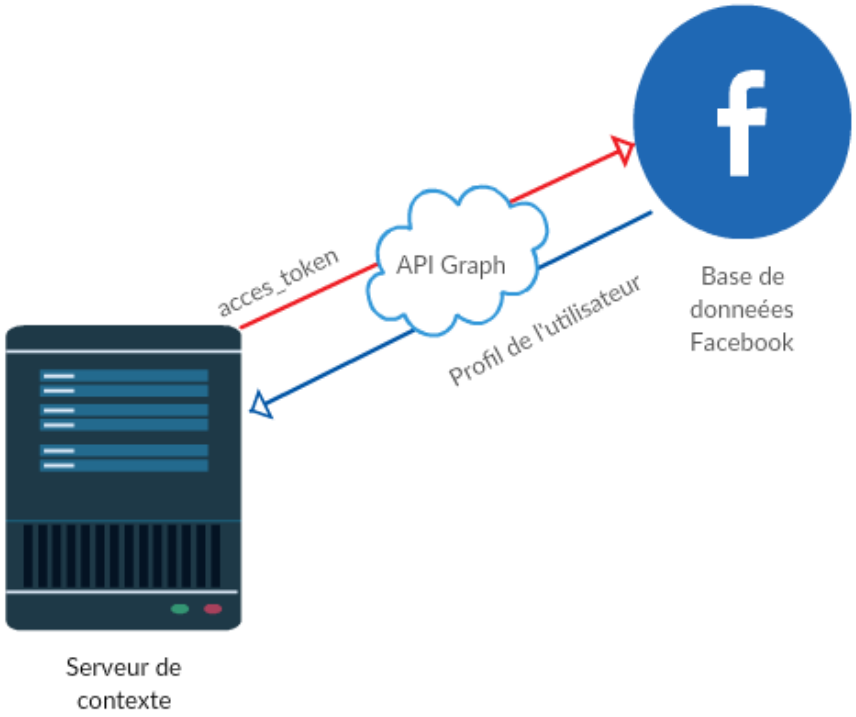


FIGURE 5.3.3 – Récupération des données de l'utilisateur

Le tableau 5.3.1 résume les différentes requêtes de l'API que nous avons utilisées dans notre programme et les objets renvoyés. Ces appels de l'API sont encapsulés dans les méthodes JAVA qui exécutent un « parsing » de l'objet JSON puis le place dans un objet JAVA.

Couche d'interprétation et de mise en forme de données

Les méthodes de cette couche ont pour but de traiter et d'interpréter les données. Les données récupérées par la couche de capture ne sont pas toujours utilisables. Par exemple, les données sur la localisation de l'utilisateur ne sont pas assez détaillées. En effet, l'API Graph renvoie uniquement un string contenant le nom du lieu et un identifiant. La couche d'interprétation va donc récupérer les détails sur ce lieu tels que la latitude, la longitude, etc.

Dans d'autres cas, l'API Graph peut renvoyer trop d'information à propos d'un objet. Certaines informations ne sont pas nécessaires pour la gestion du contexte. La couche d'interprétation a pour but cette fois-ci de supprimer le surplus d'informations afin de ne garder

Objet de retour	Requêtes REST	Description
User	GET v2.7/{user-id}	récupère les informations personnelles de l'utilisateur.
List <User>	GET v2.7/{user-id} /friends	récupère la liste d'amis de l'utilisateur .
List <Group>	GET v2.7/{user-id} /groups	récupère la liste des groupes auxquels l'utilisateur a adhéré.
List <Page>	GET me/likes	récupère la liste des pages que l'utilisateur a aimées.
List <Photo>	GET me/photos	récupère la liste des photos de l'utilisateur.
List <Video>	GET me/tagged	récupère la liste des vidéos de l'utilisateur.
List <Event>	GET v2.7/{user-id} /events	récupère la liste des événements auxquels l'utilisateur a participé.
List <User>	GET v2.7/{user-id} /family	récupère la liste de personnes qui ont un lien de famille avec l'utilisateur.
Liste <PlaceTag>	GET v2.7/{user-id} /tagged_places	récupère la liste des lieux visités par l'utilisateur.
List <Post>	GET v2.7/{user-id} /feed	récupère la liste des postes de l'utilisateur.
Object	GET v2.7/id	récupère n'importe quel type objet grâce à son identifiant .

TABLE 5.3.1 – Tableau des requêtes de l'API

que les informations pertinentes pour le serveur de contexte.

Nous avons donc créé nos propres classes (Figure 5.3.4) qui ne contiennent que les champs dont on aura besoin dans notre gestion du contexte.

Cette couche s'occupe également des traitements simples tels que le calcul de l'âge à partir de la date d'anniversaire de l'utilisateur ou encore la conversion de type (du type string au type int par exemple).

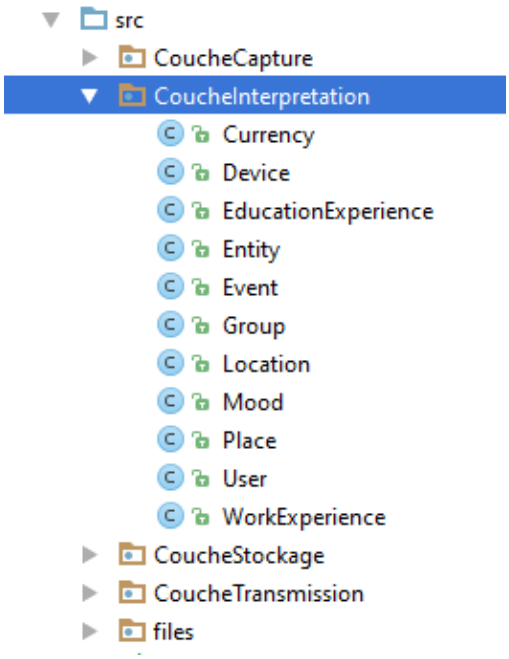


FIGURE 5.3.4 – Classe de la couche d'interprétation

Couche de stockage et de modélisation

Une fois les informations capturées et mises en forme, il faut évidemment les stocker. C'est l'objectif principal de cette couche.

Pour stocker les informations, il faut définir un modèle de données afin de rendre l'information utilisable par un programme informatique. Un modèle doit au minimum permettre de stocker pour chaque élément du contexte :

— Un type qui indique la catégorie de l'élément. Ex. : localisation, température, etc.

— Une valeur qui indique tout simplement un chiffre accompagné d'une unité qui représente la donnée même. Ex. : «Liège, Belgique», 12˚C, etc.

Dans la section 2.4.3, nous avons parcouru les différents modèles de contexte que l'on retrouve dans la littérature. On peut se demander alors quel modèle choisir pour notre application.

Dans notre solution, nous avons choisi de modéliser le contexte par une ontologie pour les raisons suivantes :

- L'expressivité. Les ontologies permettent, grâce à leur richesse d'expression, de décrire de manière détaillée les éléments du contexte. Ainsi, lors de la modélisation, elles permettent de maintenir la sémantique du graphe social. Par exemple, la relation «user1 a pour athlète favori Eden Hazard» sur le graphe social d'un utilisateur, est décrite par le lien :

  user1 —-favorite_athlete—-> Eden Hazard

  Sur le graphe social, cette relation implique que l'utilisateur aime la page du footballeur Eden Hazard.

  user1 —-like—-> Eden Hazard

  Ceci exprime le fait qu'il y a une relation de dépendance entre ces liens. Si l'utilisateur a un lien **favorite_athlete** vers une entité, alors il a également un lien *like* vers cette même entité. Ce genre de relation «Si relation X alors relation Y» n'est pas toujours facile à modéliser avec les autres approches de modélisation. Grâce au concept de «property» et de «sub-property», les ontologies OWL permettent de modéliser facilement ce cas en déclarant le lien *favorite_athlete* comme étant une sous-propriété de *like*. Ainsi un lien *favorite_athlete* sera également considéré comme un lien *like*.
- Formalisme. Une ontologie est constituée d'un vocabulaire conceptuel, c'est-à-dire un ensemble de termes dotés d'un statut «formel» précis (classe, propriété...). Ce vocabulaire est souvent organisé en une hiérarchie spécifiant une relation de spécialisation/-généralisation.

- Raisonnement et validation. Le choix de l'approche ontologique dans la modélisation du contexte était motivé également par la possibilité d'appliquer du raisonnement sur le modèle. Grâce au raisonneur, on peut facilement vérifier la cohérence de notre ontologie du contexte et inférer de nouveaux types et de nouvelles propriétés des individus. Ce qui permet de déduire un contexte de plus haut nouveau à partir des données brutes. On peut ainsi inférer différents profils d'utilisateur. Par exemple, on peut, à partir de certaines règles basées sur les pages que l'utilisateur a aimées, déduire si l'utili-

sateur est un fan de sport, fan de musique, fan d'art, etc.
On distingue :

- Le raisonnement ontologique, qui est exécuté par des règles intégrées dans la sémantique des langages OWL et RDF(s). Il permet de déduire certains faits relatifs à la hiérarchie de l'ontologie : vérifier si une classe est une sous-classe d' une autre classe ou si un individu (une instance) placé dans une classe appartient à d'autres classes en même temps.
- Les règles d' inférence qui sont définies par le développeur et qui permettent de déduire des informations de plus haut niveau.

- Monde ouvert. Contrairement aux autres approches qui sont basées sur un monde fermé [2], l'ontologie ne prétend pas détenir toute l'information sur un objet. Si une information est manquante, le raisonneur de l'ontologie ne renverra pas de réponse à propos de l'information demandée, car il ignore tout simplement si elle est vraie ou fausse. Ce qui correspond bien au traitement d'une information complexe et généralement incomplète comme le contexte.

Cependant, construire une ontologie correcte et cohérente n'est pas toujours une tâche facile. Plus le domaine à modéliser est grand, plus la construction et le traitement de l'ontologie deviennent complexes. Une bonne ontologie de contexte doit respecter les exigences suivantes [40] :

- Simplicité : les expressions et les relations utilisées doivent être assez simples afin de faciliter la compréhension et l'utilisation de l'ontologie aux développeurs.

- flexibilité et extensibilité : l'ontologie construite doit permettre l'ajout d'un nouvel élément du contexte ou d'une nouvelle relation entre éléments du contexte.

- généricité : l'ontologie ne doit pas être fortement couplée avec l'application ou avec un certain type de contexte. Elle doit pouvoir être utilisée dans une autre application.

- expressivité : l'ontologie devra décrire le contexte et ses éléments de la manière la plus claire possible et sans incohérence.

2. Dans un monde fermé, on considère que l'information que l'on possède est complète

Pour construire notre ontologie et faire des simulations, nous avons utilisé le logiciel protégé [3]. Il est installé avec plusieurs raisonneurs qui permettent d'écrire et de tester les différentes règles d'inférences ainsi que le raisonnement ontologique.

Notre ontologie se base sur les données du graphe social de l'utilisateur et présente les classes reprises dans les tableaux 5.3.2 et 5.3.3

Les classes sont connectées entre elles par des liens appelés «Object property». Ces liens sont orientés. La classe de départ est appelée «domain» et la classe cible est appelée «range». La sémantique cachée derrière ce type de lien est que le nom du lien désigne une propriété non atomique de la classe «domain». Cette propriété est du type de la classe «range».

Les «Object Properties» peuvent être hiérarchisés. On parle de «SuperProperty» et de «SubProperty». Lorsque A est une «SubProperty» de B, toute classe C possédant une propriété A possédera également une propriété B.

Dans notre modèle, les propriétés *music*, *inspirational_people*, *favorite_team*, *favorite_athlete* sont des «SubProperty» de *like*. Ce choix est motivé par le fait que sur Facebook, l'équipe favorite de l'utilisateur est une équipe dont l'utilisateur a «liké» la page. En plus du lien "*favorite_team*», il y a donc un lien "*like*» implicite. La justification est la même pour *music*, *inspirational_people*, et *favorite_athlete*

Cette hiérarchie existe également dans les classes. Si A est une sous-classe de B alors tout individu de la classe A est un individu de la classe B. Ces sous-classes ont pour but de spécialiser leur superclasse. Ce mécanisme permet d'attribut à un individu une classe beaucoup plus spécifique.

Il existe certaines classes, non reprises dans les tableaux 5.3.2 et 5.3.3, dont les définitions se font au moyen de règles. Ces classes sont prises en compte uniquement après avoir lancé le raisonneur. Ces classes permettent de récupérer un contexte de plus haut niveau. Dans notre modèle, ce mécanisme a été utilisé pour définir des informations de plus haut niveau ou types d'utilisateur. Le tableau 5.3.4 reprend ces contextes hauts d'utilisateur ainsi que leurs règles de définition.

3. http ://protege.stanford.edu/

Classe	Description	Data property(Champs atomiques)
User	Il s'agit de la classe centrale de notre modèle. Elle représente les informations personnelles de l'utilisateur.	— id : permets d'identifier l'objet – address : indique son adresse – âge : indique l'âge de l'utilisateur – birthday : indique sa date d'anniversaire – currency : indique la devise du lieu de résidence de l'utilisateur. – email : contient l'adresse mail électronique de l'utilisateur – gender : contient le sexe de l'utilisateur – has_children : indique si l'utilisateur a des enfants ou pas. – is_Extrovert : indique si l'utilisateur est extraverti ou pas – languages : indique les langues parlées par l'utilisateur – name : indique le nom de l'utilisateur – nb_friend : indique le nombre d'amis de l'utilisateur – religion : indique la religion de l'utilisateur – updated_time : indique la date de dernière mise à jour du profil de l'utilisateur
Event	Elle représente les différents événements auxquels un utilisateur participe ou a participé	— interested_count : indique le nombre de personnes intéressé par l'évènement. - attending_count : indique le nombre de personnes attendues – start_time : indique la date du début de l'évènement – end_time : indique la date de fin de l'évènement
Entity	Elle représente les différentes entités du graphe social. Il s'agit des pages publiques Facebook	— id : permet d'identifier l'objet sur l'API Graph – name : indique le nom de la page – description : indique la description de la page – type : indique la catégorie de la page(musique, art, sport, etc.)
Mood	Elle représente l'humeur de l'utilisateur	— name : indique le nom associé à l'humeur – time : indique l'heure à laquelle l'humeur a été détectée

TABLE 5.3.2 – classes de l'ontologie 1

Classe	Description	Data property(Champs atomiques)
Device	Elle représente le périphérique utilisé par l'utilisateur	— hardware : indique la marque du périphérique utilisé par l'utilisateur. – os : indique le nom du système d'exploitation du périphérique
Place	Elle représente un lieu visité par l'utilisateur	— id : permet d'identifier l'objet sur l'API Graph – name : indique le nom du lieu – overall_rating : indique le nombre de Like ou d'avis positif – time : indique l'heure de visite
Location	Elle représente une localisation donnée	— city : indique la ville – country : indique le pays – currency : indique la devise – latitude : indique la latitude – longitude : indique la longitude – name : indique le nom – region : indique la région – state : indique l'état – street : indique la rue – température : indique la température dans cette zone – zip : indique le code postal
Groupe	Elle représente un groupe dont l'utilisateur est membre	— id : permet d'identifier l'objet sur l'API Graph – name : indique le nom du groupe – description : indique la description du groupe
Experience	Elle représente une expérience de l'utilisateur. Il peut s'agir d'une expérience professionnelle ou un parcours scolaire. Cette classe est la superclasse des classes EducationExperience et WorkExperience.	
EducationExperience	Elle représente une expérience scolaire de l'utilisateur.	— name : indique le nom de l'expérience – type : indique le type de parcours (Universitaire, secondaire, etc.) – year : indique l'année du parcours
WorkExperience	Elle représente une expérience professionnelle de l'utilisateur.	- end_date : indique la date de fin de l'expérience – position : indique la position occupée pendante l'expérience start_date : indique la date de début de l'expérience

TABLE 5.3.3 – classes de l'ontologie 2

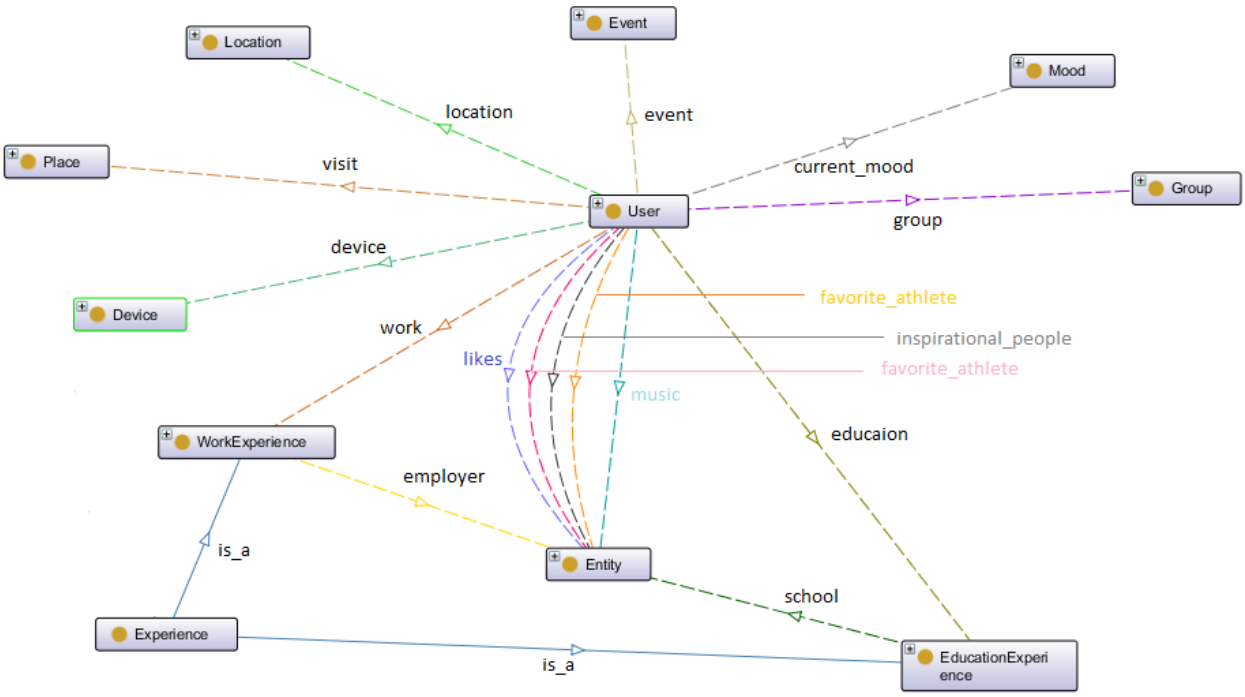


FIGURE 5.3.5 – Visualisations des classes et leurs liens «property»

Classe	Règles	Description
ArtEnthusiast	User and (likes some Museum_Art)	utilisateur fan d'art.
MusicEnthisiast	User and (music min 8 Entity)	Utilisateur fan de musique
SportEnthusiast	User and ((favorite_athlete some Entity) or (favorite_team some Entity))	Utilisateur fan de sport

TABLE 5.3.4 – Contexte haut niveau - Profil utilisateur

Couche de transmission

Une fois le contexte modélisé et stocké, il faut le rendre accessible via une interface publique pour le client. Les clients peuvent y accéder de manière synchrone ou asynchrone.

Dans notre modèle, nous avons utilisé une architecture REST, l'accès au contexte se fait par un appel synchrone aux services Web. Après traitement, le serveur renvoie le contexte dans un objet avec toutes les informations sur l'utilisateur. Cet objet est au format JSON.

L'avantage de cette approche est que cela rend la couche application indépendante de la technologie utilisée dans les couches précédentes. Ce qui permet de développer des clients de tous types en utilisant la technologie de son choix. Il suffira de faire un appel REST et de «parser» l'objet (le contexte) JSON renvoyé.

5.3.2 Le client

La couche application regroupe les différents clients qui feront appel au serveur de contexte. Dans ce travail, nous n'avons développé qu'un seul client afin de tester notre serveur de contexte. Il s'agit d'une application Windows universelle c'est-à-dire une application qui tourne aussi bien sur un ordinateur que sur un téléphone mobile Windows 1 0(figure 5.3.6 et figure 5.3.7). L'application s'adapte donc à la plate-forme qui l'accueille. Le secret derrière cette adaptation est l'utilisation du langage de description d'interface XAML que l'on a présenté dans l'état de l'art sur les outils pour l'adaptation des interfaces à la section 4.11.2.

Pour la partie traitement, nous avons opté pour le langage c# qui est le langage phare de Microsoft et qui est généralement utilisé dans le développement des applications Windows.

Le choix de cette technologie est motivé par :

— la possibilité de développer une application qui tourne à la fois sur un ordinateur et sur un terminal mobile.

— le désir d'apprendre une nouvelle technologie.

— la puissance des outils de développement offerts par Microsoft, notamment Visual Studio, qui est un IDE qui permet de faire des développements rapides et de construire de belles interfaces graphiques sans être un expert.

L'application client développée se nomme «My Guide» ou «Mon Guide» en français. L'application va utiliser le contexte actuel de l'utilisateur et va lui suggérer un ensemble de lieux.

Après connexion avec le compte Facebook, le client lance une requête au serveur de contexte afin de récupérer le contexte. Une fois le contexte reçu, il sera enrichi avec des informations fournies par le terminal de l'utilisateur telles que l'heure, la météo et surtout la localisation. La localisation fournie par le terminal sera toujours préférée à la localisation via Facebook, car la localisation du terminal est beaucoup plus précise. On obtient ainsi un nouveau contexte composé des informations sur l'utilisateur, l'heure, la météo et la localisation. En fonction de ce contexte, plusieurs adaptations ont été mises en place.

La suggestion de lieux

La suggestion de lieux se fait grâce à l'API Google Places. En fonction du contexte, le client lance des requêtes différentes afin d'obtenir des lieux qui correspondent le mieux à l'utilisa-

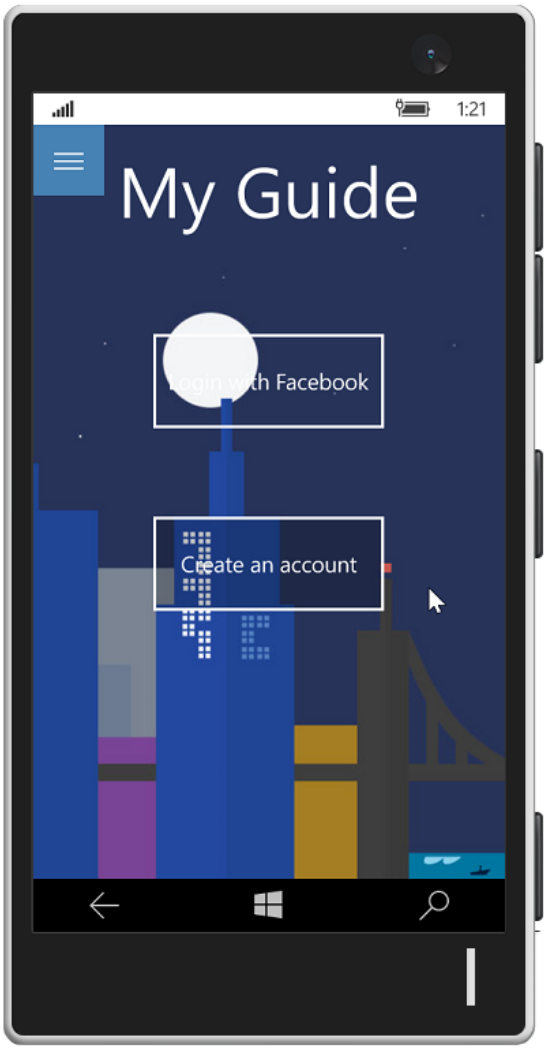


FIGURE 5.3.6 – Application sur la version smartphone

teur et à son contexte.

— Suggestions des lieux en fonction de l'heure : s' il est entre 12 h et 15 h l'application, suggère à l'utilisateur des restaurants à proximité du lieu où il se trouve.

— Suggestion des lieux en fonction du profil de l'utilisateur et de l'heure.

- Si l'utilisateur est fan d'art et que l'on est en journée, l'application suggère les galeries d'art et les musées à proximité.
- Si par contre, on est en soirée et que l'utilisateur est majeur, l'application peut suggérer des lieux de sortie tels que des boites de nuits, des cafés, des bars, ci-

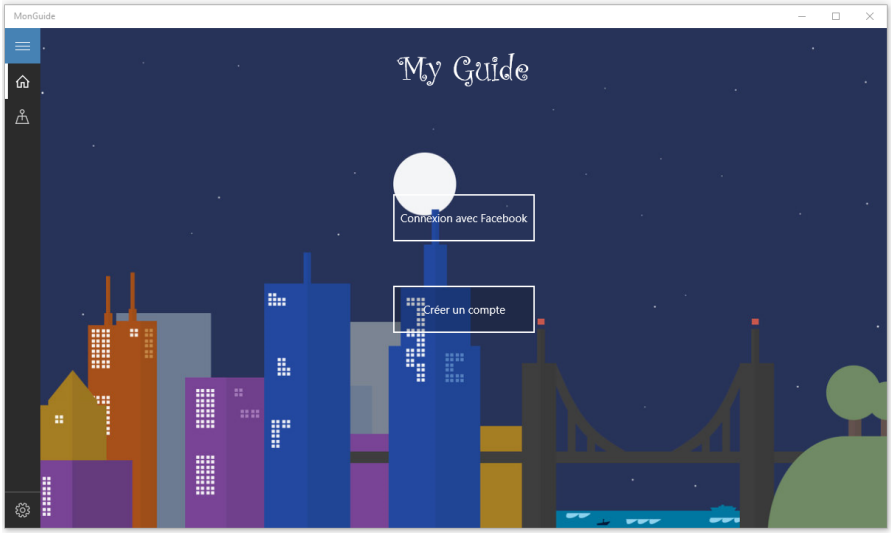


FIGURE 5.3.7 – Application sur la version ordinateur

néma, etc.

Plusieurs suggestions peuvent être faites en fonctions de différentes informations sur l'utilisateur. Quel que soit le cas, les lieux suggérés sont toujours à proximité du lieu actuel de l'utilisateur. En cliquant sur une suggestion ou sur le bouton «Map», l'application permet de visualiser ces lieux sur une carte.

Adaptation de l'interface graphique

Au-delà du fait que l'interface s'adapte au type de terminal, elle s'adapte également en fonction de :

— La langue du terminal de l'utilisateur : l'application est traduite automatiquement dans la langue cible (langue de l'utilisateur). Cette traduction se fait grâce au traducteur intégré au terminal Windows.

— L'heure : l'application a un thème "Jour" avec un fond clair en journée et bascule dans un thème plus foncé appelé thème "Nuit" entre 18 h et 5h.

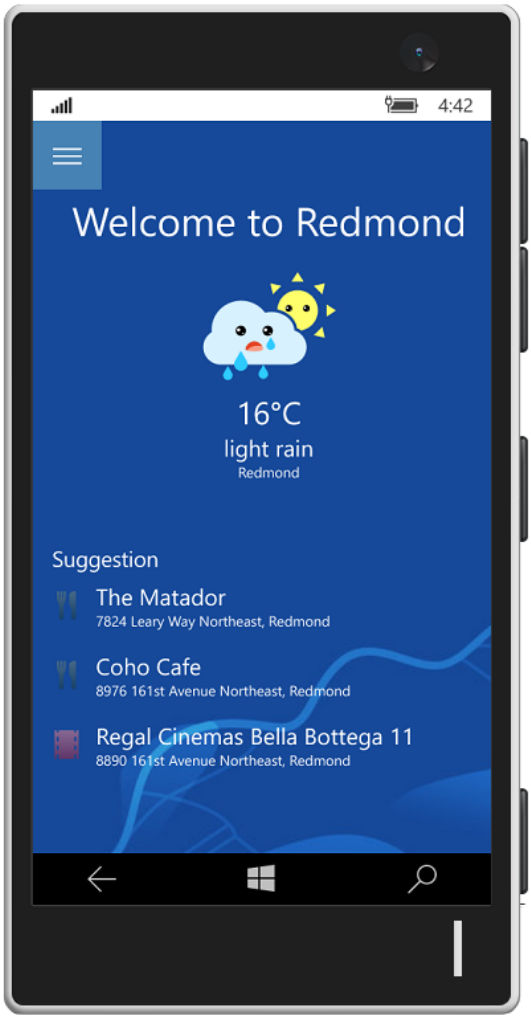


FIGURE 5.3.8 – Suggestions des lieux en soirée et adaptation de la langue en fonction de la langue de l'utilisateur

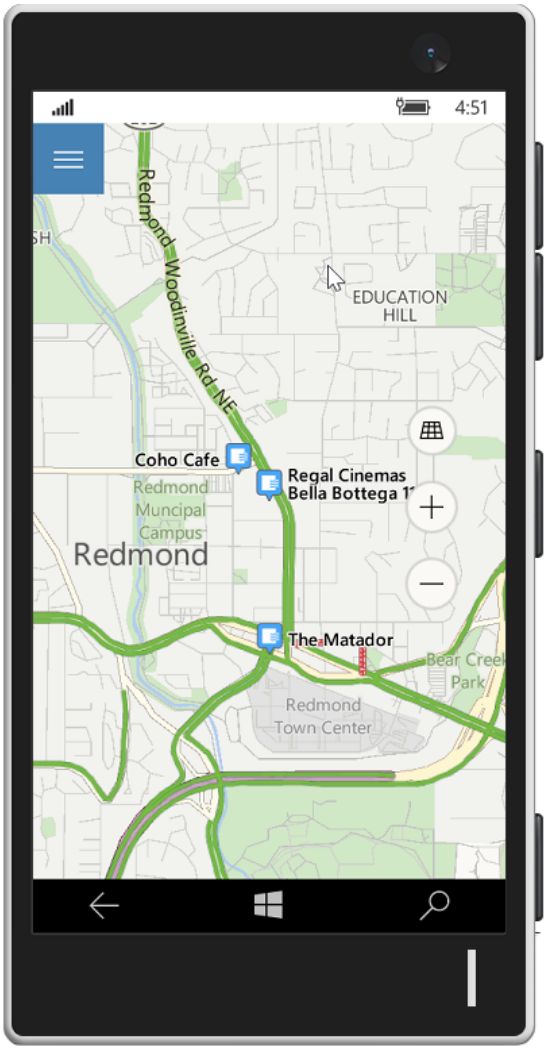


FIGURE 5.3.9 – Visualisation des lieux sur une carte

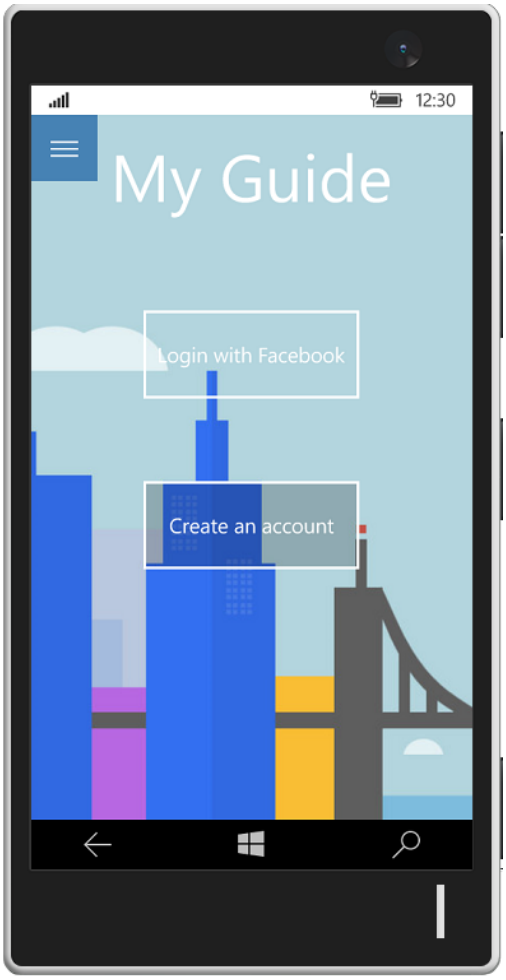


FIGURE 5.3.10 – Thème "Jour"

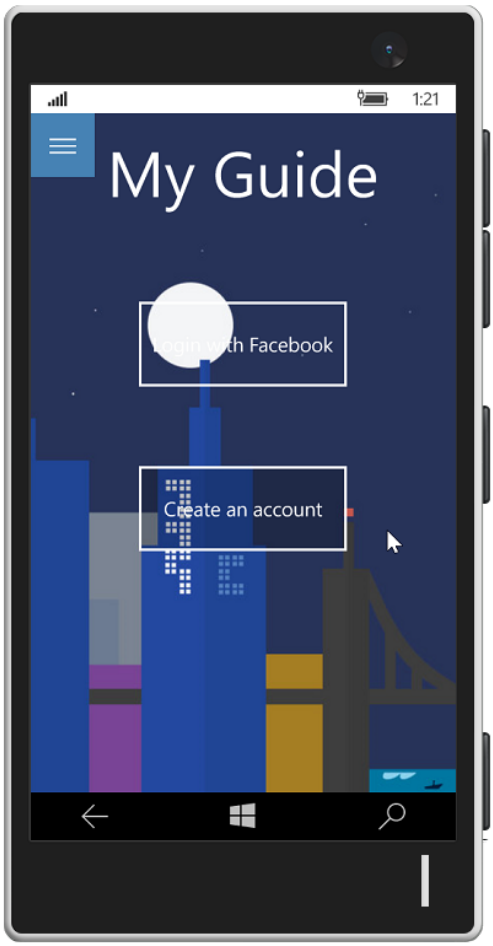


FIGURE 5.3.11 – Thème "Nuit"

Chapitre 6

Conclusion

Plusieurs recherches ont été menées sur les systèmes sensibles au contexte. Ce thème continu toujours a attiré un grand nombre de chercheurs, car plus que jamais, nous vivons l'ère de l'informatique ubiquitaire. Les applications actuelles et futures doivent être capables de gérer cette notion de contexte. Le développement de ce genre d'application nécessite de bien comprendre dans un premier temps les concepts de base des systèmes sensibles au contexte.

Dans le chapitre 2, nous avons essayé de cerner cette notion de contexte et de trouver une définition. Malgré les nombreuses recherches, il est difficile de trouver un consensus, et la définition même du contexte varie d'un auteur à l'autre. De manière générale, nous pouvons retenir que le contexte englobe toutes les informations autour de l'interaction entre l'application et l'utilisateur. Afin de pouvoir utiliser les informations du contexte, il est important de les modéliser. Dans la section 2.4.3 nous avons parcouru les différentes approches de modélisation du contexte existantes avec leurs avantages et leurs inconvénients. La modélisation du contexte est la clé de tout système dépendant du contexte. Le choix du modèle est donc d'une importance capitale. Une bonne modélisation du contexte facilite la mise en place des mécanismes d'adaptation. L'adaptation peut se faire au niveau du contenu de l'application et/ou au niveau de l'interface utilisateur de l'application.

6.1 Bilan

Malgré l'existence de diverses approches en adaptation du contexte, la plupart d'entre elles présentent plusieurs limites dont la principale se situe au niveau de la modélisation du contexte. Les deux principaux problèmes au niveau de la modélisation du contexte sont :

— Le manque de généricité : les approches existantes proposent généralement une architecture complète avec une implémentation ad hoc de la gestion du contexte. Ainsi, l'utilisation de ces solutions se limite généralement à un domaine particulier (comme la sensibilité à la localisation par exemple) ou à une technologie particulière.

— L'oubli de l'utilisateur : l'évolution technologique est en constante progression, et avec l'avènement de l'internet des objets, il y a de plus en plus de capteurs puissants qui permettent de capturer de plus en plus d'informations dans l'environnement. Ainsi dans la gestion du contexte, les approches existantes ont tendance à plus s'adapter à l'environnement et au périphérique qu'à l'utilisateur même.

Dans la solution proposée, le but était de recentrer la gestion du contexte sur l'utilisateur, ses préférences, son humeur, etc. Ainsi, nous nous sommes basés sur le réseau social Facebook, qui demeure une grande source d'information sur l'utilisateur. Nous avons essayé d'utiliser le graphe social pour en tirer un maximum d'informations sur l'utilisateur, ses activités, ce qu'il aime, etc. Nous avons construit une ontologie qui modélise les différentes connexions autour de l'utilisateur.

Cette solution se veut être :

— Extensible et évolutive. L'un des grands avantages d'une modélisation ontologique est la facilité d'extension. On peut facilement ajouter des nouveaux concepts dans l'ontologie ou combiné deux ontologies pour en faire une plus grande. Étant donné que notre modèle est centré sur l'utilisateur, on peut facilement imaginer le combiner avec une ontologie qui reprend les informations sur le terminal de l'utilisateur ainsi que l'environnement.

  L'utilisation d'une approche en couche facilite l'évolution de notre architecture. Ainsi on peut facilement changer la méthode de capture des informations du graphe social (couche de capture) sans impacter les couches supérieures.

— Indépendante d'une technologie particulière pour le client. La récupération du contexte

se fait par un appel au service Web. La majorité des langages de programmation permettent de réaliser ce genre d'appels à un service Web REST.

— Centré sur l'utilisateur : comme dit plus haut, l'ontologie est basée sur les informations du profil de l'utilisateur.

Ce travail était une occasion de travailler sur un sujet de l'heure, de comprendre la notion de contexte et la complexité qui se cache derrière l'adaptation au contexte. Ce fut également une occasion d'apprendre plusieurs technologies telles que UWP (Universal Windows Plateforme), XAML, C#, OWL, etc.

6.2 Travaux futurs

L'objectif principal du travail n'était pas de développer une application sophistiquée, mais de développer un petit prototype en se focalisant sur la modélisation du contexte, précisément sur les informations de l'utilisateur, les moyens de capture de ces informations et leur modélisation. Ainsi la solution développée dans ce travail est perfectible, car par fautes de temps et de moyens, elle présente beaucoup de limites :

— Le serveur de contexte n'est pas optimisé, il met un petit temps avant de renvoyer la réponse au client. Ceci est dû à la taille de l'ontologie et au moteur d'inférence de la libraire JENA qui met du temps dans le traitement des différentes inférences.
— Pour des raisons de taille, nous n'avons pas pu reprendre tous les éléments de l'API Graph dans notre modèle ontologique, car la modélisation et le traitement devenaient très vite complexes.
— Notre solution ne définit que trois types d'utilisateurs. L'utilisateur fan de sport, l'utilisateur fan d'art, et l'utilisateur fan de musique.
— Comme déjà signalé, notre modèle de contexte se focalise sur l'aspect «utilisateur». L'aspect plate-forme est géré en adaptant uniquement l'interface avec XAML (Smartphone ou PC) . Quant aux informations sur l'environnement, elles ne sont pas prises en compte dans le modèle.
— Il manque un module de gestion d'historique de contexte.
— Les mécanismes d'adaptation sur l'application client sont pauvres et limités. Elle consiste en une adaptation assez simple de l'interface utilisateur, et une suggestion des lieux en fonction du type d'utilisateur, de sa localisation et de l'heure.
— Les informations récupérées sur le profil d'un utilisateur ne sont pas toujours fiables. Par exemple, il y a moyen de changer sa localisation actuelle ou encore le cas de certains utilisateurs qui mettent de fausses informations sur leur profil ou qui possèdent des profils incomplets.
— Que faire des utilisateurs qui n'ont pas de compte Facebook ?

Parmi les différentes fonctionnalités auxquelles nous avons réfléchi durant le développement du prototype, plusieurs d'entre elles n'ont pas été développées ou implémentées, par manque d'expérience dans le domaine du développement d'application sensible au contexte ou par manque de temps. Vous trouverez donc ci-dessous une série d'idées, de concepts qu'il serait encore possible d'intégrer pour améliorer la solution actuelle.

6.2.1 Que reste-t-il à faire ?

Partant de la version actuelle de la solution, à court terme, nous aurions pu :

— Ajouter plus de types d'utilisateurs que ceux traiter dans notre modèle.

— Intégrer dans notre modèle ontologique, les autres aspects du contexte à savoir les informations sur la plate-forme (elles sont récupérées juste à l'exécution) et les informations sur l'environnement.

— Ajouter un module de gestion d'historique de contexte qui sauvegardera les différents contextes par lesquels l'utilisateur est passé.

— Développer un module d'adaptation afin de pousser un peu plus l'adaptation du comportement de l'application cliente ainsi que l'interface graphique. Par exemple, on pourrait ajouter le facteur météo dans la suggestion des lieux, car proposer une sortie à la plage à un utilisateur alors qu'il pleut abondamment n'est pas très pertinent.

— Optimiser le serveur de contexte en ajoutant un système de cache qui permettra d'accélérer le temps de réponse.

— Faire valider l'application par Facebook afin d'avoir accès à plus d'informations sur l'utilisateur tels que son humeur ou les groupes dont il est administrateur ou simple membre. Ces informations peuvent être utiles pour affiner les suggestions.

6.2.2 Pour aller plus loin

Partant de la version actuelle de la solution, à long terme, on peut envisager :

— Mettre en place un module d'amélioration de l'adaptation. Ce module devra mettre en place un mécanisme pour obtenir le feedback de l'utilisateur afin de déterminer si l'adaptation faite est pertinente ou non pour l'utilisateur. Les adaptations non pertinentes ne seront plus proposées à l'utilisateur.

— Laisser le choix à l'utilisateur sur l'adaptation, surtout au niveau de l'interface graphique. Par exemple, arrivée en soirée, l'interface graphique passe dans le thème «Nuit». Il peut être utile de permettre à l'utilisateur de revenir dans le thème «jour» s'il le souhaite.

— Mettre en place un module de validation des informations du profil de l'utilisation. On peut par exemple proposer à l'utilisateur de les valider lui même ou diversifier les sources d'informations en intégrant d'autres réseaux sociaux. Par exemple, on peut utiliser le profil Linkedin ou Twitter de l'utilisateur.

— Inférer la personnalité de l'utilisateur à partir des informations de son profil. Des recherches ont été faites à ce sujet [41] [5] [28] [6]. En se basant sur les résultats de ces recherches, mettre en place des règles d'adaptation en fonction de la personnalité. Par exemple, proposer des sorties en soirée à des utilisateurs plutôt extraverties qu'introverties.

— Créer un langage permettant de définir des règles d'adaptation de l'interface et des fonctionnalités de l'application afin d'avoir un format unique pour toutes les règles et ainsi faciliter le developpement des systèmes adaptables.

Bibliographie

[1] D Abowd, Anind K Dey, Robert Orr, and Jason Brotherton. Context-awareness in wearable and ubiquitous computing. *Virtual Reality*, 3(3) :200–211, 1998.

[2] Marc Abrams, Constantinos Phanouriou, Alan L Batongbacal, Stephen M Williams, and Jonathan E Shuster. Uiml : an appliance-independent xml user interface language. *Computer Networks*, 31(11) :1695–1708, 1999.

[3] Pierre A Akiki, Arosha K Bandara, and Yijun Yu. Adaptive model-driven user interface development systems. *ACM Computing Surveys*, 47(1), 2015.

[4] Mir Farooq Ali, Manuel A Perez-Quinones, Marc Abrams, and Eric Shell. Building multiplatform user interfaces with uiml. In *Computer-Aided Design of User Interfaces III*, pages 255–266. Springer, 2002.

[5] Yoram Bachrach, Michal Kosinski, Thore Graepel, Pushmeet Kohli, and David Stillwell. Personality and patterns of facebook usage. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 24–32. ACM, 2012.

[6] Mitja D Back, Juliane M Stopfer, Simine Vazire, Sam Gaddis, Stefan C Schmukle, Boris Egloff, and Samuel D Gosling. Facebook profiles reflect actual personality, not self-idealization. *Psychological science*, 2010.

[7] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4) :263–277, 2007.

[8] Joseph Bauer, Ralf-Detlef Kutsche, and Rudiger Ehrmanntraut. Identification and modeling of contexts for different information scenarios in air traffic. *Technische Universität Berlin, Diplomarbeit*, 2003.

[9] Silvia Berti, Francesco Correani, Fabio Paterno, and Carmen Santoro. The teresa xml language for the description of interactive systems at multiple abstraction levels. In *Proceedings Workshop on Developing User Interfaces with XML : Advances on User Interface Description Languages*, pages 103–110, 2004.

[10] Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, and Daniele Riboni. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2) :161–180, 2010.

[11] Peter J Brown, John D Bovey, and Xian Chen. Context-aware applications : from the laboratory to the marketplace. *IEEE personal communications*, 4(5) :58–64, 1997.

[12] Sven Buchholz, Thomas Hamann, and Gerald Hubsch. Comprehensive structured context profiles (cscp) : Design and experiences. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 43–47. IEEE, 2004.

[13] William Buxton, MR Lamb, Dave Sherman, and Kenneth Carless Smith. Towards a comprehensive user interface management system. In *ACM SIGGRAPH Computer Graphics*, volume 17, pages 35–42. ACM, 1983.

[14] Gaëlle Calvary and Joëlle Coutaz. Plasticité des interfaces : une nécessité. *information-interaction-intelligence, Actes des deuxièmes Assises nationales du GDR I*, 3 :247–261, 2002.

[15] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt. A unifying reference framework for multi-target user interfaces. *Interacting with computers*, 15(3) :289–308, 2003.

[16] Gaëlle Calvary, Olfa Dâassi, Joëlle Coutaz, and Alexandre Demeure. Des widgets aux comets pour la plasticité des systèmes interactifs. *Revue d'Interaction Homme Machine, Europia*, 6(1), 2005.

[17] Tarak Chaari. *Adaptation d'applications pervasives dans des environnements multi-contextes*. PhD thesis, INSA de Lyon, 2007.

[18] Christophe Chassot, Karim Guennoun, Khalil Drira, Francois Armando, Ernesto Exposito, and André Lozes. Towards autonomous management of qos through model-driven adaptability in communication-centric systems. *ITSSA*, 2(3) :255–264, 2006.

[19] Harry Chen, Tim Finin, and Anupam Joshi. An ontology for context-aware pervasive computing environments. *The knowledge engineering review*, 18(03) :197–207, 2003.

[20] Keith Cheverst, Keith Mitchell, and Nigel Davies. Design of an object model for a context sensitive tourist guide. *Computers & Graphics*, 23(6) :883–891, 1999.

[21] D Clark. From abstract to concrete : designing auiml renderers, 2000.

[22] Anind K Dey. Context-aware computing : The cyberdesk project. In *Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments*, pages 51–54, 1998.

[23] Anind K Dey, Daniel Salber, Masayasu Futakawa, and Gregory D Abowd. An architecture to support context-aware applications. 1999.

[24] Hartmut Dieterich, Uwe Malinowski, Thomas Kühme, and Matthias Schneider-Hufschmidt. State of the art in adaptive user interfaces. *Human factors in information technology*, 10 :13–13, 1993.

[25] Paul Dourish. *Where the action is : the foundations of embodied interaction*. MIT press, 2004.

[26] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2) :114–131, 2003.

[27] Jérémy Fierstone, Anne-Marie Dery-Pinna, and Michel Riveill. Langage sunml. 2003.

[28] Jennifer Golbeck, Cristina Robles, and Karen Turner. Predicting personality with social media. In *CHI'11 extended abstracts on human factors in computing systems*, pages 253–262. ACM, 2011.

[29] Donatien Grolaux, Jean Vanderdonckt, Peter Van Roy, et al. Flexclock : A plastic clock written in oz with the qtk toolkit. In *Proceedings de TAMODIA 2002 (First International Workshop on Task Models and Diagrams for User Interface Design)*, 2002.

[30] Josefina Guerrero-Garcia, Juan Manuel Gonzalez-Calleros, Jean Vanderdonckt, and Jaime Muoz-Arteaga. A theoretical survey of user interface description languages : Preliminary results. In *Web Congress, 2009. LA-WEB'09. Latin American*, pages 36–43. IEEE, 2009.

[31] Philipp Gutheim. An ontology-based context inference service for mobile applications in next-generation networks. *IEEE Communications Magazine*, 49(1) :60–66, 2011.

[32] Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy. Modeling context information in pervasive computing systems. In *International Conference on Pervasive Computing*, pages 167–180. Springer, 2002.

[33] Johan Hjelm, Bruce Martin, and Peter King. Wap forum-w3c cooperation white paper. *W3C Note*, 30, 1998.

[34] Jadwiga Indulska, Ricky Robinson, Andry Rakotonirainy, and Karen Henricksen. Experiences in using cc/pp in context-aware systems. In *International Conference on Mobile Data Management*, pages 247–261. Springer, 2003.

[35] Jadwiga Indulska and Peter Sutton. Location management in pervasive systems. In *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003-Volume 21*, pages 143–151. Australian Computer Society, Inc., 2003.

[36] Michael Johnston. Building multimodal applications with emma. In *Proceedings of the 2009 international conference on Multimodal interfaces*, pages 47–54. ACM, 2009.

[37] G Jun-Zhong. Context aware computing. *Journal of East China Normal University (Natural Science)*, 5 :1–20, 2009.

[38] Abdelmadjid Ketfi, Noureddine Belkhatir, and Pierre-Yves Cunin. Adaptation dynamique, concepts et experimentations. In *Proceedings of ICSSEA*, 2002.

[39] Michael Knappmeyer, Saad Liaquat Kiani, Cristina Frà, Boris Moltchanov, and Nigel Baker. Contextml : a light-weight context representation and context management schema. In *Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on*, pages 367–372. IEEE, 2010.

[40] Panu Korpipää and Jani Mäntyjärvi. An ontology for mobile device sensor-based context awareness. In *International and Interdisciplinary Conference on Modeling and Using Context*, pages 451–458. Springer, 2003.

[41] Michal Kosinski, Yoram Bachrach, Pushmeet Kohli, David Stillwell, and Thore Graepel. Manifestations of user personality in website choice and behaviour on online social networks. *Machine learning*, 95(3) :357–380, 2014.

[42] Jean Le Feuvre, Cyril Concolato, and Jean-Claude Dufourd. Widgets mobility. In *Proceedings of the 6th International Conference on Mobile Technology, Application & Systems*, page 25. ACM, 2009.

[43] Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, and Víctor López-Jaquero. Usixml : a language supporting multi-path development of user interfaces. In *International Workshop on Design, Specification, and Verification of Interactive Systems*, pages 200–220. Springer, 2004.

[44] Ahmed Said Loubiri. Utilisation d'une ontologie et du réseau social facebook pour la modélisation du contexte pour les applications mobiles dépendantes du contexte. 2012.

[45] Lori MacVittie. *XAML in a Nutshell*. " O'Reilly Media, Inc.", 2006.

[46] Philip K McKinley, Seyed Masoud Sadjadi, Eric P Kasten, and Betty HC Cheng. A taxonomy of compositional adaptation. *Rapport Technique numéroMSU-CSE-04-17*, 2004.

[47] Ronald A Merrick. Auiml : An xml vocabulary for describing user interfaces. device independent user interfaces in xml, 2001.

[48] Mozilla Developer Network. Xml user interface language (xul), 2011.

[49] Mikael Nilsson, Johan Hjelm, and Hidetaka Ohto. Composite capabilities/preference profiles : Requirements and architecture. *W3C Working Draft*, 21 :2–28, 2000.

[50] Hector Olmedo, David Escudero, and Valentín Cardeñoso. Multimodal interaction with virtual worlds xmmvr : extensible language for multimodal interaction with virtual reality worlds. *Journal on Multimodal User Interfaces*, 9(3) :153–172, 2015.

[51] Jason Pascoe, David Morse, and Nick Ryan. Developing personal technology for the field. *Personal Technologies*, 2(1) :28–36, 1998.

[52] Angel Puerta and Jacob Eisenstein. Ximl : a common representation for interaction data. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 214–215. ACM, 2002.

[53] Gaëtan Rey and Joëlle Coutaz. Le contexteur : une abstraction logicielle pour la réalisation de systèmes interactifs sensibles au contexte. In *Proceedings of the 14th Conference on l'Interaction Homme-Machine*, pages 105–112. ACM, 2002.

[54] Diego Rios, P Dockhorn Costa, Giancarlo Guizzardi, L Ferreira Pires, J Goncalves Filho, and MJ van Sinderen. Using ontologies for modeling context-aware services platforms. 2003.

[55] Nick Ryan, Jason Pascoe, and David Morse. Enhanced reality fieldwork : the context aware archaeological assistant. *Bar International Series*, 750 :269–274, 1999.

[56] Anthony Savidis and Constantine Stephanidis. Developing dual user interfaces for integrating blind and sighted users : the homer uims. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 106–113. ACM Press/Addison-Wesley Publishing Co., 1995.

[57] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90. IEEE, 1994.

[58] Albrecht Schmidt, Michael Beigl, and Hans-W Gellersen. There is more to context than location. *Computers & Graphics*, 23(6) :893–901, 1999.

[59] Nathalie Souchon and Jean Vanderdonckt. A review of xml-compliant user interface description languages. In *Interactive Systems. Design, Specification, and Verification*, pages 377–391. Springer, 2003.

[60] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey. In *Workshop Proceedings*, 2004.

[61] Thomas Strang, Claudia Linnhoff-Popien, and Korbinian Frank. Cool : A context ontology language to enable contextual interoperability. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 236–247. Springer, 2003.

[62] Piyawadee Sukaviriya, James D Foley, and Todd Griffith. A second generation user interface design environment : the model and the runtime architecture. In *Proceedings of the INTERACT'93 and CHI'93 Conference on Human Factors in Computing Systems*, pages 375–382. ACM, 1993.

[63] David Thevenin. *Adaptation en Interaction Homme-Machine : le cas de la Plasticité*. PhD thesis, Université Joseph-Fourier-Grenoble I, 2001.

[64] David Thevenin, Gaëlle Calvary, and Joëlle Coutaz. La plasticité en interaction homme-machine. In *Actes IHM99, Actes de 11ème Conférence Francophone Interaction Homme-Machine (Montpellier, 22-26 novembre)*, pages 152–155, 1999.

[65] David Thevenin and Joëlle Coutaz. Plasticity of user interfaces : Framework and research agenda. In *Proceedings of INTERACT*, volume 99, pages 110–117, 1999.

[66] Xiao Hang Wang, D Qing Zhang, Tao Gu, and Hung Keng Pung. Ontology based context modeling and reasoning using owl. In *Pervasive Computing and Communications Work-*

shops, 2004. Proceedings of the Second IEEE Annual Conference on, pages 18–22. Ieee, 2004.

[67] Andy Ward, Alan Jones, and Andy Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5) :42–47, 1997.

[68] Mark Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7) :75–84, 1993.

[69] Bowden Wise and Ephraim P Glinert. Metawidgets for multimodal applications. In *In proceedings of the RESNA'95 conference*. Citeseer, 1995.

[70] Daqiang Zhang, Hongyu Huang, Chin-Feng Lai, Xuedong Liang, Qin Zou, and Minyi Guo. Survey on context-awareness in ubiquitous media. *Multimedia tools and applications*, 67(1) :179–211, 2013.

[71] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2) :4–10, 2012.

[72] Andreas Zimmermann. *Context management and personalisation : A tool suite for context-and user-aware computing*. Shaker, 2007.

ANNEXES

Annexe A

Questionnaire de l'enquête contextuelle

Données de l'interviewé

Âge :

Sexe :

Utilisation :

Type de téléphone : ☐ Smartphone ☐ Traditionnel

Fréquence d'utilisation : ☐ Haute ☐ normal ☐ faible

Questionnaire

Q1) Aimez-vous voyager et découvrir d'autres villes ?

R :

Q2) Utilisez-vous souvent une carte ou un guide lors de vos voyages ?

R :

Q3) Utilisez-vous une application de guide touristique ? Sinon l'idée d'avoir une telle application vous intéresse ?

R :

Q4) (Proposition des écrans… Explication de l'approche : ce que je vais vous montrer est le premier prototype de la future application…) Que pensez –vous de cette première approche ? Quelles informations vous semblent pertinentes sur ces écrans ?

R :

Q5) L'application pourra par exemple :

- Proposer des lieux en fonction de la météo et de vos préférences
- Proposer un itinéraire modifiable en fonction de vos envies

Sur base de cette description de fonctionnalité, avez-vous des suggestions ? Est-ce que vous souhaitez une fonctionnalité particulière ?

R :

Q6) Imaginons que vous n'avez plus assez de batterie. L'application vous propose deux modes de fonctionnement au choix (Illustration…) :

- Mode normal : L'application continue de fonctionner normalement avec le risque de se retrouver sans batterie.
- Mode économie d'énergie : Diminuer la précision (en coupant le signal GPS par exemple) afin de fonctionner le plus longtemps possible. L'application vous indiquera par exemple le nom d'un point stratégique dans les environs ainsi que sa localisation approximative.

Quelle serait votre réaction face à une telle situation où l'application demande votre collaboration ? Que pensez-vous de cette fonctionnalité ? Avez-vous des suggestions par rapport à cela ?

R :

Q7) Afin d'optimiser l'application et avoir des suggestions qui vous correspondent le mieux, l'application devra avoir accès à un certain nombre d'informations personnelles. Cela vous semble t'il gênant ou êtes-vous prêt à donner vos informations pour avoir un meilleur service ? Pouvez-vous cocher les informations que vous souhaitez partager ?

☐ Lien avec d'autres comptes (Facebook, Twitter, Google…)

☐ Nom et prénom

☐ Âge

☐ Sexe

☐ Fumeur

☐ Adresse

☐ Statut

☐ Position géographique

Annexe B

Résultats de l’enquête contextuelle

Compilation des données des interviews

Nombre d'interviews réalisés : 10

Parité : 6 hommes, 4 femmes

Âge moyen : 26 ans

Type de téléphone utilisé : 10/10 Smartphone

Fréquence d'utilisation : haute 6/10

Normale 3/10

Basse 1/10

Utilisation d'un support imprimé : 7/10

Utilisation d'une application : 8/10 (3 personnes seulement utilisent une application spécialisée, le reste utilise Google Map).

Données personnelles : 8/10 sont prêt à donner leurs données personnelles, mais uniquement pour améliorer les suggestions

Réaction générale sur l'adaptation par rapport à la batterie :

de manière générale, les utilisateurs ont été agréablement surpris que l'application s'adapte autant. 100 % des utilisateurs ont déclaré qu'ils choisiraient le mode économie d'énergie. Cependant 2 personnes ont déclaré que cette fonctionnalité ne les intéresse pas spécialement.

Suggestion des interviewés :

– Fonction de rappel par rapport à mes centres d'intérêt. Par exemple, si je suis intéressé par l'histoire de Paris et l'application me rappelle que je n'ai pas encore visité le château de Versailles, les lieux historiques de Paris, etc.

– Par rapport à mon historique de recherche ou mes précédentes visites, l'application peut me faire suggestion des lieux qui peuvent m'intéresser.

– L'application me propose un ensemble de lieux en fonction du lieu où je me trouve.

– L'application se met en veille lorsque je suis à l'arrêt. Si je suis par exemple en train de manger.

– S’il fait très chaud, l’application me suggère d’aller à la plage ou d'aller faire du shooping en m'indiquant les magasins qui proposent des réductions ou des magasins en fonction de mes gouts vestimentaires, etc.

– L’application me permet de voir l’avis d’autres voyageurs sur le lieu où je me trouve...

– L’application me raconte l’histoire d’un lieu, par exemple je suis devant la tour Eiffel, l’application me raconte de manière brève son histoire. L’application peut aussi m’informer sur la culture locale, la langue locale…

– L’application ajuste la luminosité par rapport à la quantité de lumière

– L’application utilise le texte et des images si j’ai le téléphone en mains et utilise la voix si j’ai le téléphone dans ma poche ou si je suis en train de conduire.

– Ça peut être intéressant si l’application me propose de visiter des lieux insolites, pas nécessairement les lieux les plus connus de la région.

– L’application me permet de faire une réservation d’hôtel, achat de billet… ce genre de chose que l’on fait avant le voyage. Pour moi, l’application idéale serait celle dans laquelle je peux tout faire, c’est-à-dire organiser tout mon voyage.

– La suggestion des lieux insolites par exemple. Au-delà des lieux connus dans une ville, c’est toujours sympa de visiter des lieux moins connus, mais qui peuvent être intéressants. En tant qu’amateur de photographie, j’aime bien ce genre de lieu.

– L’application pourrait se baser sur mes voyages précédents et me suggérer des destinations.

– Comme suggestion, je dirais au lieu de me proposer une visite guidée par jour, l’application pourrait me proposer des activités en fonction de l’heure, en fonction de la météo et de mes préférences. Par exemple, me proposer une séance de sport le matin, puis une visite de la ville, et ensuite me suggérer une activité pour la soirée. Une sorte d’assistant personnel qui me fera des suggestions en fonction de mon humeur, de mes préférences, de la météo…

– En fonction du lieu où je me trouve, l’application pourrait me proposer des petits restaurants où je peux manger les spécialités locales.

– En fonction de mes préférences l’application pourrait me proposer une activité qui me correspond. Par exemple, si je suis un fan de sport, l’application pourrait m’informer sur les clubs de football de la région, les terrains de sports, les différents matchs, les installations sportives de la région, les musées dédiés au sport…

Annexe C

Code source du modèle du contexte

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.MonGuide.com#"
     xml:base="http://www.MonGuide.com"
     xmlns:obs="http://localhost/SensorSchema/ontology#"
     xmlns:prefix="http://purl.oclc.org/NET/ssnx/qu/prefix#"
     xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
     xmlns:skos="http://www.w3.org/2004/02/skos/core#"
     xmlns:lexinfo="http://www.lexinfo.net/ontology/2.0/lexinfo#"
     xmlns:ns2="eeo:#"
     xmlns:ns1="eeo:"
     xmlns:Ontology1466434524700="http://www.semanticweb.org/ontologies/2016/5/
         Ontology1466434524700.owl#"
     xmlns:fra="http://kaiko.getalp.org/dbnary/fra/"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:dcterms="http://purl.org/dc/terms/"
     xmlns:vin="http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#"
     xmlns:locn="http://www.w3.org/ns/locn#"
     xmlns:exp="http://www.pr-owl.org/examples/pr-owl2/ProcurementFraud/
         ProcurementFraud.owl#"
     xmlns:adms="http://www.w3.org/ns/adms#"
     xmlns:qu="http://purl.oclc.org/NET/ssnx/qu/qu#"
     xmlns:var="urn:swrl#"
     xmlns:ifcowl="http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#"
     xmlns:module="http://www.example.org/module/"
     xmlns:domain="http://www.utc.fr/hds/ici/humans/domain#"
     xmlns:qu-rec20="http://purl.oclc.org/NET/ssnx/qu/qu-rec20#"
     xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#"
     xmlns:institute="http://www.example.org/institute/"
```

```
27      xmlns:odp="http://data.gov.sk/def/ontology/odp/"
28      xmlns:schema="http://www.w3.org/2001/XMLSchema/"
29      xmlns:aaa="http://www.omada.gr/obo/aaa#"
30      xmlns:geosim2="http://minsky.gsi.dit.upm.es/~gpoveda/geosim/0.1.2/geosim.owl#"
31      xmlns:study="http://www.example.org/study/"
32      xmlns:dires="http://www.disaster20.eu/dires#"
33      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
34      xmlns:bp="http://www.biopax.org/release/biopax-level3.owl#"
35      xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
36      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
37      xmlns:movieontology="http://www.movieontology.org/2009/10/01/movieontology.owl
            #"
38      xmlns:lecture="http://www.example.org/lecture/"
39      xmlns:event="http://purl.org/NET/c4dm/event.owl#"
40      xmlns:geosim="http://minsky.gsi.dit.upm.es/~gpoveda/geosim/0.1.3/ns.owl#"
41      xmlns:cc="http://creativecommons.org/ns#"
42      xmlns:a="http://www.example.com/#"
43      xmlns:food="http://www.w3.org/TR/2003/PR-owl-guide-20031209/food#"
44      xmlns:TexturePainting="http://www.semanticweb.org/ontologies/2013/SMA_CP#
            TexturePainting&amp;"
45      xmlns:s="https://univie.ac.at/~a1507714/schema/"
46      xmlns:otherOnt="http://example.org/otherOntologies/families/"
47      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
48      xmlns:inst="http://linkedbuildingdata.net/ifc/resources20160709_181019/"
49      xmlns:SMA_CP="http://www.semanticweb.org/ontologies/2013/SMA_CP#"
50      xmlns:dc="http://purl.org/dc/elements/1.1/"
51      xmlns:Film="http://www.semanticweb.org/ontologies/2013/SMA_CP#Film&amp;"
52      xmlns:owl="http://www.w3.org/2002/07/owl#"
53      xmlns:dbnary="http://kaiko.getalp.org/dbnary#"
54      xmlns:dim="http://purl.oclc.org/NET/ssnx/qu/dim#"
55      xmlns:express="http://purl.org/voc/express#"
56      xmlns:appointment="http://www.example.org/appointment/"
57      xmlns:ub="http://swat.cse.lehigh.edu/onto/univ-bench.owl#"
58      xmlns:www="http://www.movieontology.org/2009/11/09/"
59      xmlns:vann="http://purl.org/vocab/vann/"
60      xmlns:ontology="http://dbpedia.org/ontology/"
61      xmlns:foaf="http://xmlns.com/foaf/0.1/"
62      xmlns:group="http://www.example.org/group/"
63      xmlns:res="http://www.slovpedia.com/knowledge/spec/1.0/resource.owl#"
64      xmlns:CameraOperationF="http://www.semanticweb.org/ontologies/2013/SMA_CP#
            CameraOperationF&amp;"
65      xmlns:movieontology2="http://www.movieontology.org/2009/11/09/movieontology.
            owl#"
66      xmlns:swrl="http://www.w3.org/2003/11/swrl#"
```

```
67      xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
68      xmlns:voaf="http://purl.org/vocommons/voag#"
69      xmlns:bfo="http://www.ifomis.org/bfo/1.1#"
70      xmlns:list="http://www.co-ode.org/ontologies/list.owl#"
71      xmlns:j.0="http://example.org/ontology#"
72      xmlns:VisualizationF="http://www.semanticweb.org/ontologies/2013/SMA_CP#
            VisualizationF&amp;"
73      xmlns:unit="http://purl.oclc.org/NET/ssnx/qu/unit#"
74      xmlns:ex="http://example.org/stuff/1.0/"
75      xmlns:nuts="http://ec.europa.eu/eurostat/ramon/rdfdata/nuts/"
76      xmlns:adolena="http://ksg.meraka.co.za/adolena.owl#"
77      xmlns:submodule="http://www.example.org/submodule/"
78      xmlns:page="http://dbpedia.org/page/"
79      xmlns:vs="http://www.w3.org/2003/06/sw-vocab-status/ns#"
80      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
81      xmlns:LightingF="http://www.semanticweb.org/ontologies/2013/SMA_CP#LightingF&
            amp;"
82      xmlns:game="http://www.semanticweb.org/ontologies/game.owl#"
83      xmlns:ns="http://silab.hevs.ch/ontology/project/swisselectric/0.1.1/ns.owl#"
84      xmlns:dcam="http://purl.org/dc/dcam/"
85      xmlns:BusinessAnalysisEssentials="http://www.udemy.com/
            BusinessAnalysisEssentials.owl#"
86      xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
87      xmlns:Editing="http://www.semanticweb.org/ontologies/2013/SMA_CP#Editing&amp;"
88      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
89      xmlns:eeo="http://localhost/ontologies/eeo/eeo.owl#"
90      xmlns:cao="http://www.biocenit.cat/ontologies/city_anatomy/CA.owl#"
91      xmlns:lemon="http://lemon-model.net/lemon#"
92      xmlns:ContentCreation="http://www.semanticweb.org/ontologies/2013/SMA_CP#
            ContentCreation&amp;"
93      xmlns:pizza="http://www.co-ode.org/ontologies/pizza/pizza.owl#"
94      xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
95      xmlns:rov="http://www.w3.org/TR/vocab-regorg/"
96      xmlns:eng="http://kaiko.getalp.org/dbnary/eng/"
97      xmlns:ann="http://www.slovpedia.com/knowledge/nlp/1.0/annotation.owl#"
98      xmlns:quantity="http://purl.oclc.org/NET/ssnx/qu/quantity#"
99      xmlns:oboInOwl="http://www.geneontology.org/formats/oboInOwl#"
100     xmlns:gn="http://www.geonames.org/ontology#"
101     xmlns:rdaa="http://rdaregistry.info/Elements/a/"
102     xmlns:lecturer="http://www.example.org/lecturer/"
103     xmlns:gr="http://purl.org/goodrelations/v1#"
104     xmlns:mfg="http://www.workingontologist.org/Examples/Chapter3/Product.owl#"
105     xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
106     xmlns:cse="https://94d435b2.ngrok.io/ontologies/cse#"
```

```
107      xmlns:NAP="file://null/home/aurona/0AlleWerk/Navorsing/Ontologies/NAP/NAP#"
108      xmlns:obo="http://purl.obolibrary.org/obo/"
109      xmlns:snap="http://www.ifomis.org/bfo/1.1/snap#">
110    <owl:Ontology rdf:about="http://www.MonGuide.com"/>
111
112
113
114    <!--
115    ////////////////////////////////////////////////////////////////////////////////////////
116    //
117    // Annotation properties
118    //
119    ////////////////////////////////////////////////////////////////////////////////////////
120     -->
121
122
123
124
125    <!-- http://www.MonGuide.com#is_extrovert -->
126
127    <owl:AnnotationProperty rdf:about="http://www.MonGuide.com#is_extrovert"/>
128
129
130
131    <!--
132    ////////////////////////////////////////////////////////////////////////////////////////
133    //
134    // Object Properties
135    //
136    ////////////////////////////////////////////////////////////////////////////////////////
137     -->
138
139
140
141
142    <!-- http://www.MonGuide.com#current_mood -->
143
144    <owl:ObjectProperty rdf:about="http://www.MonGuide.com#current_mood">
145        <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
146        <rdfs:range rdf:resource="http://www.MonGuide.com#Mood"/>
```

```
147     </owl:ObjectProperty>
148
149
150
151     <!-- http://www.MonGuide.com#device -->
152
153     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#device">
154         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
155         <rdfs:range rdf:resource="http://www.MonGuide.com#Device"/>
156     </owl:ObjectProperty>
157
158
159
160     <!-- http://www.MonGuide.com#education -->
161
162     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#education">
163         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
164         <rdfs:range rdf:resource="http://www.MonGuide.com#EducationExperience"/>
165     </owl:ObjectProperty>
166
167
168
169     <!-- http://www.MonGuide.com#employer -->
170
171     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#employer">
172         <rdfs:domain rdf:resource="http://www.MonGuide.com#WorkExperience"/>
173         <rdfs:range rdf:resource="http://www.MonGuide.com#Entity"/>
174     </owl:ObjectProperty>
175
176
177
178     <!-- http://www.MonGuide.com#event -->
179
180     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#event">
181         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
182         <rdfs:range rdf:resource="http://www.MonGuide.com#Event"/>
183     </owl:ObjectProperty>
184
185
186
187     <!-- http://www.MonGuide.com#favorite_athlete -->
188
189     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#favorite_athlete">
190         <rdfs:subPropertyOf rdf:resource="http://www.MonGuide.com#likes"/>
```

```
191         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
192         <rdfs:range rdf:resource="http://www.MonGuide.com#Entity"/>
193     </owl:ObjectProperty>
194
195
196
197     <!-- http://www.MonGuide.com#favorite_team -->
198
199     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#favorite_team">
200         <rdfs:subPropertyOf rdf:resource="http://www.MonGuide.com#likes"/>
201         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
202         <rdfs:range rdf:resource="http://www.MonGuide.com#Entity"/>
203     </owl:ObjectProperty>
204
205
206
207     <!-- http://www.MonGuide.com#group -->
208
209     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#group">
210         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
211         <rdfs:range rdf:resource="http://www.MonGuide.com#Group"/>
212     </owl:ObjectProperty>
213
214
215
216     <!-- http://www.MonGuide.com#hometown -->
217
218     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#hometown">
219         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
220         <rdfs:range rdf:resource="http://www.MonGuide.com#Location"/>
221     </owl:ObjectProperty>
222
223
224
225     <!-- http://www.MonGuide.com#inspirational_people -->
226
227     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#inspirational_people">
228         <rdfs:subPropertyOf rdf:resource="http://www.MonGuide.com#likes"/>
229         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
230         <rdfs:range rdf:resource="http://www.MonGuide.com#Entity"/>
231     </owl:ObjectProperty>
232
233
234
```

```
235     <!-- http://www.MonGuide.com#likes -->
236
237     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#likes">
238         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
239         <rdfs:range rdf:resource="http://www.MonGuide.com#Entity"/>
240     </owl:ObjectProperty>
241
242
243
244     <!-- http://www.MonGuide.com#location -->
245
246     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#location">
247         <rdfs:domain>
248             <owl:Class>
249                 <owl:unionOf rdf:parseType="Collection">
250                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
251                     <rdf:Description rdf:about="http://www.MonGuide.com#Place"/>
252                     <rdf:Description rdf:about="http://www.MonGuide.com#User"/>
253                 </owl:unionOf>
254             </owl:Class>
255         </rdfs:domain>
256         <rdfs:range rdf:resource="http://www.MonGuide.com#Location"/>
257     </owl:ObjectProperty>
258
259
260
261     <!-- http://www.MonGuide.com#music -->
262
263     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#music">
264         <rdfs:subPropertyOf rdf:resource="http://www.MonGuide.com#likes"/>
265         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
266         <rdfs:range rdf:resource="http://www.MonGuide.com#Entity"/>
267     </owl:ObjectProperty>
268
269
270
271     <!-- http://www.MonGuide.com#school -->
272
273     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#school">
274         <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#
            topObjectProperty"/>
275         <rdfs:domain rdf:resource="http://www.MonGuide.com#EducationExperience"/>
276         <rdfs:range rdf:resource="http://www.MonGuide.com#Entity"/>
277     </owl:ObjectProperty>
```

```
278
279
280
281     <!-- http://www.MonGuide.com#visit -->
282
283     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#visit">
284         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
285         <rdfs:range rdf:resource="http://www.MonGuide.com#Place"/>
286     </owl:ObjectProperty>
287
288
289
290     <!-- http://www.MonGuide.com#work -->
291
292     <owl:ObjectProperty rdf:about="http://www.MonGuide.com#work">
293         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
294         <rdfs:range rdf:resource="http://www.MonGuide.com#WorkExperience"/>
295     </owl:ObjectProperty>
296
297
298
299     <!--
300     ///////////////////////////////////////////////////////////////////////////////////////
301     //
302     // Data properties
303     //
304     ///////////////////////////////////////////////////////////////////////////////////////
305      -->
306
307
308
309
310     <!-- http://www.MonGuide.com#address -->
311
312     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#address">
313         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
314         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
315     </owl:DatatypeProperty>
316
317
318
319     <!-- http://www.MonGuide.com#age -->
```

```
320
321     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#age">
322         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
323         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
324     </owl:DatatypeProperty>
325
326
327
328     <!-- http://www.MonGuide.com#attending_count -->
329
330     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#attending_count">
331         <rdfs:domain rdf:resource="http://www.MonGuide.com#Event"/>
332         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
333     </owl:DatatypeProperty>
334
335
336
337     <!-- http://www.MonGuide.com#birthday -->
338
339     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#birthday">
340         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
341         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
342     </owl:DatatypeProperty>
343
344
345
346     <!-- http://www.MonGuide.com#city -->
347
348     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#city">
349         <rdfs:domain rdf:resource="http://www.MonGuide.com#Location"/>
350         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
351     </owl:DatatypeProperty>
352
353
354
355     <!-- http://www.MonGuide.com#country -->
356
357     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#country">
358         <rdfs:domain rdf:resource="http://www.MonGuide.com#Location"/>
359         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
360     </owl:DatatypeProperty>
361
362
363
```

```
364     <!-- http://www.MonGuide.com#currency -->
365
366     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#currency">
367         <rdfs:domain>
368             <owl:Class>
369                 <owl:unionOf rdf:parseType="Collection">
370                     <rdf:Description rdf:about="http://www.MonGuide.com#Location"/>
371                     <rdf:Description rdf:about="http://www.MonGuide.com#User"/>
372                 </owl:unionOf>
373             </owl:Class>
374         </rdfs:domain>
375         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
376     </owl:DatatypeProperty>
377
378
379
380     <!-- http://www.MonGuide.com#description -->
381
382     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#description">
383         <rdfs:domain>
384             <owl:Class>
385                 <owl:unionOf rdf:parseType="Collection">
386                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
387                     <rdf:Description rdf:about="http://www.MonGuide.com#Event"/>
388                     <rdf:Description rdf:about="http://www.MonGuide.com#Group"/>
389                     <rdf:Description rdf:about="http://www.MonGuide.com#
                            WorkExperience"/>
390                 </owl:unionOf>
391             </owl:Class>
392         </rdfs:domain>
393         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
394     </owl:DatatypeProperty>
395
396
397
398     <!-- http://www.MonGuide.com#email -->
399
400     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#email">
401         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
402         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
403     </owl:DatatypeProperty>
404
405
406
```

```
407     <!-- http://www.MonGuide.com#end_date -->
408
409     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#end_date">
410         <rdfs:domain rdf:resource="http://www.MonGuide.com#WorkExperience"/>
411         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
412     </owl:DatatypeProperty>
413
414
415
416     <!-- http://www.MonGuide.com#end_time -->
417
418     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#end_time">
419         <rdfs:domain rdf:resource="http://www.MonGuide.com#Event"/>
420         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
421     </owl:DatatypeProperty>
422
423
424
425     <!-- http://www.MonGuide.com#gender -->
426
427     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#gender">
428         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
429         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
430     </owl:DatatypeProperty>
431
432
433
434     <!-- http://www.MonGuide.com#hardware -->
435
436     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#hardware">
437         <rdfs:domain rdf:resource="http://www.MonGuide.com#Device"/>
438         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
439     </owl:DatatypeProperty>
440
441
442
443     <!-- http://www.MonGuide.com#has_children -->
444
445     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#has_children">
446         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
447         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
448     </owl:DatatypeProperty>
449
450
```

```
451
452     <!-- http://www.MonGuide.com#has_sociallife -->
453
454     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#has_sociallife">
455         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
456         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
457     </owl:DatatypeProperty>
458
459
460
461     <!-- http://www.MonGuide.com#id -->
462
463     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#id">
464         <rdfs:domain>
465             <owl:Class>
466                 <owl:unionOf rdf:parseType="Collection">
467                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
468                     <rdf:Description rdf:about="http://www.MonGuide.com#Event"/>
469                     <rdf:Description rdf:about="http://www.MonGuide.com#Group"/>
470                     <rdf:Description rdf:about="http://www.MonGuide.com#Place"/>
471                     <rdf:Description rdf:about="http://www.MonGuide.com#User"/>
472                 </owl:unionOf>
473             </owl:Class>
474         </rdfs:domain>
475         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
476     </owl:DatatypeProperty>
477
478
479
480     <!-- http://www.MonGuide.com#interested_count -->
481
482     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#interested_count">
483         <rdfs:domain rdf:resource="http://www.MonGuide.com#Event"/>
484         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
485     </owl:DatatypeProperty>
486
487
488
489     <!-- http://www.MonGuide.com#is_Extrovert -->
490
491     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#is_Extrovert">
492         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
493         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
494     </owl:DatatypeProperty>
```

```
495
496
497
498     <!-- http://www.MonGuide.com#languages -->
499
500     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#languages">
501         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
502         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
503     </owl:DatatypeProperty>
504
505
506
507     <!-- http://www.MonGuide.com#latitude -->
508
509     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#latitude">
510         <rdfs:domain rdf:resource="http://www.MonGuide.com#Location"/>
511         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#double"/>
512     </owl:DatatypeProperty>
513
514
515
516     <!-- http://www.MonGuide.com#located_in -->
517
518     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#located_in">
519         <rdfs:domain rdf:resource="http://www.MonGuide.com#Location"/>
520         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
521     </owl:DatatypeProperty>
522
523
524
525     <!-- http://www.MonGuide.com#longitude -->
526
527     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#longitude">
528         <rdfs:domain rdf:resource="http://www.MonGuide.com#Location"/>
529         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#double"/>
530     </owl:DatatypeProperty>
531
532
533
534     <!-- http://www.MonGuide.com#name -->
535
536     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#name">
537         <rdfs:domain>
538             <owl:Class>
```

```
539                 <owl:unionOf rdf:parseType="Collection">
540                     <rdf:Description rdf:about="http://www.MonGuide.com#
                            EducationExperience"/>
541                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
542                     <rdf:Description rdf:about="http://www.MonGuide.com#Event"/>
543                     <rdf:Description rdf:about="http://www.MonGuide.com#Group"/>
544                     <rdf:Description rdf:about="http://www.MonGuide.com#Location"/>
545                     <rdf:Description rdf:about="http://www.MonGuide.com#Mood"/>
546                     <rdf:Description rdf:about="http://www.MonGuide.com#Place"/>
547                     <rdf:Description rdf:about="http://www.MonGuide.com#User"/>
548                 </owl:unionOf>
549             </owl:Class>
550         </rdfs:domain>
551         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
552     </owl:DatatypeProperty>
553
554
555
556     <!-- http://www.MonGuide.com#nb_friend -->
557
558     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#nb_friend">
559         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
560         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
561     </owl:DatatypeProperty>
562
563
564
565     <!-- http://www.MonGuide.com#os -->
566
567     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#os">
568         <rdfs:domain rdf:resource="http://www.MonGuide.com#Device"/>
569         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
570     </owl:DatatypeProperty>
571
572
573
574     <!-- http://www.MonGuide.com#overall_rating -->
575
576     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#overall_rating">
577         <rdfs:domain rdf:resource="http://www.MonGuide.com#Place"/>
578         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#long"/>
579     </owl:DatatypeProperty>
580
581
```

```
582
583     <!-- http://www.MonGuide.com#political -->
584
585     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#political">
586         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
587         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
588     </owl:DatatypeProperty>
589
590
591
592     <!-- http://www.MonGuide.com#position -->
593
594     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#position">
595         <rdfs:domain rdf:resource="http://www.MonGuide.com#WorkExperience"/>
596         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
597     </owl:DatatypeProperty>
598
599
600
601     <!-- http://www.MonGuide.com#quotes -->
602
603     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#quotes">
604         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
605         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
606     </owl:DatatypeProperty>
607
608
609
610     <!-- http://www.MonGuide.com#region -->
611
612     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#region">
613         <rdfs:domain rdf:resource="http://www.MonGuide.com#Location"/>
614         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
615     </owl:DatatypeProperty>
616
617
618
619     <!-- http://www.MonGuide.com#relationship_status -->
620
621     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#relationship_status">
622         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
623         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
624     </owl:DatatypeProperty>
625
```

```
626
627
628     <!-- http://www.MonGuide.com#religion -->
629
630     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#religion">
631         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
632         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
633     </owl:DatatypeProperty>
634
635
636
637     <!-- http://www.MonGuide.com#sport -->
638
639     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#sport">
640         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
641         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
642     </owl:DatatypeProperty>
643
644
645
646     <!-- http://www.MonGuide.com#start_date -->
647
648     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#start_date">
649         <rdfs:domain rdf:resource="http://www.MonGuide.com#WorkExperience"/>
650         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
651     </owl:DatatypeProperty>
652
653
654
655     <!-- http://www.MonGuide.com#start_time -->
656
657     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#start_time">
658         <rdfs:domain rdf:resource="http://www.MonGuide.com#Event"/>
659         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
660     </owl:DatatypeProperty>
661
662
663
664     <!-- http://www.MonGuide.com#state -->
665
666     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#state">
667         <rdfs:domain rdf:resource="http://www.MonGuide.com#Location"/>
668         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
669     </owl:DatatypeProperty>
```

```
670
671
672
673     <!-- http://www.MonGuide.com#street -->
674
675     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#street">
676         <rdfs:domain rdf:resource="http://www.MonGuide.com#Location"/>
677         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
678     </owl:DatatypeProperty>
679
680
681
682     <!-- http://www.MonGuide.com#temperature -->
683
684     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#temperature">
685         <rdfs:domain rdf:resource="http://www.MonGuide.com#Location"/>
686         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
687     </owl:DatatypeProperty>
688
689
690
691     <!-- http://www.MonGuide.com#time -->
692
693     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#time">
694         <rdfs:domain>
695             <owl:Class>
696                 <owl:unionOf rdf:parseType="Collection">
697                     <rdf:Description rdf:about="http://www.MonGuide.com#Mood"/>
698                     <rdf:Description rdf:about="http://www.MonGuide.com#Place"/>
699                 </owl:unionOf>
700             </owl:Class>
701         </rdfs:domain>
702         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
703     </owl:DatatypeProperty>
704
705
706
707     <!-- http://www.MonGuide.com#timezone -->
708
709     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#timezone">
710         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
711         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#double"/>
712     </owl:DatatypeProperty>
713
```

```
714
715
716     <!-- http://www.MonGuide.com#type -->
717
718     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#type">
719         <rdfs:domain>
720             <owl:Class>
721                 <owl:unionOf rdf:parseType="Collection">
722                     <rdf:Description rdf:about="http://www.MonGuide.com#
                            EducationExperience"/>
723                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
724                     <rdf:Description rdf:about="http://www.MonGuide.com#Event"/>
725                 </owl:unionOf>
726             </owl:Class>
727         </rdfs:domain>
728         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
729     </owl:DatatypeProperty>
730
731
732
733     <!-- http://www.MonGuide.com#updated_time -->
734
735     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#updated_time">
736         <rdfs:domain rdf:resource="http://www.MonGuide.com#User"/>
737         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
738     </owl:DatatypeProperty>
739
740
741
742     <!-- http://www.MonGuide.com#year -->
743
744     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#year">
745         <rdfs:domain rdf:resource="http://www.MonGuide.com#EducationExperience"/>
746         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
747     </owl:DatatypeProperty>
748
749
750
751     <!-- http://www.MonGuide.com#zip -->
752
753     <owl:DatatypeProperty rdf:about="http://www.MonGuide.com#zip">
754         <rdfs:domain rdf:resource="http://www.MonGuide.com#Location"/>
755         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
756     </owl:DatatypeProperty>
```

```
757
758
759
760     <!--
761     ////////////////////////////////////////////////////////////////////////////////////////
762     //
763     // Classes
764     //
765     ////////////////////////////////////////////////////////////////////////////////////////
766      -->
767
768
769
770
771     <!-- http://www.MonGuide.com#Actor_Director -->
772
773     <owl:Class rdf:about="http://www.MonGuide.com#Actor_Director">
774         <owl:equivalentClass>
775             <owl:Class>
776                 <owl:intersectionOf rdf:parseType="Collection">
777                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
778                     <owl:Restriction>
779                         <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
780                         <owl:hasValue>Actor/Director</owl:hasValue>
781                     </owl:Restriction>
782                 </owl:intersectionOf>
783             </owl:Class>
784         </owl:equivalentClass>
785         <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
786     </owl:Class>
787
788
789
790     <!-- http://www.MonGuide.com#ArtEnthusiast -->
791
792     <owl:Class rdf:about="http://www.MonGuide.com#ArtEnthusiast">
793         <owl:equivalentClass>
794             <owl:Class>
795                 <owl:intersectionOf rdf:parseType="Collection">
796                     <rdf:Description rdf:about="http://www.MonGuide.com#User"/>
797                     <owl:Restriction>
```

```
798                         <owl:onProperty rdf:resource="http://www.MonGuide.com#likes
                                "/>
799                         <owl:someValuesFrom rdf:resource="http://www.MonGuide.com#
                                Museum_Art"/>
800                     </owl:Restriction>
801                 </owl:intersectionOf>
802             </owl:Class>
803         </owl:equivalentClass>
804         <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#User"/>
805     </owl:Class>
806
807
808
809     <!-- http://www.MonGuide.com#Artist -->
810
811     <owl:Class rdf:about="http://www.MonGuide.com#Artist">
812         <owl:equivalentClass>
813             <owl:Class>
814                 <owl:intersectionOf rdf:parseType="Collection">
815                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
816                     <owl:Restriction>
817                         <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
818                         <owl:hasValue>Artist</owl:hasValue>
819                     </owl:Restriction>
820                 </owl:intersectionOf>
821             </owl:Class>
822         </owl:equivalentClass>
823         <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
824     </owl:Class>
825
826
827
828     <!-- http://www.MonGuide.com#Athlete -->
829
830     <owl:Class rdf:about="http://www.MonGuide.com#Athlete">
831         <owl:equivalentClass>
832             <owl:Class>
833                 <owl:intersectionOf rdf:parseType="Collection">
834                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
835                     <owl:Restriction>
836                         <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
837                         <owl:hasValue>Athlete</owl:hasValue>
```

```
838                     </owl:Restriction>
839                 </owl:intersectionOf>
840             </owl:Class>
841         </owl:equivalentClass>
842         <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
843     </owl:Class>
844 
845 
846 
847     <!-- http://www.MonGuide.com#Author -->
848 
849     <owl:Class rdf:about="http://www.MonGuide.com#Author">
850         <owl:equivalentClass>
851             <owl:Class>
852                 <owl:intersectionOf rdf:parseType="Collection">
853                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
854                     <owl:Restriction>
855                         <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
856                         <owl:hasValue>Author</owl:hasValue>
857                     </owl:Restriction>
858                 </owl:intersectionOf>
859             </owl:Class>
860         </owl:equivalentClass>
861         <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
862     </owl:Class>
863 
864 
865 
866     <!-- http://www.MonGuide.com#Book -->
867 
868     <owl:Class rdf:about="http://www.MonGuide.com#Book">
869         <owl:equivalentClass>
870             <owl:Class>
871                 <owl:intersectionOf rdf:parseType="Collection">
872                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
873                     <owl:Restriction>
874                         <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
875                         <owl:hasValue>Book</owl:hasValue>
876                     </owl:Restriction>
877                 </owl:intersectionOf>
878             </owl:Class>
879         </owl:equivalentClass>
```

```
880         <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
881     </owl:Class>
882
883
884
885     <!-- http://www.MonGuide.com#Church_Religious -->
886
887     <owl:Class rdf:about="http://www.MonGuide.com#Church_Religious">
888         <owl:equivalentClass>
889             <owl:Class>
890                 <owl:intersectionOf rdf:parseType="Collection">
891                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
892                     <owl:Restriction>
893                         <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
894                         <owl:hasValue>Church/Religious Organization</owl:hasValue>
895                     </owl:Restriction>
896                 </owl:intersectionOf>
897             </owl:Class>
898         </owl:equivalentClass>
899         <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
900     </owl:Class>
901
902
903
904     <!-- http://www.MonGuide.com#Club -->
905
906     <owl:Class rdf:about="http://www.MonGuide.com#Club">
907         <owl:equivalentClass>
908             <owl:Class>
909                 <owl:intersectionOf rdf:parseType="Collection">
910                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
911                     <owl:Restriction>
912                         <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
913                         <owl:hasValue>Club</owl:hasValue>
914                     </owl:Restriction>
915                 </owl:intersectionOf>
916             </owl:Class>
917         </owl:equivalentClass>
918         <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
919     </owl:Class>
920
921
```

```
922
923     <!-- http://www.MonGuide.com#Comedian -->
924
925     <owl:Class rdf:about="http://www.MonGuide.com#Comedian">
926         <owl:equivalentClass>
927             <owl:Class>
928                 <owl:intersectionOf rdf:parseType="Collection">
929                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
930                     <owl:Restriction>
931                         <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
932                         <owl:hasValue>Comedian</owl:hasValue>
933                     </owl:Restriction>
934                 </owl:intersectionOf>
935             </owl:Class>
936         </owl:equivalentClass>
937         <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
938     </owl:Class>
939
940
941
942     <!-- http://www.MonGuide.com#Community -->
943
944     <owl:Class rdf:about="http://www.MonGuide.com#Community">
945         <owl:equivalentClass>
946             <owl:Class>
947                 <owl:intersectionOf rdf:parseType="Collection">
948                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
949                     <owl:Restriction>
950                         <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
951                         <owl:hasValue>Community</owl:hasValue>
952                     </owl:Restriction>
953                 </owl:intersectionOf>
954             </owl:Class>
955         </owl:equivalentClass>
956         <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
957     </owl:Class>
958
959
960
961     <!-- http://www.MonGuide.com#Company -->
962
963     <owl:Class rdf:about="http://www.MonGuide.com#Company">
```

```
964         <owl:equivalentClass>
965             <owl:Class>
966                 <owl:intersectionOf rdf:parseType="Collection">
967                     <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
968                     <owl:Restriction>
969                         <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
970                         <owl:hasValue>Company</owl:hasValue>
971                     </owl:Restriction>
972                 </owl:intersectionOf>
973             </owl:Class>
974         </owl:equivalentClass>
975         <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
976     </owl:Class>
977 
978 
979 
980     <!-- http://www.MonGuide.com#Device -->
981 
982     <owl:Class rdf:about="http://www.MonGuide.com#Device"/>
983 
984 
985 
986     <!-- http://www.MonGuide.com#EducationExperience -->
987 
988     <owl:Class rdf:about="http://www.MonGuide.com#EducationExperience">
989         <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Experience"/>
990     </owl:Class>
991 
992 
993 
994     <!-- http://www.MonGuide.com#Entertainer -->
995 
996     <owl:Class rdf:about="http://www.MonGuide.com#Entertainer">
997         <owl:equivalentClass>
998             <owl:Class>
999                 <owl:intersectionOf rdf:parseType="Collection">
1000                    <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
1001                    <owl:Restriction>
1002                        <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
1003                        <owl:hasValue>Entertainer</owl:hasValue>
1004                    </owl:Restriction>
1005                </owl:intersectionOf>
```

```
1006                </owl:Class>
1007            </owl:equivalentClass>
1008            <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
1009        </owl:Class>
1010
1011
1012
1013        <!-- http://www.MonGuide.com#Entity -->
1014
1015        <owl:Class rdf:about="http://www.MonGuide.com#Entity"/>
1016
1017
1018
1019        <!-- http://www.MonGuide.com#Event -->
1020
1021        <owl:Class rdf:about="http://www.MonGuide.com#Event"/>
1022
1023
1024
1025        <!-- http://www.MonGuide.com#Experience -->
1026
1027        <owl:Class rdf:about="http://www.MonGuide.com#Experience"/>
1028
1029
1030
1031        <!-- http://www.MonGuide.com#Group -->
1032
1033        <owl:Class rdf:about="http://www.MonGuide.com#Group"/>
1034
1035
1036
1037        <!-- http://www.MonGuide.com#Location -->
1038
1039        <owl:Class rdf:about="http://www.MonGuide.com#Location"/>
1040
1041
1042
1043        <!-- http://www.MonGuide.com#Mood -->
1044
1045        <owl:Class rdf:about="http://www.MonGuide.com#Mood"/>
1046
1047
1048
1049        <!-- http://www.MonGuide.com#Movie -->
```

```
1050
1051    <owl:Class rdf:about="http://www.MonGuide.com#Movie">
1052        <owl:equivalentClass>
1053            <owl:Class>
1054                <owl:intersectionOf rdf:parseType="Collection">
1055                    <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
1056                    <owl:Restriction>
1057                        <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
1058                        <owl:hasValue>Movie</owl:hasValue>
1059                    </owl:Restriction>
1060                </owl:intersectionOf>
1061            </owl:Class>
1062        </owl:equivalentClass>
1063        <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
1064    </owl:Class>
1065
1066
1067
1068    <!-- http://www.MonGuide.com#Museum_Art -->
1069
1070    <owl:Class rdf:about="http://www.MonGuide.com#Museum_Art">
1071        <owl:equivalentClass>
1072            <owl:Class>
1073                <owl:intersectionOf rdf:parseType="Collection">
1074                    <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
1075                    <owl:Restriction>
1076                        <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
1077                        <owl:hasValue>Museum/Art</owl:hasValue>
1078                    </owl:Restriction>
1079                </owl:intersectionOf>
1080            </owl:Class>
1081        </owl:equivalentClass>
1082        <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
1083    </owl:Class>
1084
1085
1086
1087    <!-- http://www.MonGuide.com#MusicEnthusiast -->
1088
1089    <owl:Class rdf:about="http://www.MonGuide.com#MusicEnthusiast">
1090        <owl:equivalentClass>
1091            <owl:Class>
```

```
1092                <owl:intersectionOf rdf:parseType="Collection">
1093                    <rdf:Description rdf:about="http://www.MonGuide.com#User"/>
1094                    <owl:Restriction>
1095                        <owl:onProperty rdf:resource="http://www.MonGuide.com#music
                                "/>
1096                        <owl:minCardinality rdf:datatype="http://www.w3.org/2001/
                                XMLSchema#nonNegativeInteger">3</owl:minCardinality>
1097                    </owl:Restriction>
1098                </owl:intersectionOf>
1099            </owl:Class>
1100        </owl:equivalentClass>
1101        <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#User"/>
1102    </owl:Class>
1103
1104
1105
1106    <!-- http://www.MonGuide.com#Musician_Band -->
1107
1108    <owl:Class rdf:about="http://www.MonGuide.com#Musician_Band">
1109        <owl:equivalentClass>
1110            <owl:Class>
1111                <owl:unionOf rdf:parseType="Collection">
1112                    <owl:Class>
1113                        <owl:intersectionOf rdf:parseType="Collection">
1114                            <rdf:Description rdf:about="http://www.MonGuide.com#
                                    Entity"/>
1115                            <owl:Restriction>
1116                                <owl:onProperty rdf:resource="http://www.MonGuide.
                                        com#type"/>
1117                                <owl:hasValue>Musician/Band</owl:hasValue>
1118                            </owl:Restriction>
1119                        </owl:intersectionOf>
1120                    </owl:Class>
1121                    <owl:Restriction>
1122                        <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
1123                        <owl:hasValue>Song</owl:hasValue>
1124                    </owl:Restriction>
1125                </owl:unionOf>
1126            </owl:Class>
1127        </owl:equivalentClass>
1128        <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
1129    </owl:Class>
1130
```

```
1131
1132
1133    <!-- http://www.MonGuide.com#Place -->
1134
1135    <owl:Class rdf:about="http://www.MonGuide.com#Place"/>
1136
1137
1138
1139    <!-- http://www.MonGuide.com#Product_Service -->
1140
1141    <owl:Class rdf:about="http://www.MonGuide.com#Product_Service">
1142        <owl:equivalentClass>
1143            <owl:Class>
1144                <owl:intersectionOf rdf:parseType="Collection">
1145                    <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
1146                    <owl:Restriction>
1147                        <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
1148                        <owl:hasValue>Product/Service</owl:hasValue>
1149                    </owl:Restriction>
1150                </owl:intersectionOf>
1151            </owl:Class>
1152        </owl:equivalentClass>
1153        <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
1154    </owl:Class>
1155
1156
1157
1158    <!-- http://www.MonGuide.com#PublicFigure -->
1159
1160    <owl:Class rdf:about="http://www.MonGuide.com#PublicFigure">
1161        <owl:equivalentClass>
1162            <owl:Class>
1163                <owl:intersectionOf rdf:parseType="Collection">
1164                    <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
1165                    <owl:Restriction>
1166                        <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
1167                        <owl:hasValue>Public Figure</owl:hasValue>
1168                    </owl:Restriction>
1169                </owl:intersectionOf>
1170            </owl:Class>
1171        </owl:equivalentClass>
1172        <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
```

```
1173    </owl:Class>
1174
1175
1176
1177    <!-- http://www.MonGuide.com#Song -->
1178
1179    <owl:Class rdf:about="http://www.MonGuide.com#Song">
1180        <owl:equivalentClass>
1181            <owl:Class>
1182                <owl:intersectionOf rdf:parseType="Collection">
1183                    <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
1184                    <owl:Restriction>
1185                        <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
1186                        <owl:hasValue>Song</owl:hasValue>
1187                    </owl:Restriction>
1188                </owl:intersectionOf>
1189            </owl:Class>
1190        </owl:equivalentClass>
1191        <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
1192    </owl:Class>
1193
1194
1195
1196    <!-- http://www.MonGuide.com#Sport -->
1197
1198    <owl:Class rdf:about="http://www.MonGuide.com#Sport">
1199        <owl:equivalentClass>
1200            <owl:Class>
1201                <owl:intersectionOf rdf:parseType="Collection">
1202                    <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
1203                    <owl:Restriction>
1204                        <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
1205                        <owl:hasValue>Sport</owl:hasValue>
1206                    </owl:Restriction>
1207                </owl:intersectionOf>
1208            </owl:Class>
1209        </owl:equivalentClass>
1210        <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
1211    </owl:Class>
1212
1213
1214
```

```
1215    <!-- http://www.MonGuide.com#SportEnthusiast -->
1216
1217    <owl:Class rdf:about="http://www.MonGuide.com#SportEnthusiast">
1218        <owl:equivalentClass>
1219            <owl:Class>
1220                <owl:intersectionOf rdf:parseType="Collection">
1221                    <rdf:Description rdf:about="http://www.MonGuide.com#User"/>
1222                    <owl:Restriction>
1223                        <owl:onProperty rdf:resource="http://www.MonGuide.com#
                                favorite_athlete"/>
1224                        <owl:someValuesFrom rdf:resource="http://www.MonGuide.com#
                                Entity"/>
1225                    </owl:Restriction>
1226                </owl:intersectionOf>
1227            </owl:Class>
1228        </owl:equivalentClass>
1229        <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#User"/>
1230    </owl:Class>
1231
1232
1233
1234    <!-- http://www.MonGuide.com#SportTeam -->
1235
1236    <owl:Class rdf:about="http://www.MonGuide.com#SportTeam">
1237        <owl:equivalentClass>
1238            <owl:Class>
1239                <owl:intersectionOf rdf:parseType="Collection">
1240                    <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
1241                    <owl:Restriction>
1242                        <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
1243                        <owl:hasValue>Sports Team</owl:hasValue>
1244                    </owl:Restriction>
1245                </owl:intersectionOf>
1246            </owl:Class>
1247        </owl:equivalentClass>
1248        <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
1249    </owl:Class>
1250
1251
1252
1253    <!-- http://www.MonGuide.com#TVShow -->
1254
1255    <owl:Class rdf:about="http://www.MonGuide.com#TVShow">
```

```
1256        <owl:equivalentClass>
1257            <owl:Class>
1258                <owl:intersectionOf rdf:parseType="Collection">
1259                    <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
1260                    <owl:Restriction>
1261                        <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
1262                        <owl:hasValue>TV Show</owl:hasValue>
1263                    </owl:Restriction>
1264                </owl:intersectionOf>
1265            </owl:Class>
1266        </owl:equivalentClass>
1267        <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
1268    </owl:Class>
1269
1270
1271
1272    <!-- http://www.MonGuide.com#University -->
1273
1274    <owl:Class rdf:about="http://www.MonGuide.com#University">
1275        <owl:equivalentClass>
1276            <owl:Class>
1277                <owl:intersectionOf rdf:parseType="Collection">
1278                    <rdf:Description rdf:about="http://www.MonGuide.com#Entity"/>
1279                    <owl:Restriction>
1280                        <owl:onProperty rdf:resource="http://www.MonGuide.com#type"
                                />
1281                        <owl:hasValue>University</owl:hasValue>
1282                    </owl:Restriction>
1283                </owl:intersectionOf>
1284            </owl:Class>
1285        </owl:equivalentClass>
1286        <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Entity"/>
1287    </owl:Class>
1288
1289
1290
1291    <!-- http://www.MonGuide.com#User -->
1292
1293    <owl:Class rdf:about="http://www.MonGuide.com#User"/>
1294
1295
1296
1297    <!-- http://www.MonGuide.com#WorkExperience -->
```

```
1298
1299      <owl:Class rdf:about="http://www.MonGuide.com#WorkExperience">
1300          <rdfs:subClassOf rdf:resource="http://www.MonGuide.com#Experience"/>
1301      </owl:Class>
1302 </rdf:RDF>
1303
1304
1305
1306 <!-- Generated by the OWL API (version 4.2.1.20160306-0033) https://github.com/
         owlcs/owlapi -->
```