Long Life Applications: Un mécanisme de détection de contexte et d'analyse des situations

Riadh Karchoud, Philippe Roose, Marc Dalmau
LIUPPA
2 Allée du Parc de Montaury, Anglet, France
prenom.nom@univ-pau.fr

Arantza Illarramendi
UPV/EHU
Paseo de Manuel Lardizabal, 1, 20018 Donostia, Gipuzkoa, Espagne
a.illarramendi@ehu.es

Sergio Ilarri
UNIZAR
Pedro Cerbuna, 12 50009 Zaragoza, Espagne
silarri@unizar.es

RESUME

De nos jours, les appareils mobiles accueillent de nombreuses applications qui sont directement téléchargées et installées à partir d'un "Store". L'existence d'une telle quantité d'applications pour une telle multitude d'utilisations constitue une tâche redondante et pénible pour les utilisateurs qui sont doivent choisir, installer, exécuter puis supprimer ces applications. De plus, les développeurs d'applications mobiles négligent généralement de prendre en compte le contexte de l'utilisateur et d'utilisation en proposant des scénarios fixes et non évolutifs. Répondre à ces préoccupations suppose fournir une application unique, constamment exécutée sur des dispositifs mobiles qui gèrent automatiquement des composants logiciels en fonction des besoins des utilisateurs. Dans ce cadre, notre proposition est celle d'un concept d'application éternelle (Long Life Application ou LLA) qui constitue une nouvelle façon de répondre aux besoins de l'utilisateur de manière dynamique et intelligente. Une telle application évolue en cours d'exécution (par inclusion / exclusion de fonctionnalités, par mise à jour des modes d'interaction et par migration des exécutions) en fonction des besoins de l'utilisateur lors de ses déplacements dans un environnement connecté.

CCS Concepts

• **Human-centered computing~Mobile computing** • *Human-centered computing~Ubiquitous and mobile computing* • *Human-centered computing~Ubiquitous and mobile computing systems and tools*

Mots clés

Applications mobiles distribuées, Applications Long-life, Manipulation de contexte, smart-*, Informatique ubiquitaire.

1. INTRODUCTION

Le monde des applications mobiles a connu trois grandes transformations [1]. Au début, les applications transformaient le téléphone en appareil mono-objectif pour servir d'outil dédié à un besoin spécifique (Calculatrice, Alarme etc.). Puis, vint l'ère de l'application « d'accueil » où les applications ont lutté pour devenir la référence pour les utilisateurs en présentant de multiples onglets de fonctionnalités et de services. La troisième phase, la nouvelle ère des applications mobiles, est «l'ère sans applications » et il y a une application pour ça [2] !

En termes simples, l'application doit être au courant de ce qui se passe et de ce qui est susceptible d'arriver à l'utilisateur en fonction de son contexte spatio- temporel et social. Elle doit utiliser ce contexte de façon intelligente pour pouvoir se manifester seulement en cas de besoin. Comme le dit Paul Féval dans « Le Bossu » (1857) : « Si tu ne viens pas à Lagardère, Lagardère ira à toi ! ». Nous pensons que l'application doit aller vers l'utilisateur même s'il ne la cherche pas.

Les « Long Life Applications » (LLA) ou applications a longue durée de vie appelées aussi applications éternelles, applications silencieuses ou même applications que l'on n'ouvre jamais, ne sont pas un concept familier aux utilisateurs ni aux développeurs. Les applications mobiles classiques, que les utilisateurs téléchargent actuellement, ne servent qu'à des fins spécifiques et sont supprimées ou oubliées la plupart du temps dès la première utilisation. Dans [3] Anindya Datta dit: " Une application qui est retenue par 30% des utilisateurs est considérée comme « collante ». Il indique qu'environ 80% à 90% des applications sont finalement supprimées des téléphones des utilisateurs parce qu'elles sont trop limitées ou statiques.

TechRepublic [4] et Wired magazine [5] déclarent que les applications mobiles doivent considérer le contexte pour atteindre la bonne cible et que l'absence de prise en compte de ce contexte est en train de tuer le marché des applications mobiles. Par conséquent, l'engagement contextuel est la meilleure solution capable d'offrir à l'utilisateur une véritable expérience omniprésente qui répond à ses attentes.

Nous avons donc imaginé une nouvelle façon de répondre aux besoins de l'utilisateur d'une manière intelligente et dynamique. Notre objectif, à ce niveau, est de proposer une première approche d'application éternelle qui fournirait les services et les fonctionnalités appropriées, de manière invisible à l'utilisateur, selon ses besoins et dans son contexte actuel. Cette proposition, contrairement aux travaux existants liés à ce domaine, offre une capture de contexte précise qui aide l'application à répondre à l'utilisateur de la meilleure façon possible par orchestration de composants logiciels dynamiques.

Dans cet article, l'objectif principal sera de présenter le mécanisme que nous utilisons pour gérer et capturer les événements liés à l'utilisateur et augmenter ainsi le taux de rétention des applications. C'est également une solution qui permettra à l'utilisateur de profiter de l'intelligence (future) de son environnement sans avoir à se soucier de la façon donc c'est fait.

Le reste de cet article est structuré comme suit : La section 2 commence par présenter un scénario d'exemple. La section 3, indique les défis à relever pour ce travail. La section 4 présente une étude des technologies existantes. La section 5 décrit l'intégration des technologies existantes que nous avons utilisée pour introduire

Ubimob'11, July 5, 2016, Lorient, France.

DOI: http://dx.doi.org/10.1145/12345.67890

une valeur ajoutée par rapport aux travaux existants et faire une proposition solide. Afin de justifier notre proposition, nous montrons, dans la section 6, comment nous avons mis en œuvre notre proposition par un prototype. L'article se termine par une conclusion et quelques remerciements.

2. Scenario

Considérons le scénario simple d'une journée classique avec quelques événements inattendus pour illustrer le comportement de l'application en fonction des besoins de l'utilisateur. La figure 1 présente ce scénario de manière graphique. Dans ce schéma, la lettre L représente l'emplacement, D représente l'heure / date, S la situation et N une notification.

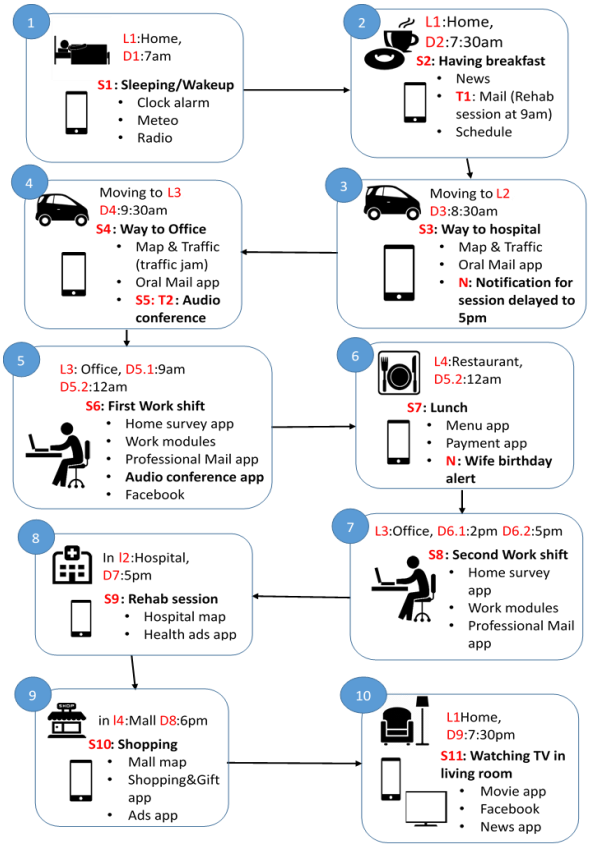

Figure 1. Une journée classique d'un utilisateur

Lorsque l'utilisateur va dormir, l'application «personnelle» déploie l'application d'alarme sur le téléphone. Quand le téléphone sonne, elle lance le module météo et le module radio en ligne pour diffuser les nouvelles du matin. Lorsque l'utilisateur prend son petit déjeuner, il a l'habitude de lire ses mails personnels, c'est pourquoi la fonctionnalité « Emails » est déployée et lancée alors que les composants antérieurs sont retirés silencieusement. Dans un mail, l'utilisateur apprend qu'il a une séance de rééducation à 9h à l'hôpital. Quand il rentre dans sa voiture, le GPS est automatiquement connecté au téléphone, la messagerie est automatiquement améliorée pour intégrer la fonctionnalité vocale de lecture électronique, tandis qu'un guide de la circulation est lancé sur l'écran de sa voiture et lui indique la meilleure façon de se rendre à l'hôpital.

Pendant qu'il conduit, il reçoit une notification lui indiquant que son rendez-vous est reporté à 17 heures. L'utilisateur change de direction pour se rendre à son travail. Comme il est un peu en retard, une audio conférence est dynamiquement déployée afin d'assurer sa participation à la réunion prévue. En entrant dans son bureau, l'audio conférence est migrée vers son PC de bureau et la fonctionnalité vidéo y est ajoutée pour mieux suivre les débats. Les applications relatives à son travail sont également déployées sur son ordinateur.

À 12h l'utilisateur va au restaurant. Dès qu'il rentre, l'application présentant le menu et celle de paiement sont déployées sur son téléphone. Pendant son déjeuner, il reçoit une notification pour l'anniversaire de sa femme. Après manger, il se remet au travail. À 17h heures, l'utilisateur doit se rendre à l'hôpital pour son rendez-vous. L'entrée dans l'hôpital va lancer l'affichage d'un plan de l'hôpital et des annonces médicales. À la sortie de l'hôpital il se dirige vers le supermarché où l'application du supermarché est démarrée. Là, il achète un cadeau pour sa femme avec l'aide de l'application "Shopping & Gift" automatiquement déployée sur son téléphone. En quittant le centre commercial, les composants relatifs au supermarché sont retirés. Enfin, lorsque l'utilisateur rentre à la maison et se trouve dans son salon, une application de film est téléchargée et déployée sur sa télévision tandis que Facebook et News (info) sont déployés sur son téléphone. De plus, tout ce qui concerne son travail est supprimé.

Même si ce scénario semble simple, disposer d'une application dynamique capable de capturer le contexte de l'utilisateur et de s'y adapter soulève un certain nombre de défis.

3. Les défis

Ce type d'application pose des problèmes en raison de leur conception spécifique et de l'absence de méthodologie et d'outils. La multiplicité des configurations, leur hétérogénéité et leur accès constitue notre premier défi.

Même en faisant abstraction de l'hétérogénéité des dispositifs, obtenir les résultats attendus suppose de dépasser trois défis principaux : la détection du contexte permettant de détecter des changements dans le contexte, le changement de contexte qui amène le système dans un nouvel état qui doit être pris en charge et, finalement, la manipulation de contexte qui adapte les possibilités d'interaction dans le contexte actuel. L'évolution dynamique de ces applications nous oblige à repenser la façon dont nous les concevons.

Dans une approche plus technique, la mise en œuvre de ce type d'application constitue une tâche assez difficile. Premièrement, nous devons tenir compte de la vie privée de l'utilisateur qui est, pour la plupart des usagers, le facteur le plus important. Ceci pose un problème aux LLA qui doivent utiliser des informations contextuelles et avoir accès à toutes les ressources de l'utilisateur afin d'en extraire des informations spécifiques et détaillées sur l'utilisateur.

Ces informations doivent être extraites de façon continue la plupart du temps à partir de capteurs (autour de l'utilisateur) qui sont la pierre angulaire de ce genre d'applications. Cela nous amène naturellement à prendre en considération les problèmes de durée de vie de la batterie sur les dispositifs accueillant ces capteurs. Le problème étant de choisir entre la durée de vie de la batterie et l'exactitude de la capture de contexte.

Nous ne pouvons également pas ignorer la complexité des APIs permettant d'accéder à ces capteurs et aux ressources. La variété des façons d'y accéder complexifie la tâche. D'ailleurs, même une fois résolu le problème de l'API et obtenues les données nécessaires, le véritable défi sera d'extraire les informations décisionnelles à partir des données brutes. Bien que l'exploration de données ne soit pas notre objectif, nous devons trouver un moyen simple d'extraire des connaissances utiles à partir de données brutes afin de construire un contexte d'utilisation pouvant aider l'application à réagir.

De plus, la manipulation de contexte soulève également des questions sur la qualité des services proposés et l'exactitude de la proposition. Ces propositions, qui déploieront ou modifieront des services pour répondre aux besoins de l'utilisateur, peuvent ne pas correspondre à ce qu'il attend ou veut. Il est vrai que les utilisateurs sont toujours attirés par ce qu'ils peuvent obtenir facilement, mais l'adéquation est tout de même essentielle. Le déploiement du mauvais service pourrait perturber l'utilisateur et aboutir exactement au contraire de ce que nous cherchons à accomplir.
Enfin, le déploiement dynamique est une tâche délicate car elle implique l'utilisation de composants logiciels mobiles capables de communiquer les uns avec les autres et contrôlés par une plate-forme qui peut les exécuter / arrêter / migrer en cas de besoin. Le problème réside également dans la gestion des composants au niveau de leur exécution (reconfiguration à chaud) qui consiste à modifier ou à migrer un composant alors qu'il est en marche.

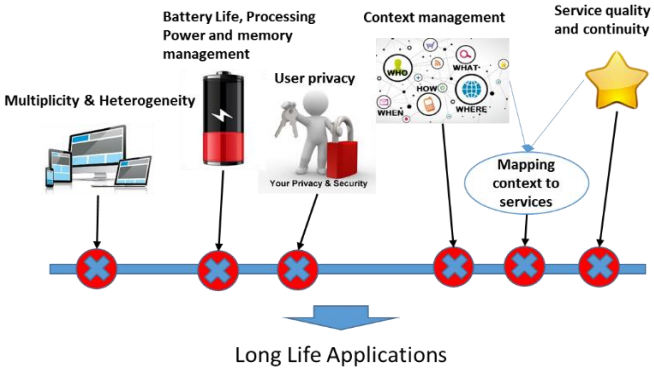

Figure 2. Présentation des défis de notre travail

Les défis sont multiples et les enjeux sont élevés. Nous nous concentrerons maintenant sur la résolution de certains d'entre eux afin d'atteindre nos objectifs. Actuellement, des projets de recherche (section 4) proposent des résultats intéressants (middleware, réseaux, mobilité, etc.) mais ne répondent pas entièrement aux défis clés et, la plupart du temps, sont dédiés à des situations spécifiques qui ne résolvent qu'une partie du problème.

4. Travaux adjacents

Une étude générale des d'applications mobiles à longue durée de vie montre quelques projets et approches intéressantes. Les applications sensibles au contexte ont été utilisées pendant de nombreuses années dans différents domaines comme, par exemple, les musées offrant les visites guidées [6]. Les approches plus récentes et avancées, décrites plus en détail dans [7], sont centré sur les services.

4.1 Les applications sensibles au contexte

Le premier exemple qui vient à l'esprit quand on parle d'applications sensibles au contexte est Google Now [8]. Il est l'une des premières implémentations qui fournit un moyen facile et rapide de trouver des résultats et des services personnalisés basés sur l'emplacement de l'utilisateur et son environnement social.
Choreos [9] met en œuvre un cadre pour le développement de chorégraphies évolutives. L'objectif est de permettre aux experts du domaine de développer des solutions décentralisées sur une échelle ultra-large composée de services hétérogènes. D'autre part, Docker [10] construit et déploie des conteneurs à l'intérieur desquels peuvent être emballées les applications et services afin qu'ils puissent être facilement utilisés en tant que parties d'une application plus vaste et orchestrés en couches dédiées à une utilisation plus large.
Dans le même esprit, Flybits [11] est une application mobile basée sur le Cloud. Quand un utilisateur mobile déclenche une action ou entre dans une zone, le contenu de l'application et/ou services récurrent, selon ce contexte, à un outil qui fournit des services plus adaptés comme le font les systèmes de recommandation [12].
La deuxième partie de notre recherche sur les travaux existants sera axée sur les systèmes qui peuvent offrir une ré-adaptabilité et une sensibilité au contexte. Les LLA sont basés sur une architecture dynamique à composants qui doit changer à chaque instant pour suivre les évolutions du contexte. Ainsi apparaît la nécessité d'un système dynamique capable de détecter le contexte, d'évaluer les conditions et de prendre des décisions. Le système le plus adéquat pour couvrir tous ces aspects est le mécanisme de règles « Event Condition Action » (ECA) que nous allons présenter dans la section 4.2.

4.2 Approche ECA

Cette approche permet, pour chaque situation, de déclencher un événement. Cet événement sera associé à une action après vérification de certaines conditions modélisées par l'architecte du service dans un langage spécifique que nous définirons dans nos futurs travaux.
L'utilisation de règles ECA donnera à notre application la flexibilité nécessaire pour saisir le contexte et le manipuler.
Les règles ECA sont utilisés pour obtenir ce genre de comportement réactif dans de nombreux milieux, incluant les bases de données actives, la gestion des flux de données (publication / abonnement) et la mise en œuvre de processus métiers. Elles spécifient le comportement réactif souhaité et peuvent être définies manuellement par les concepteurs ou générées automatiquement.
[13] propose un environnement d'exécution d'application qui permet aux développeurs d'applications sensibles au contexte d'utiliser des journaux (Logs), sur les différents appareils, de sorte que le contexte de l'utilisateur puisse être accessible. Leur travail aborde le problème de permettre aux développeurs d'applications de créer facilement des algorithmes utilisant la détection du contexte de l'utilisateur. Plus clairement, ils proposent une spécification de règles ECA qui satisfait aux exigences fonctionnelles de couverture des dimensions sémantiques et de la manipulation des données collectées à partir des capteurs des smartphones.
La contribution de ce travail est de permettre à un développeur de définir un algorithme pour générer le contexte primaire (l'information contextuelle de base) souhaité. Ceci est réalisé en utilisant, pour traiter l'événement et les conditions, des fonctions d'analyse des données du journal (Log) et en fournissant des balises d'action incluant des fonctions de manipulation du contexte primaire.
CybreMinder [14] est basé sur l'infrastructure Context Toolkit, il prend en charge les utilisateurs en envoyant ou recevant des rappels, par exemple des e-mail, des messages vocaux ou des SMS qui peuvent être combinés pour décrire complètement les situations incluant le temps, le lieu et d'autres éléments du contexte. Le principal mérite de cet outil est la facilité qu'il offre aux utilisateurs. Il leur permet de spécifier, par un éditeur graphique, les situations et les spécifications de contexte associées à un rappel.
Le problème majeur, cependant, est que le seul type d'action qui peut être déclenché à la suite d'une situation est une notification. Aucun autre type d'action ne peut être déclenché pour l'utilisateur.
EventManager [15] poursuit un objectif similaire à CybreMinder. Il présente aux utilisateurs une interface graphique de gestion d'événements avec laquelle ils peuvent définir les spécifications des événement-action. Ces spécifications sont limitées à la configuration suivante : lorsque <personne> est / sont <relation>

<emplacement> alors <action>. Cela signifie que l'utilisateur est limité à émettre des notifications liées à la localisation personnelle ; aucune autre source de contexte n'est acceptée.
Les LLA, tels que nous les définissons (Scalables, continus etc.), doivent être distribués afin de couvrir tous nos critères. Par conséquent, l'approche ECA doit suivre la même logique de distribution. Dans le tableau 1, nous présentons les différences entre les applications présentées ci-dessus d'un point de vue centralisé / distribué sur les trois éléments de l'ECA.

Table 1. Différences entre les ECA étudiés

	Détection d'évènement	Évaluation de Condition	Action
[13]	distribué	centralisé	centralisé
[14]	centralisé	centralisé	Non existant
[15]	centralisé	Non existant	centralisé

Le tableau 1 montre que ces outils utilisent des techniques différentes pour arriver au même but : la construction d'une application reconfigurable et intelligente. Il est clair qu'ils ne suivent pas tous le modèle ECA à la lettre. Notre point de vue de l'application nous a conduits à choisir une détection d'événement distribuée afin de gérer toutes les informations qui pourraient être importantes pour le contexte indépendamment de leur source. Nous avons opté pour une Évaluation de Condition hybride (à la fois distribuée et centralisée) dans laquelle seule la décision est centralisée sur le périphérique principal de l'utilisateur (par exemple, son ordinateur, son téléphone, etc.). Toutefois, si ce périphérique n'est pas disponible pour une raison quelconque, l'architecture de notre système étant modulaire et mobile, elle sera en mesure de réagir à des problèmes inattendus et de migrer le processus d'évaluation vers un autre appareil. Ce choix constitue notre contribution à l'approche ECA qui est destinée à éviter les surcharges et la redondance des décisions d'adaptation tout en assurant la dynamicité et la continuité du système.
Néanmoins notre proposition dépasse les travaux précédents en mettant en œuvre un mécanisme d'action distribué qui donnera à l'application de la flexibilité et la dynamique nécessaires à la faire fonctionner au sein du domaine de l'utilisateur.

4.3 Les middlewares qui supportent le développement à base de composants

La dernière partie de notre étude concernera la gestion de la distribution et de la dynamique. Pour cela nous avons recours à une plate-forme capable de gérer des composants logiciels à la volée (en cours d'exécution) décrite plus en détail dans [5].
Dans ce contexte, de nombreuses plates-formes ont été créées, chacune apporte quelque chose de nouveau dédié à sa propre vision de la distribution du middleware. OSGi [16] est l'une des premières plates-formes qui met en œuvre la distribution en utilisant un modèle de composants connectés par des services et des paquets.
CADeComp [17] et FRASCATI [18] fonctionnent sur une architecture offrant un contexte d'adaptation par la gestion des composants d'une manière flexible.
Kalimucho [19] est considérée comme l'une de ces plates-formes car elle offre une autre vision des applications distribuées en les décomposant en éléments indépendants déployables dynamiquement (en cours d'exécution). Kalimucho convient aux applications modulaires à base de composants logiciels distribués. Cette modularité offre des solutions garantissant la continuité et la durabilité des applications au fil du temps. Ce middleware sera le support sur lequel les composants fonctionneront et interagiront les uns avec les autres. Après avoir présenté la plupart de nos défis et l'étude des travaux existants autour de notre thème, nous proposons notre approche dans la section 5.

5. Approche proposée

Nous présentons dans cette section notre proposition de LLA. La combinaison de nombreux mécanismes et leur intégration crée le noyau de notre application. En fait, l'application façade (front end) sera le résultat de l'assemblage des composants logiciels déployés sur le dispositif (ou des dispositifs) de l'utilisateur selon ses besoins. Mais derrière ce que l'utilisateur voit il y a une combinaison de processus et de modules travaillant simultanément à lui offrir les résultats recherchés.

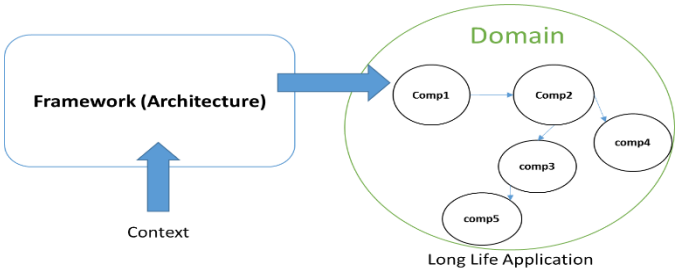

Figure 3. Vue globale sur notre approche

Comme on le voit sur la figure 3, la démarche commence par la collecte des informations de contexte. Ces informations aideront le Framework et le noyau de notre application (Back end) à déployer les composants (Comp1, Comp2, Comp3 etc.) qui formeront l'application (Front end) dans l'environnement de l'utilisateur.
L'architecture proposée répond aux défis les plus critiques (capture, modification et manipulation de contexte) indiqués dans la section 3. L'entrée du contexte sera gérée par événements.
Les parties qui suivent donnent des définitions aux concepts énoncés dans la Fig. 3. En d'autres termes ce que nous entendons exactement par Domaine, Contexte, Composants et Architecture.

5.1 Formulation du Contexte

L'application, telle que nous l'avons définie, est un ensemble de situations qui se présentent à l'utilisateur alors qu'il évolue dans son environnement. La somme de ces situations crée l'image de l'application. L'entité principale appelée « Situation » est le facteur déterminant le comportement de l'application. Le paragraphe suivant présente cet élément clé de notre proposition. Cette entité sera notre représentation du contexte de l'utilisateur et notre réponse au défi de gestion de contexte.
La situation est l'élément clé dans notre système. Nous l'envisageons comme une projection sur trois axes. Une situation est la combinaison de temps, de lieux et d'activités. Nous avons estimé que nous avions besoin de ces axes afin d'exploiter la puissance du contexte et de personnaliser notre système d'adaptation. Donc, l'application sera un ensemble de situations d'interaction avec l'utilisateur et son environnement qui représente ses besoins et sa vie quotidienne.
Mais avant d'entrer dans la description d'une situation, nous devons définir quelques concepts et variables clés liés aux trois axes de nos situations.

5.1.1 Definitions des concepts

- **Activity** est la tâche que l'utilisateur est en train de faire.
- **Primitive** est une opération atomique sur un axe donné (par exemple, before, while et after pour l'axe Temps).

- **Basic situation** est une situation définie par des primitives minimales (3 au maximum). Elle peut être de 1er niveau, de 2ème ou 3ème niveau selon le nombre de primitives utilisées.
- **Situation Priority** est la priorité d'une situation donnée.

5.1.2 Variables

- **Date**: Paramètre représentant l'aspect temporel (Année, Mois, Jour, Heure, Minute, Seconde).
- **DTPrecision**: Précision sur la date et l'heure qui permet aux primitives d'accepter une marge de tolérance.
- **Location**: Paramètre représentant l'aspect spatial (Longitude, Latitude, Zone, Ville etc.)
- **LPrecision**: Tolérance sur la localisation.
- **Agenda**: Calendrier de l'utilisateur qui contient toutes ses taches planifiés.

5.1.3 Situations de premier niveau

Le premier niveau est celui où se situent les situations les plus fondamentales et générales qui n'impliquent que l'un des axes et l'une des primitives sur cet axe **.**

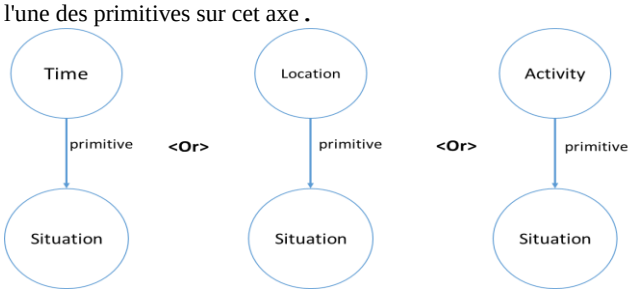

Figure 4. Modèle du « 1st Level situation »

Les tableaux suivants (tableau 2, tableau 3 et le tableau 4) décrivent les primitives associées à chaque axe.

Table 2. Primitives temps

Primitive	Paramètres	Type de retour	Description
Before	Date, DTPrecision	boolean	Renvoie vrai si la date indiquée dans les paramètres est avant la date actuelle
After	Date, DTPrecision	boolean	Renvoie vrai si la date indiquée dans les paramètres est après la date actuelle
While	Date, DTPrecision	boolean	Renvoie vrai si la date indiquée dans les paramètres est égale la date actuelle (à la tolérance près)

Table 3. Primitives de localisation

Primitive	Paramètres	Type de retour	Description
Inside	Location, LPrecision	boolean	Renvoie vrai si l'utilisateur est à l'intérieur d'un lieu
Outside	Location, LPrecision	boolean	Renvoie vrai si l'utilisateur est en dehors d'un lieu

Table 4. Primitives d'activité

Primitives	Paramètres	Types retour	Description
Free	Agenda	boolean	Renvoie vrai si l'utilisateur est libre
PlannedTask	Agenda	boolean	Renvoie vrai si l'utilisateur a une tache planifiée prévue
UnplannedTask	Agenda	boolean	Renvoie vrai si l'utilisateur a une tache non planifiée prévue

5.1.4 Situations de deuxième niveau

Le deuxième niveau est le niveau dans lequel nous combinons deux axes pour représenter une situation. Cette combinaison des axes offre plus de primitives et des situations plus spécifiques et plus riches.

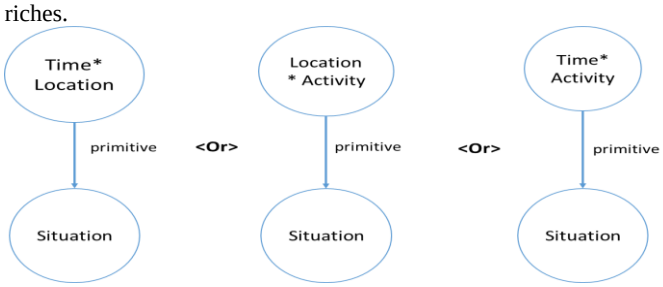

Figure 5. Modèle de « 2nd Level situation »

Les nouvelles primitives associées à la combinaison de deux axes sont les suivantes.

Table 5. Primitives Temps*Activité

Primitive	Paramètres	Type de retour	Description
almostStart	Date, DTPrecision, Task	boolean	Renvoie vrai si la date est proche de l'heure de début de la tâche
almostFinish	Date, DTPrecision, Task	boolean	Renvoie vrai si la date est proche de l'heure de début de la tâche

almostStart et almostFinish sont deux nouvelles primitives que nous avons créées en combinant temps et activité. Elles nous aideront à prédire et gérer le comportement de notre application en sachant que certaines activités sont sur le point de démarrer ou de se terminer.

Table 6. Primitives Temps*Localisation

Primitive	Paramètres	Type de retour	Description
Closer	Date, DTPrecision, Task	boolean	Renvoie vrai si l'utilisateur se rapproche d'un emplacement

Further	Date, DTPrecision, Task	boolean	Renvoie vrai si l'utilisateur s'éloigne d'un emplacement

La combinaison du temps et du lieu apporte une meilleure compréhension de la façon dont l'utilisateur évolue dans son environnement. Nous pouvons détecter, grâce aux nouvelles primitives Closer et Further si l'utilisateur se dirige quelque part ou de quitte un endroit.

Table 7. Primitives Localisation*Activité

Primitive	Paramètres	Type de retour	Description
Opportunity	Location, LPrecision, Task	boolean	Retourne vrai s'il est possible de faire une certaine tâche à cet endroit

Cette nouvelle primitive indique si un emplacement peut potentiellement être intéressant pour l'utilisateur pour faire l'une de ses activités.

5.1.5 Situations de troisième niveau

Ce niveau est le plus riche pour exprimer une situation puisque nous y combinons tous les axes afin de placer la situation dans une couverture contextuelle précise. Le diagramme en Fig. 6 montre comment nous pouvons construire une situation en combinant les primitives énoncées précédemment de toutes les façons possibles.

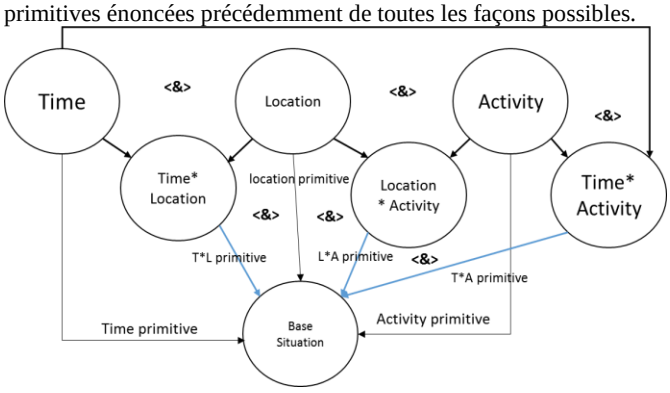

Figure 6. Modèle du « 3rd Level situation »

La figure 6 illustre les différents états que la situation peut prendre en combinant des primitives sur les trois axes différents.

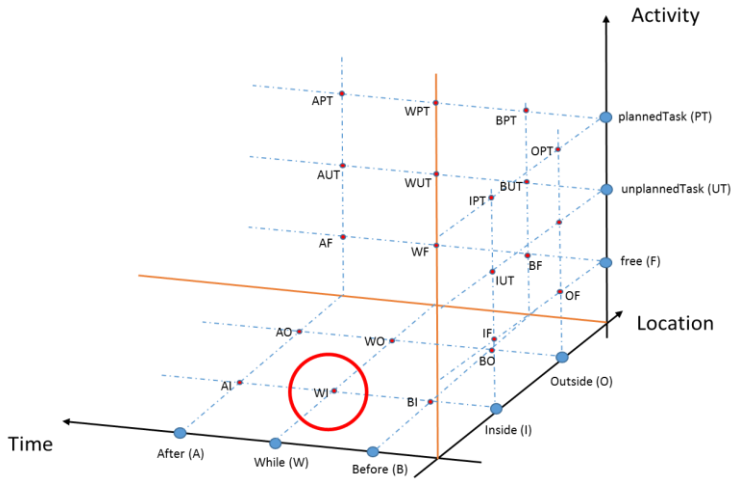

Figure 7. Représentation des combinaisons des situations, axes et primitives

Nous allons utiliser ce modèle pour représenter toutes les situations. Par exemple, imaginons une situation S (réunion de travail) qui se produit lorsque l'utilisateur est à l'intérieur d'un lieu appelé Université avant 10h alors qu'il a une tâche planifiée appelée réunion. Nous considérons un DTPrecision de 5 minutes et un LPrecision de 10 mètres. Le modèle serait celui présenté sur la Figure 8.

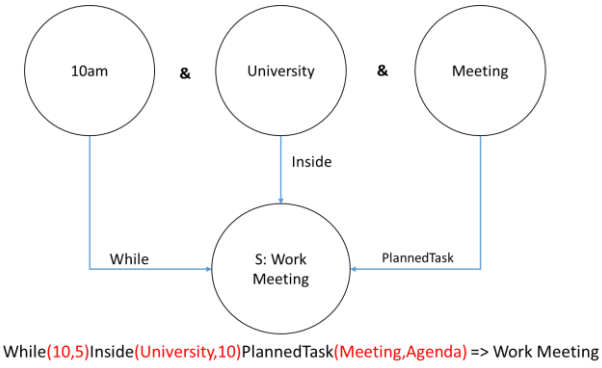

Figure 8. Exemple de situation spécifique

5.1.6 Processus d'identification de situations

Le processus d'identification est le mécanisme qui surveille les axes et crée les situations en utilisant des données recueillies autour de l'utilisateur et en les combinant avec les primitives.

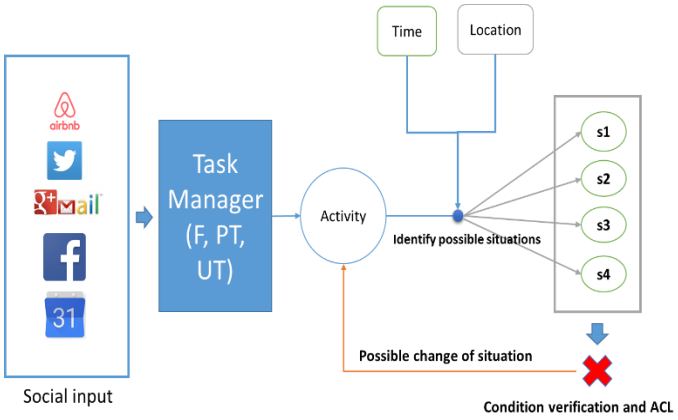

Figure 9. Processus d'identification de situations

Notre gestionnaire de tâches est un « thread » continument en cours d'exécution. Il extrait des données de réseaux sociaux car c'est la source primaire des événements et des activités sociales de l'utilisateur et la source la plus possible d'informations à disposition de l'utilisateur. Après organisation de ces tâches et déduction du temps libre de l'utilisateur, ces informations nourrissent l'axe d'activité. En parallèle, nous identifions les situations possibles en surveillant le Temps et la Localisation de l'utilisateur. Le choix de la situation constitue alors la prochaine étape.

Cette approche par manipulation et détection de contexte est la clé de l'architecture de notre travail. La partie suivante présente la technologie qui est derrière notre déploiement dynamique et la façon dont elle fonctionne et relie les composants afin d'obtenir le comportement de notre application.

5.2 Le corps de l'Application (Architecture)

Après avoir défini théoriquement les concepts entourant notre travail, nous allons décrire dans cette section l'architecture de base de notre système de ré-adaptation. Afin de généraliser la distribution, cette architecture est elle-même composée de plusieurs composants logiciels [19] reliés les uns aux autres et supportés par la plateforme Kalimucho. Cela signifie que notre architecture est flexible et exécutable quel que soit le dispositif. Du moins tant que les dispositifs, dans le domaine de l'utilisateur, sont disponibles et supportent la plateforme Kalimucho. Ceci justifie nos choix liés à la distribution / centralisation présentés dans la section 4. L'architecture en Fig. 10 n'est qu'une partie de l'architecture complète : celle qui gère le contexte et les situations.

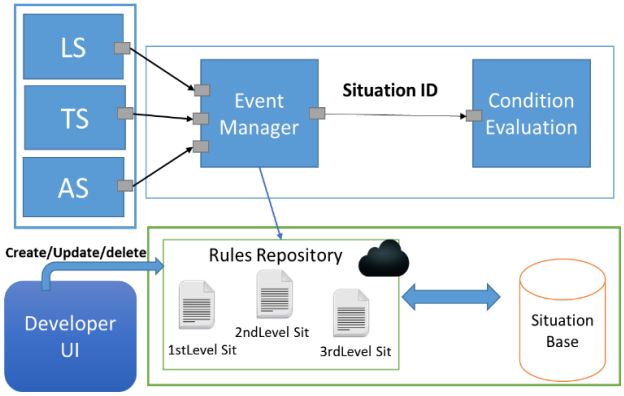

Figure 10. Les modules de capture de contexte et d'identification de situations dans notre architecture

LS (Capteur de localisation), TS (Capteur de temps) et AS (capteur d'activité) sont les trois éléments qui agissent à titre d'écouteurs ou de capteurs fournissant à l'Event Manager des notifications sur les changements sur les trois axes.

L'UI Développeur est un outil externe fourni aux développeurs qui leur permet de créer des situations précises et de les injecter en tant que règles. C'est le lien entre le contexte et la stratégie de déploiement. Les situations ainsi créées sont introduites dans le référentiel des règles comme un ensemble de règles. Ce module est l'une des solutions qui nous aideront à surmonter le défi de la cartographie entre les services et le contexte capturé. D'autres solutions seront étudiées dans nos futurs travaux.

Le Repository de Règles est le support qui contient toutes les règles à toutes les situations possibles selon la logique de notre modèle (voir A. Formulation de contexte). Il divise les règles en trois fichiers différents correspondant à chacun des niveaux de situation. Les développeurs peuvent mettre à jour ce référentiel en ajoutant / modifiant / supprimant des situations via l'interface utilisateur qui génère les règles en respectant le format que nous avons défini. Ces règles sont stockées sue le Cloud parce qu'elles doivent être partagées et synchronisées entre les utilisateurs et les développeurs.

La Base de situations est une base de données enregistrant toutes les occurrences détectées de situations qui peuvent contribuer à comprendre le comportement de l'utilisateur et à proposer des adaptations précises.

Le gestionnaire d'événements est un composant fonctionnant continuellement afin de surveiller le Temps, la Localisation et l'Activité de l'utilisateur. Ce module est celui qui illustre notre flux de données (voir Fig. 9) en supervisant le référentiel de règles et en essayant de détecter l'apparition d'une situation. Il constitue notre contribution principale par rapport au système ECA. Nous basons la détection d'événements non seulement sur les notifications et alertes mais également sur une réelle compréhension du contexte de l'utilisateur.

Bien qu'il ne soit pas dans notre objectif dans cet article d'illustrer la partie de déploiement (Domaine de l'utilisateur + Composants) de notre travail, nous allons présenter brièvement le reste du domaine de l'utilisateur. Il sera intégré et entièrement expliqué dans nos travaux futurs, ainsi que le mécanisme ECA.

5.3 Domaine de l'utilisateur

La nature des LLA provoque l'obligation de gérer plusieurs appareils en même temps. Ces dispositifs ont besoin de communiquer indépendamment des différences de matériel et de logiciel.

Afin de mettre notre application dans l'environnement adéquat, nous devons définir le domaine d'utilisation (l'environnement physique) où le composant sera en mesure de fonctionner. Kalimucho sera le middleware de communication entre les appareils et composants. Il sera également notre outil de découverte de périphériques. Les instances de la plateforme existant sur les périphériques sont capables de se découvrir et de communiquer même au travers de réseaux hétérogènes. Cela répond à notre défi de gestion de la multiplicité et de l'hétérogénéité des dispositifs [19].

Le domaine est constitué des appareils auxquels l'utilisateur peut accéder. Toute ressource autour de l'utilisateur peut être un hôte possible ou une source d'information importante. Notre application utilise non seulement les appareils mobiles mais aussi tous les appareils ayant une capacité d'interaction (TV, haut-parleurs, Ordinateur. Etc.) ainsi que les appareils qui ne produisent que des informations de contexte comme les capteurs. Ce domaine constitue l'hôte des composants logiciels qui évoluent dynamiquement afin de masquer l'hétérogénéité et de gérer la multiplicité des périphériques.

6. PROTOTYPE

Notre prototype réalise le scénario décrit dans la section 2. Il fait appel à de nombreuses compétences différentes liées au développement et / ou à l'utilisation de middleware, aux interactions, au génie logiciel, à la gestion de profils d'utilisateurs, à la sémantique, à la représentation des connaissances et au raisonnement, au traitement de requêtes en temps réel, à l'informatique mobile, à l'Internet des objets, etc.

Sur la base de notre modèle de contexte (Contexte Formulation) le scénario est représenté à la Fig. 11.

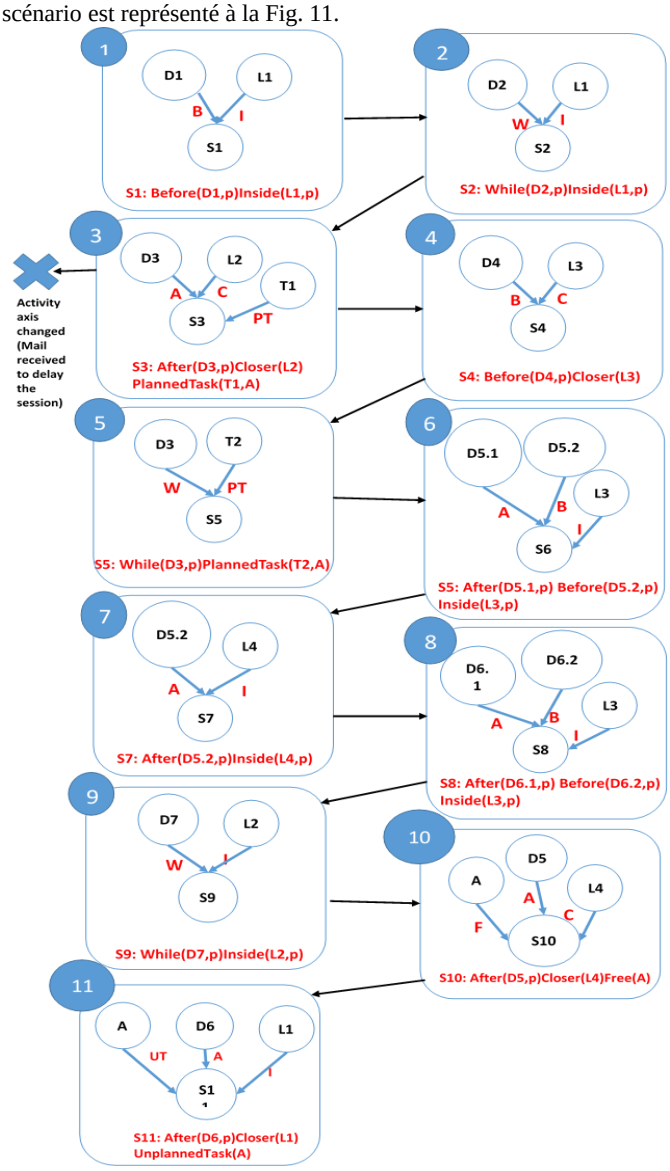

Figure 11. Scenario représenté par un modèle de situations (voir Fig. 8)

Cette modélisation mise en œuvre dans notre architecture nous a permis d'obtenir des résultats prometteurs validés par un premier prototype web dédié à illustrer l'avancement du travail (détection des événements et capture du contexte). Notre prototype est basé sur trois grandes parties. La première est un simulateur qui remplace les capteurs (voir Fig. 10). Il est utilisé pour simuler le comportement de l'utilisateur sur les trois axes. L'utilisation de ce simulateur est la meilleure façon de tester les changements contextuels sans avoir à vraiment respecter les axes en particulier de Temps et de Localisation. La partie suivante est l'interface utilisateur du développeur qui permet de définir diverses situations. La dernière partie, la plus importante, est le gestionnaire d'événements qui applique notre mécanisme de détection d'événements (voir Contexte Formulation). Il respecte notre modèle de situations en utilisant les informations produites par le simulateur afin de capturer le contexte de l'utilisateur. Les captures d'écran suivantes montrent les principales interfaces de notre prototype.

Simulateur de situation: il permet de simuler l'état actuel de l'utilisateur sur les trois axes.

Figure 12. Interface de simulation du contexte utilisateur

Cette simulation permet de découvrir les événements à venir.

Figure 13. Interface présentant les situations possibles identifiées

Résultats: cette interface montre les situations actuelles de l'utilisateur en donnant des détails sur chaque situation et en montrant les activités à venir (toDoList).

Figure 14. Résultats montrant les situations courantes

Une vidéo de ce prototype sera disponible sur : www.iutbayonne.univ-pau.fr/~roose/pub/pisco/pisco.avi mais aucune évaluation n'est disponible. L'évaluation de notre prototype est une tâche future qui demande de déterminer les critères adéquats d'évaluation et de les mettre en perspective avec les applications présentées dans l'état de l'art (voir section 4.1).

7. CONCLUSION

L'avenir dans lequel nos applications sauront ce dont nous avons besoin avant même que nous n'y ayons pensé, est-il utopique ? Peut-être mais l'opportunité est trop importante et le potentiel trop fort pour arrêter ce phénomène.

Il s'agit d'une véritable percée technologique que de proposer une seule application à utilisateur pour effectuer toutes ses tâches et gérer de manière transparente toutes les fonctionnalités et modules supplémentaires qu'il peut exiger. C'est une situation idéale non disponible de nos jours. Une telle application va évoluer pendant son exécution en fonction des besoins des utilisateurs (personnel et / ou professionnel) et fournir des services et des données pertinents en continu et adaptés au contexte.

Dans cet article, nous avons couvert la partie concernant les événements de notre système ECA tout en expliquant la logique derrière notre proposition. La prochaine étape de notre travail sera d'étudier les parties Condition et Action pour compléter notre architecture. Nous les validerons en mettant en œuvre l'architecture proposée et l'intégration des technologies abordées dans notre prototype qui nous aideront à tester les limites de notre proposition et à réévaluer nos idées .

8. REMERCIEMENTS

This work was supported by the CICYT project TIN2013-46238-C4-4-R and DGA-FSE. First Page Copyright Notice

9. REFERENCES

[1] "Foursquare's Swarm And The Rise Of The Invisible App" Matthew Panzarino, http://techcrunch.com/2014/05/15/foursquares-swarm-and-the-rise-of-the-invisible-app/. 2014 [Accessed on 14/06/2015]

[2] "The future is without apps" Donney Reynolds https://medium.com/fwd-thoughts/the-future-is-without-apps. 2015 [Accessed on 07/05/2016]

[3] 16 mobile mistakes that plummet user retention rate, Kendrick Wang http://apptimize.com/blog/2015/10/16-mobile-mistakes-that-plummet-user-retention-rates/.2015 [Accessed on 9/04/2015]

[4] "Mobile apps need context to hit the right targets", Scott Matteson, http://www.techrepublic.com/article/mobile-apps-need-context-to-hit-the-right-targets/. 2015 [Accessed on 10/06/2015]

[5] "Why lack of context is killing your mobile app" posted by AshishThusoo on Wired Magazine http://www.wired.com/insights/2013/12/why-lack-of-context-is-killing-your-mobile-app/. 2015 [Accessed on 20/10/2015]

[6] D. Raptis, N. Tselios and Nikolaos Avouris, Context-based design of mobile applications for museums: A survey of existing practices, In MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services, 2005, 153—160, ACM press.

[7] R. Karchoud, P. Roose, M. Dalmau, I. de Courchelle, P. Dibon, Kalimucho for smart-* one step towards eternal applications, Industrial Technologie (ICIT), IEEE International Conference, pp 2426-2432, Seville, 17-19 March, 2015.

[8] Android quick start guide android 5.0 lollipop, 2014.

[9] Amira Ben Hamida, Fabio Kon, Gustavo Ansaldi Oliva, Carlos Eduardo Moreira Dos Santos, Jean-Pierre Lorré, Marco Autili, Guglielmo De Angelis, Apostolos Zarras, Nikolaos Georgantas, Valérie Issarny, Antonia Bertolino. An Integrated Development and Runtime Environment for the Future Internet, Lecture Notes in Computer Science Volume 7281, pp 81-92, 2012.

[10] The Docker book, James Turnbull August 4, 2014 Version: v1.0.7.

[11] The easy way to deliver personalized Mobile experiences. Flybits.inc Palo alto, USA, 2015www.Flybits.com [Accessed on 04/01/2016]

[12] M. Böhmer, G. Bauer, and A. Krüger, "Exploring the design space of context-aware recommender systems that suggest mobile applications," in Proceedings of the 2nd Workshop on Context-Aware Recommender Systems (CARS '10), pp. 1–5, Barcelona, Spain, September 2010.

[13] T. Nakagawa, C. Doi, K. Ohta, and H. Inamura, Customizable Context Detection for ECA rule-based Context-aware Applications, ICMU, May 2012.

[14] Dey, A. K., & Abowd, G. D. (2000a). CybreMinder: A context-aware system for supporting reminders. Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC2K). Heidelberg, Germany: Springer-Verlag.

[15] McCarthy J.F., Anagnost T.D. "Event Manager": Support for the Peripheral Awareness of Events".

[16] Neil Bartlett, OSGi In Practice, December 17, 2009.

[17] Dhouha Ayed, Nabiha Belhanafi, Chantal Taconet, Guy Bernard. Deployment of Component-based Applications on Top of a Context-aware Middleware - The IASTED International Conference on Software Engineering (SE 2005) - Innsbruck, Austria - February 15-17, 2005.

[18] Lionel Seinturier, Philippe Merle, Romain Rouvoy, Daniel Romero, Valerio Schiavoni and Jean Bernard Stefani, A component-based middleware platform for reconfigurable service-oriented architectures, Softw. Pract. Exper journal vol 42 N°5, 559-583, 2012.

[19] Keling Da, Marc Dalmau, Philippe Roose – Kalimucho : Middleware for Mobile Applications – ACM SAC 2014 – pp. 413-419 – ACM Press –24-28/03 – Gyeongju, Korea, 2014.